Project report
News Recommendation Engine

Timothée Ly, Axel Orrenius, Sam Shahriari
Group 8

Abstract

The main focus of the report is the implementation of a news recommendation engine based on Elasticsearch and web scraping. The report mentions state of the art approaches to recommendation systems, namely content-based and collaborative filtering, and the advantages of deep learning models in news recommendation systems. The report also presents the evaluation results of the search and recommendation algorithms, including precision scores for the search implementation and user experience feedback for the recommendation algorithm. The findings suggest that a search in both headline and full text is more favorable than just a headline search and that the recommendation algorithm produces articles that are mostly on topic or an extension of the topic for the user. The report also discusses the limitations and challenges of implementing a news recommendation engine.

1 Project description

In this report, we present our implementation of a news recommendation engine based on Elasticsearch and web scraping. The implementation features a query search, both implicit and explicit feedback and a recommendation system.

In a recommendation system, we consider two classes, users and items. Users have preferences for given items and the goal of the recommendation system is to extract these preferences from the user previous ratings to then recommend other items that the user is most likely to appreciate. For each user-item pair, a value represents the degree of preference of that user for that item, this value can be `None` if the user has not rated an item. All these pairs are stored in a matrix of size (number of users, number of items) called the utility matrix.

2 State of the Art / Previous Work

We distinguish 2 approaches to recommendation systems : (i) Content-based, (ii) Collaborative Filtering. Collaborative Filtering consists in recommending items that similar users rated highly. While popular, this method suffers from the cold start problem. It needs enough users to find a match, and the matrix of user / ratings is very sparse so it's hard to find users that rated the same item. To determine the best features of the items / users, one can use a latent factor based method. It is a matrix decomposition of the utility matrix in two factors to make appear hidden latent factors, useful to explain the ratings given by users. This method yields very good results when paired up with collaborative filtering [1][2]. However the same issue applies here, one needs an available dataset of user-ratings which we did not have. Note also that the rating of a series between 0 and 5 is hardly applicable to news articles. At best the user can like or dislike an article. For this reason, we chose to implement a content-based recommendation system. Basically, a content-based recommendation system tries to recommend items to a user similar to previous items that he/she positively rated.

Other than general algorithmic solutions mentioned above there has been a large upswing in Deep learning models for News Recommendation Systems in the last years. These models include Multi-layer perceptrons, Autoencoders, Convolutional neural network, Recurrent neural networks and Transformers. One of the strengths of these Deep learning models is the ability to handle huge amounts of data, even multimodal data. They also prove to be good at learning the different interactions between users and items in the recommender system. A final advantage is that they handle sequential modeling tasks better than traditional models, being able to track changing user behaviour over time, which is noted to be a major problem in news recommendation systems[2].

Since Deep learning is out of the scope of the assignment, our implementation will focus on an algorithmic approach.

3 Method

3.1 Web Scraper

The data used for the implementation stems from two different sources. The base data was collected from the 'News Category Dataset'[3], which contains 210 000 news articles. From this dataset 20 000 news articles were collected , the data collected per article is 'link', 'headline', 'text', 'tags', 'summary', 'authors' and 'date'. The larger dataset was complemented with more recent data from a web scraper. To scrape the web for news articles a Python-package, newscatcher[4], was used. A programme utilizing newscatcher was built to get the latest news feeds from NBC-news, Vox and CNN and append these to the News Category Dataset. The package helped fetch basic data about the article, such as link and title. The link is then used to scrape the article text from the web page.

