|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
|  |  | |
|  | LitU  White Paper | |
|  | |  | | --- | | Opening Literary Doors | | |
|  | |  |  | | --- | --- | |  |  | | |

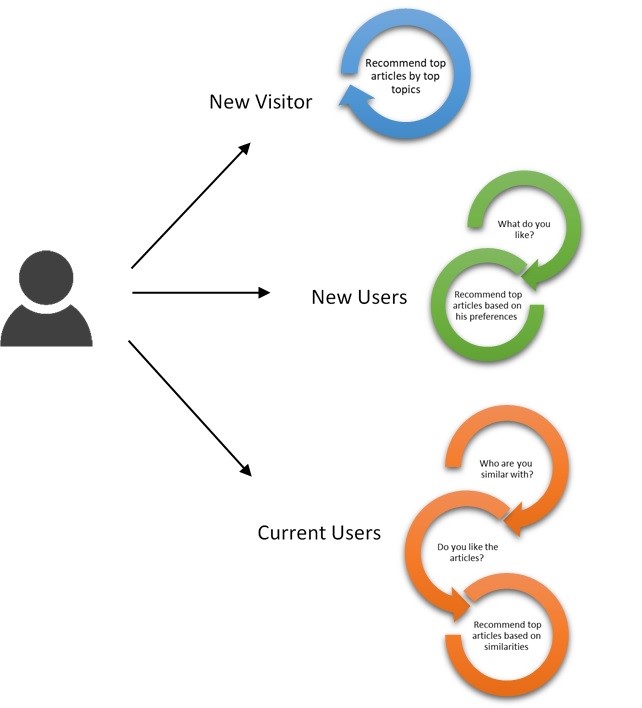
**EXECUTIVE SUMMARY**

The idea of LitU is to help professionals and academics find trustworthy articles, papers, reports and the like. We want people to find the wisdom they desire without having to search through thousands of irrelevant contents. To that end, LitU has developed a Quora-like hybrid recommendation system which not only provides relevant content but also avoids creating a filter bubble so that users can keep on acquiring new wisdom.

LitU’s system has several layers, each one serving a specific purpose:

**Content-Based Recommender:** Our initial plan was to build a collaborative filtering system for our users. We soon noticed that such a system did not solve the cold start problem: what do we recommend to a user that has only just joined the website? We then came up with the idea of using a content-based recommender system as the first layer on the platform. We would ask a new client for topic preferences and recommend articles based on a proprietary ranking algorithm.

This system not only took care of giving new clients articles based on their preferences, we could also use the rankings to provide non-personalised recommendations of top articles within different categories (topics) when clients first came to our website (even before they announced their preferences).



**Collaborative Recommender:** This would get users using our service but how would we calculate user similarities for our Collaborative Recommender? We decided that the best thing to do is to let the user decide for herself. Once a user has read one of our recommendations from the content-based filtering system, we would encourage them to give the content a like (1). If the user did not give it a like, we would assume she did not like what we provided (0). We would then use this information for our entire client base to build collaborative recommendations.

**Randomizer:** All these steps would take care of the relevance problem. We then needed to find a solution to the filter bubble because we wanted our users to also find new wisdom. We realised that we had to add a randomizer to the platform. Rather than providing only the top recommendations based on our collaborative filtering algorithm, our system would also provide an “articles you may like” section with top content from our ranking algorithm in areas / topics not in the users list of preferred topics. This would encourage users to read new content but more importantly, over time, if we see that a user is consistently giving likes (1) to a certain category of randomised articles, we could use that to update the users preferred topics without having to explicitly ask time and again.

**litu for new users**

New users, including those with accounts on LitU and those that simply landed on our page, had to be provided with some form of **non-personalised recommendations**. Why? Well we have little to no information on them so it’s impossible to personalise anything. In the industry, this is done by providing the “top” content on the platform, based on a ranking. This could be top viewed, top read, top liked. In our case, this is done by using a proprietary rankings formula that is described below.

For those users that create accounts we have a slight advantage. If we ask these users to select their preferred topics from a “Topic Checklist” (similar to those provided by StumbleUpon, Quora i.e.), we could filter out items in topics outside the user’s interests using a **content-based recommendation system.** We could then provide to users only the top content from the ranking formula only for the topics the user is interested in.

**ranking formula**

For the non-personalised (semi-personalised in the case of a user who has signed up recently), we need to define what articles to recommend, curating the best content using a formula that considers four main features about the items in order to curate the content. These are:

1. The number of well-respected journals and papers within the topic that the item has been published in. This is what we consider to be “first hand” information and an example would be The Journal of Finance
2. The number of citations the item has received (from authors not including the one that wrote the item) in journals and papers within the same topic. This is what we consider to be “second hand” information and an example would be the number of citations provided by Google scholar.
3. The number of citations that the general public gives to this item. This could be newspapers, books, journals in other topics etc.
4. The date the paper was written. The significance of this is described in the following sections.

POTENTIAL ISSUES

One way of building the rankings would be get the average of the number of publications, ratings and citations for each item. By doing so, we might have been able to get a number for the “popularity” of each item. However, this could cause a few problems:

1. The number of publications in journals are usually much lower than the number of citations by other authors or by newspapers and the like. As an article gets published in more journals, it gets better known and receives more citations and references.
2. By averaging, we would be giving the same weight (importance) to a citation in a newspaper than to a journal, which should not be the case.
3. Older articles naturally have had more time to be reviewed so the publications and citations they receive will usually be higher than for newer articles. This penalises newer articles and increases the likelihood that a user will only be recommended older articles.