3.2 Recommendation Service

We settled for an hybrid implementation between item-item and user-item recommendation systems. This implementation relies on the possibility for the user to provide active feedback. An important part of content-based recommendation system is the features we choose to represent the user profile or the items profile. As mentioned previously, we could not implement a matrix factorization method (SVD or CUR) for our recommendation system since we lacked user ratings. In that case, we could have used these latent representations to make up for document features. We therefore proceeded the following way. Each user profile stored in elastic search has two relevant fields : **history** and **preferences**. The **preferences** are an array of floats (can be positive or negative) which indicates the interest of the user for different topics. These topics were obtained as the tags of the documents in the dataset and we considered them to be sufficiently vast to make up for decent features.
Previous work [5] showed that the single action of clicking on an article / a link is already a form of interest and can be taken into account to build the user's preferences. However, it does not tackle the issue of negative feedback. We decided to use the following rules :

- If the user just read the article, add it to **history** and slightly increase corresponding **preferences**
- If the user liked the article, add it to **history** and increase corresponding **preferences**
- If the user disliked the article, decrease corresponding **preferences**

The news recommendation function takes into account the articles previously read by the user (note that disliked articles are not considered as read) and finds similar articles using the `more_like_this` function from elasticsearch. Of course, as a news recommendation engine, the older articles are filtered out of the recommendation (not from the boolean search query). Due to the technical difficulties to have a lot of articles up to date, we set articles as too old when they were published more than two years ago. To take into account the user's negative feedback, the negative preferences are filtered out from the recommendation. It means that all articles whose tag is associated with a negative value won't be included in the recommendation. One could argue that the user should be able to change his mind. The search feature enables the user to explore new topics and if he likes them, a negative preference can turn in a positive one and then appear in the recommendations.

The `more_like_this` function from elasticsearch relies on the tf-idf [6]. It takes in input a set of documents (in our case the user's reading history which is weighted so newly read articles gets a higher boost) and finds the terms of highest tf-idf. This step is weighted by the place of the document in the read history. A decay applies to read documents to encourage the engine to adapt when the users changes his preferences. A query is formed with these terms and used with the Lucene formula find the documents more likely to match the user's previous input.

The mathematical formula is as follows :

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \in q} \text{tf}(t,d) \cdot \text{idf}(t)^2 \cdot \text{boost}(t) \cdot \text{norm}(t,d)$$

where (the terms not mentionned are not very important) :

- $\text{tf}(t,d)$ is the term frequency of term t in document d
- $\text{idf}(t)$ is the inverse document frequency of term t
- $\text{coord}(q,d)$ is a factor boosting documents containing several terms of the query

A common issue with news recommendation systems is the cold start problem[2]. The most straightforward way to solve it is to compel the user to deliver some information about his preferences. Whenever a new user signs in to the recommender system, he has to choose at least 3 positive tags (negative feedback is optional). We fixed a threshold value of 3.0 for preference value initialization. Each topic category has an associated default article that was handpicked to make sure it would be as representative as possible of the category (title, keywords). When a user requests recommendations, their preferences are scanned and all preferences that exceed the threshold have their default article temporarily added to the list of read articles. The articles are not directly added to the user history so that they can change their mind and eventually unlike the category.

3.3 User Query Search

For query search, the user provides one or more search terms. A spelling analysis is first done, see below. A boolean search query is then created that looks at both the articles' headline and full text.

To make the search more personalized the user's topic preferences is taken into consideration. This is done by adding popular topics (a score higher than 2) to the search query, but instead of the must-tag the categories gets the should-tag. If an article then contains a should category its score is boosted in comparison if the same article does not include the a should category. In contrast to the recommendation service, an article with a disliked category will still be shown.

3.3.1 Spelling correction

A spelling corrector for the user input in the search query was implemented. The spelling corrector is built to both handle isolated-term as well as context-sensitive correction.

Isolated-term The isolated-term correction was implemented by first building a bigram index from all the words present in the database. The tokens from the text were cleaned so that no tokens containing numbers or that were not part of the English dictionary were processed. The package PyEnchant was used to check if a token is in the English dictionary[7]. The bigram index was written to a file for faster computation when running the program. If the search query imputed contains one word and that word is not in the database, a recommendation is generated. It is generated by finding all the bigrams of the query word, collecting all words from the index containing at least one of the bigrams, the Jaccard score for each of these words is calculated and only words with a score of more than 0.4 are kept. Lastly the edit distance between the query words and the suggested words are calculated with nltk, if the edit distance is 2 or less it is kept as a suggestion[8]. The word out of these suggestions with the highest frequency in the dataset is automatically searched and presented to the user[9].

Context-sensitive For the context-sensitive suggestions a bi-word probability table was created. To achieve this a bi-word and uni-word count was created, counting the number of appearances of single words and combination of two words. This leaves probabilities for which words follow which, adding context to each word. This probability table is also written to file for faster processing. If the user search query is more than one word long and contains at least one word that is not present in the dataset the program will retrieve a context-sensitive suggestion. For each misspelled word in the query the isolated-term suggestions are retrieved (all suggestions that pass the jaccard and edit distance test). Each of these words are then substituted for the misspelled word in the query, creating a set of combinations. From these combinations the bi-word probabilities are fetched and multiplied. The combination with the highest probability will be returned as the suggestion and used for the search. If the scores evaluate to 0 (the index has not seen this combination of words in this order), it will pick the combination with the highest sum of scores[10].

3.4 Evaluation

Evaluating the quality of the recommendations given by the system can be difficult since it can be seen as subjective. Therefore we performed two different types of evaluation, one focusing on the output of the system (accuracy) and one focusing on the subjective measures of users (user experience)[2].

Accuracy To evaluate the accuracy of the query search and recommendation service, 4 user profiles were created with different information needs (preferences and reading histories (figure 1)). Each article the search engine retrieved were either marked as relevant or non relevant. For recommendation service the top 10 articles were graded and for search the top 30 articles were graded. In the latter a separate query search was also preformed with just headline search (instead of headline and full text search) in order to evaluate if the correct fields were searched through Elasticsearch. To quantify the results two parameters were calculated: precision (P) and recall (R). In recall the total number of relevant documents needs to be known, but since we were not able to go through all of the articles in the dataset an assumption were made that every query had 100 relevant documents.

To evaluate the recommendations the same user profiles where used. 10 recommendations where generated. The articles were read and classified as a true positive or true negative based on its tag and contents in relation to the predefined user profile. A precision score was calculated from these. The 4 profiles were chosen as such because each of them highlights a feature of our implementation : cold-start solution, dislike feature (politics), focus feature (sport) , new preference (space hence science tag that was not liked at first).

1. **Cold start-profile** Likes: Politics, Food & Drink, Parents. Dislikes: Comedy, Tech.
2. **Changed mind-profile** Likes: Politics, Food & Drink, Parents. Dislikes: Comedy, Tech. Has marked ten articles about politics non interesting.
3. **Sports fan-profile** Likes: Sports, Healthy living, Food & Drink. Marks sport articles interesting.
4. **Space fan-profile** Likes: US News, World news, Weird news. Only reads articles about space.

Figure 1: Different user profiles evaluated

User experience There are a few small-scale frameworks for evaluation of user experience, we chose to take inspiration from a simple framework that allows the user to rank the recommended articles by *appropriate*, *like*, *surprising.* The evaluation was conducted on 5 users, each user performed the cold start opening and was asked to like three articles from the search. Then recommendations for the user was generated, the top 5 articles were read and ranked. The ranking was explained to each user as:

- *Like*: articles similar to the ones liked by the user
- *Appropriate*: articles that is an extensions of the topic of the liked articles
- *Surprising*: articles on another topic that the user found interesting

4 Results

Accuracy The results from the accuracy evaluation of query search is shown in table 1.