SOLUTIONS

To account for the issues mentioned above, we devised two weighting methodologies for the features:

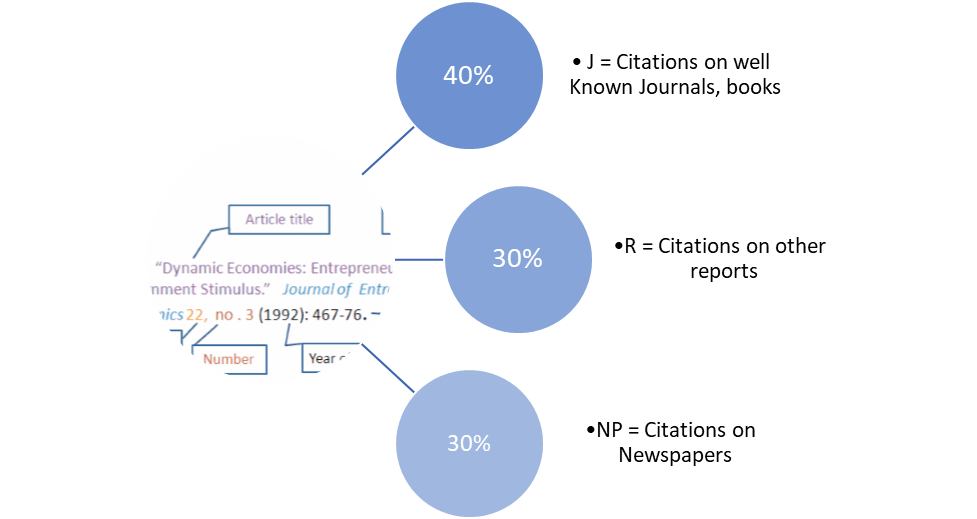
1. The first is a time weight. As explained above, on the one hand, older articles will normally have higher publications, citations and references. On the other hand, an article that, for example, was published in 1920 could be referenced by other authors in 1950, 60, 70 and published in newspapers as late as 2019.

How can we account for these discrepancies? For the former, consider the current date to be 2019 and the date of publication to be1950 i.e 69 years have passed since the publication of the article. We divide the number of citations by the number of years (in the case that publication is same as current year, it would be considered as 1). For example, if we consider the number of citations to be 50, the weighted citations would come out to be 0.72 for an article that is 69 years old and 2.63 for an article that is only 19 years old. In that sense, we are weighting articles by the time-factor.

What happens then in the latter case? Say an article has 5 citations each in the years 2000, 1990 and 1975. Using our formula, the weight for each of the citations will be 0.26, 0.17, 0.11. The citations are now “weighted and ordered” by the time-factor. The sum of these weighted citations (0.54 in this case) will be the time-weighted citations for the item.

Another advantage of having items time-weighted is that if an item, say published in 1920, only becomes relevant in 2010 (think Bayesian probabilities relevance to Machine Learning), the time factor would be able to track this change.

1. The second problem we have is related to the importance. We somehow needed to give importance to journals because they are the most credible source for high quality content. We do this by further weighting the time-weighted citations and references by importance. We have defined the weights as follows:
2. First-hand information receives the highest weight : 40%
3. Second-hand information receives the second highest weight: 30%
4. Third-hand information receives the lowest weight: 30%.



By further weightings the numbers we account for both higher importance being given to higher quality sources and for the fact that there will usually be higher number of references for newspapers than citations and more citations than journal publications.

The weighted citations can then be built as follows:



Once the weighted citations are calculated for all the articles in the database, they can be ordered and the top ones can be provided as recommendations.

**litu for existing users**

After users interact and use the platform, the approach to generating recommendations changes to a **Personalized** **Collaborative Filtering Recommender.** This increases the likelihood of a content being relevant to the user.

In order to do so, we had to find a way of calculating a similarity measure with the rest of the users. If done correctly, the personalised recommender would increase our platform usage by keeping users engaged with relevant content.

We decided to **ask the users whether** he liked the recommendations given by the platform using the ranking formula when he first signed up. With this explicit binary rating (0, 1), we could build users vs articles matrix in order to find neighbour users and recommend based on items liked by the user’s closest neighbours (ones which the user has not yet read).

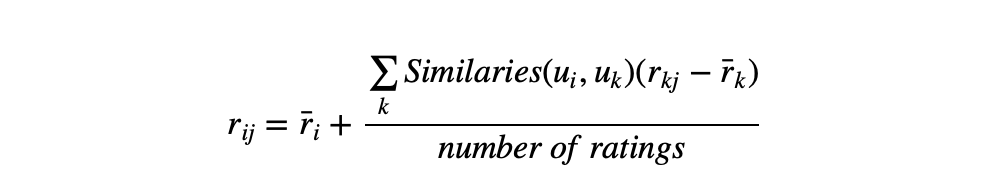


NEIGHBOURHOOD SELECTION STRATEGY

The key for a successful user-based collaborative filter is the strategy using which users are selected in order to compute similarity. Considering that normally 25 to 100 users are used to find similarity, our strategy is to use 50 users in our matrix, split as follows:

* 35 with a similarity > 0.75 (using the Pearson Correlation)
* 15 random users, to include some random recommendations to give try to avoid the filter bubble and expose users to new content that may be relevant to them.

After generating these matrices, it is only a matter of using the formula below to generate personalised recommendations.



**rANDOMIZER & IMPLICIT PREFERENCE UPDATES**

Since the similarity matrix has some users that are chosen at random, the system will by default generate some random recommendations. If over time a user likes certain types of random recommendations, the system is also built to start taking these likes into account not only to calculate similarity but also to start updating the user’s preferences implicitly.