Table 1: Results from query search evaluation

		Headline + full text						Headline					
User	**Query**	**P@10**	**P@20**	**P@30**	**R@10**	**R@20**	**R@30**	**P@10**	**P@20**	**P@30**	**R@10**	**R@20**	**R@30**
1	food regulation	0.40	0.35	0.53	0.04	0.07	0.16	0.30	0.40	0.50	0.03	0.08	0.15
2	family recipes	0.60	0.50	0.63	0.06	0.10	0.19	0.60	0.45	0.47	0.06	0.09	0.14
3	win	1.00	0.80	0.63	0.1	0.16	0.19	1.00	1.00	0.73	0.1	0.2	0.22
4	space launch	0.60	0.45	0.30	0.06	0.09	0.09	0.40	0.35	0.233	0.04	0.07	0.07

The results from the recommendation relevance can be found in 2.

Table 2: Recommendation relevance

User	Precision	Tags & Cardinal
1	0.9	Parenting : 4, Politics : 2, US news : 1, World news : 1, Home & Living : 1, Wellness : 1
2	0.8	Parenting : 3, Food & Drink : 2, Home & Living : 2, World news : 1, Wellness : 1, Media : 1
3	0.7	Sports : 8, US news : 2
4	0.7	World news : 3, US news : 3 Science :2, Comedy : 2

User experience The result from the user experience evaluation is shown in table 3. Where the numbers correspond to the amount of articles that received that ranking.

Table 3: User experience evaluation

User	Main Interest	Like	Appropriate	Surprising
1	Food	2	3	0
2	Stock market	2	1	2
3	Sport	3	0	2
4	Travel	3	2	0
5	Sport	2	3	0

5 Discussion and Conclusion

Evaluation method Evaluation of news recommendation systems is painstaking since there is never a 'correct' answer, the user decides what is relevant or not. These user preferences can fluctuate over time and are difficult to manage. Users want news that they like to read but also news they did not expect, meaning that too much personalization might make the recommendations worse[2].

Because of the nature of news we chose to both evaluate the accuracy and the user experience. The accuracy focuses on how our system performs in recommending articles to a few different user-profiles. This evaluation gives us an idea of how efficient the system is in finding relevant articles for a static user.

In contrast to the accuracy evaluation we evaluated the user experience, in this case how they perceive the articles recommended to them. This is an approach to try to evaluate the dynamic nature of both users and news. Since we chose a rather simple model for this evaluation we can see the broad strokes of the user experience, but we can not notice more subtle patterns. Therefore we put more weight into our accuracy evaluation, both since it is more extensive and since it is more in-line with the course material.

Improvements that could be done on the evaluation is increasing the amount of participants in the user experience evaluation, as well as applying a more in-depth framework for the user experience. A possible improvement for the accuracy evaluation is having more people decide which articles are relevant for a profile and not, currently we (the authors) decided how relevant each article where. Taking an average over a larger group of people might yield more 'fair' results since, as we established, the nature of news is dynamic. Another way to improve accuracy is to take into consideration more information needs. The course book [9] proposes around 50 which is much greater than the 4 that we created.

Result From the accuracy evaluation results we can conclude that a search in both headline and full text is in more cases favorable than just headline search which is the reason we kept this alternative. The precision scores that are around 0.5 also shows that our search implementation is significant compared to the assignments previously in the course which had a score of around 0.3. We will not make any conclusions based on the recall score as it is based on assumptions of how many relevant articles there are and most likely to differ between different search query.

The results for the recommendations relevance only computes the precision measure - a good news recommender should provide only true positives, but it does not need to give them all. The results are shown in table 2. For the sports fan-profile, we eventually had to dislike a few (less than 5) articles with the 'politics' tag to get the result we obtained. This is due to the imbalance in the dataset as most articles are politics related. Note that without the dislikes, the precision was 0.6. For the last user profile, the issue is that the 2 year-window for old articles is too narrow because all the space related articles in the dataset are from more than 2 years ago. We extended that window to obtain our results.

The results from the user experience evaluation is quite showing. Out of the 25 articles recommended 12 where *like*, 9 *appropriate* and 4 *surprising*. Meaning that almost half of the articles where on topic from what the user previously read and 84 % of the articles where on topic or an extension of the topic (*like* and *appropriate*). The rank with the least articles is *surprising*, showing that our recommendations are most often what the user expects. This might both be positive and negative. Positive since the user gets to read articles the user showed explicit interest in, negative since the user might miss out on the broader scope of the news. Since the evaluation only regards the top 5 articles it is possible that there are more varied articles further down in the recommendations, for which a more extensive evaluation would be needed.

Limitations As previously mentioned, when the user creates his profile, an article is arbitrarily chosen as a good representative of the category. We made our best to pick what we reckon to be a credible representative but we cannot eliminate the bias induced by some words found in the article (*e.g.* 'covid') so the recommendation is somehow altered. Now, this effect is only notable when the user did not read any article since our recommendation function decays the weights of articles in the tail of the reader's history. In our implementation, these artificial articles are placed at the bottom of the pile. Once the user has read a few articles, their influence is much lesser.

Another limitation is the web scraper for finding the articles. It is quite slow at scraping news and cleaning the text from the different web pages, it is also only scraping news from three different sources. To get a better experience with the news recommendations we would need to implement another way to get news from more sources into the database, to get real time data.

Lastly, user query search just uses a binary approach when adding tags to the should list of the query. This means that a tag with a score of 3 and a tag with a score of 30 will give the same boost for documents. A better approach might be to take the actual score into consideration and boost it on a logarithmic scale.

Conclusion This project was a good opportunity to implement the algorithms studied in class for a more demanding task. The search query model yields similar results to the one implemented for the assignments. The recommendation algorithm was trickier to fine tune in order to get decent results as many parameters need to be taken into account - the user's history, what he likes, what he dislikes, how important are articles' tag compared to the actual text. While also factoring in the user experience and the nature of news recommendations it becomes difficult to evaluate what a good result is. We can therefore conclude that our implementation is efficient at recommending similar articles to what the user previously read. But a recommendation service is complex and there is room for further development and research.

6 How to run

All the instructions are included in the `Readme.md` in the **ir** directory of the repository. The repository is public - Make sure to stay on branch 7.14.

References

[1] Andreas Töscher and Michael Jahrer. “The BigChaos Solution to the Netflix Grand Prize”. In: (Jan. 2009).

[2] Shaina Raza and Chen Ding. “News recommender system: a review of recent progress, challenges, and opportunities”. In: *Artificial Intelligence Review* 55 (2021), pp. 749–800. URL: https://link.springer.com/article/10.1007/s10462-021-10043-x#Tab2.

[3] Rishabh Misra. *News Category Dataset*. 2018. URL: https://www.kaggle.com/rmisra/news-category-dataset.

[4] Artem Bugara. *Newscatcher*. 2021. URL: https://github.com/kotartemiy/newscatcher.

[5] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. “Personalized News Recommendation Based on Click Behavior”. In: *Proceedings of the 15th International Conference on Intelligent User Interfaces*. IUI ’10. Hong Kong, China: Association for Computing Machinery, 2010, pp. 31–40. ISBN: 9781605585154. DOI: 10.1145/1719970.1719976. URL: https://doi.org/10.1145/1719970.1719976.

[6] June 2014. URL: https://lucene.apache.org/core/4_9_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html.

[7] PyEnchant contributors. *PyEnchant Tutorial*. https://pyenchant.github.io/pyenchant/tutorial.html. Accessed: 2023-05-05. 2021.

[8] nltk contributors. *Natural Language Toolkit (NLTK)*. https://www.nltk.org/. Accessed: 2023-05-05. 2001.

[9] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[10] Daniel Jurafsky and James H Martin. *Speech and Language Processing*. 3rd ed. Stamford, CT: J. Ross Publishing, 2020. URL: https://web.stanford.edu/~jurafsky/slp3/3.pdf.