News Recommendation

Aditya Bhansali Albin Byström Felix Broberg Run Yan Tan

May 2019

Abstract

The goal of this report is to present a solution to a news recommendation service. Our proposed news recommendation solution is a hybrid approach utilizing content-based filtering and a popularity-based algorithm. Additionally, we trained two classifier models to enhance the results of our content-based filtering. Next, we evaluated our classifier models and our scoring function, and discuss their performance. Finally, we discuss potential future work that our project relates to.

1 Introduction

The primary goal of a news recommendation service is to suggest content for its users to consume. A good recommendation engine will take into account the user's preferences and past behavior, and combine this with other metrics of popularity, authority and relevance, to suggest content that the user would be interested in. This content can take the form of news reports, opinion columns, or even blogs, and can come in a variety of media formats, from text to podcasts to videos.

Our news recommendation engine focuses primarily on textual content, taken from small selection of reputable sources. When recommending content to users, we take a threefold approach which combines temporal relevance, content-based filtering, and user preferences.

2 Related Work

Recommendation systems can, in general, be categorized into three main classes: collaborative filtering, content-based filtering and popularity-based algorithms [1][2]. All of these are relevant to the topic of news recommendation.

Collaborative filtering is a widely popular recommendation system alternative which recommends news based on what similar users have liked before [3][4]. Because of the nature of this approach, a large amount of data on users and their interests is necessary. The approach also suffers from a "cold start" problem, where new users can not be compared to other users reliably [5].

Content-based filtering bases its recommendations on the content of the user's previous interests. It has the advantage of not needing a large amount of data of previous users and avoids the problem of "cold starts". As this type of recommendation system bases its decisions on the content alone, it suffers from not following trends and similar aspects [1]. One particularly interesting way of performing content comparisons is through comparing low-dimensional content vectors, created through methods such as Latent Dirichlet Allocation (LDA) or other classifiers, using cosine similarity and has been shown to produce good results [6].

Popularity-based algorithms base their recommendations on the current popularity of a certain item. This can include both temporal aspects as well as metadata such as clicks. This approach benefits from the fact that news articles are time-sensitive and often only relevant to a user within the first few days of its release. This approach suffers from the lack of context and user interests. [2]

Hybrid approaches have also been used which combines several classes of recommendation systems [2][3]. The idea is to try to mix different systems to counter their respective downsides and create a final system with most of the benefits and as few of the downsides as possible.

3 Method

In this section implementation details are presented. The implementation was divided into 3 parts. We present web crawling in section 3.1, news classification in section 3.2 and news suggestion in section 3.3.

3.1 Web Crawling and Infrastructure

Scrapy is a python web crawler framework [7]. It supports creation of independent crawlers that can push items into a common pipeline for processing. In this project Scrapy was used for crawling news sites and indexing articles into the recommendation engine. It had 4 main functions, find news articles, duplication checks, two-label classification and indexing. To find news articles, a crawler was created for each news site. The reason for one crawler per news site was that each website differ in structure. For each found article the following information was extracted, title, main text, published date and source. These values are stored in an item and passed to the pipeline.

In the pipeline every article goes through 3 stages. The first stage is duplication check. The goal of this stage is to remove articles already indexed by the recommendation engine. The source url was used as identification for a article. The second stage is the two-label classification for a article. The goal for this stage is to predict a "content" label and a "geography" label for every article. The last stage is to index the article into the recommendation engine.

3.2 Naïve Bayes Classifiers

We implemented the Naïve Bayes classifier, using Sci-Kit Multinomial Naïve Bayes, to classify the news articles we scraped everyday into our elastic search engine. In particular, we created 2 Naïve Bayes models, one to classify the "content" and the other to classify the "geography" of each article. Our implementation requires 2 assumptions – class-conditional independence between words, and also, independence between the 2 labels. The Naïve Bayes word probabilities are computed using the TF-IDF values of the words, which are in turn extracted from the Sci-Kit text vectorizer.

We realised that in all the news websites that we scraped from, their articles may not always be explicitly tagged with the 2 labels that we seek to capture. We call these articles unlabelled (no labels available), partially labelled (1 label available) or labelled (2 labels available). Therefore, to realize our scoring function that takes into account the "content" relevance and "geography" relevance, we employ a novel method to fully label all the articles.

There are 3 noteworthy characteristics of our method:

1. We leverage both the labelled and partially labelled articles from a diversified pool of news websites as training data to subsequently label the unlabelled and partially labelled articles.

2. A diversified pool of news sources allows a more generalised training of our model (with an "averaging" effect from all training data), and reduces the risk of bias towards any single news website. The trained model would therefore be more reliable and representative of real world news

3. Pooling labels from various news websites to label the unlabelled and partially labelled articles is highly efficient for a production search engine. Additionally, it saves costly manpower otherwise required to perform the labelling.

Our Datasets

Dataset	**Source**	**Features (Words)**	**Labels**
20 News Groups	Scikit-Learn	Available	Fully labelled
Reuters News	www.reuters.com	Available	Unlabelled and partially labelled
China Daily News	www.chinadaily.com.cn	Available	Unlabelled and partially labelled

Table 1: Datasets for training the classifiers

We also scraped BBC news for user queries, but it was not used in the model training as the volume of BBC articles were too low (1% of the news sources indicated above).

Our Features

The following 4 points describe the main characteristics of our features:

1. TF-IDF word features
2. Laplace factor of 0.01 to achieve 2 functionalities:
   (a) Smoothing of class-conditional word probabilities
   (b) Ascribing class-conditional probabilities to out-of-vocabulary words
3. Removal of stop words and changing diacritics to base alphabets
4. Sparse matrix feature representation for efficiency

Our Labels

There were 20 unique labels from "20 News Groups", 36 unique labels from "Reuters News" and 8 unique labels from "China Daily News". These labels could be a "content" label, a "geography" label, or a label of another type specific to the news source which then requires us to map to our own labels (please see our labels below). As described earlier, we endeavour to produce the two-labels of all articles by pooling and learning from the one-labels. Our two-labels (content label and geography label) are:

#	**"Content" label**	#	**"Geography" label**
1	Autos	1	Asia
2	Business and Finance	2	Europe
3	Entertainment and Lifestyle	3	India
4	Religion	4	Japan
5	Politics	5	Russia
6	Science	6	US
7	Sports	7	World
8	Technology	8	Other_geography
9	Other_content		

Table 2: Content and Geography Labels

For example, the news article "Trump slams Democrats" should be classified with the two-labels "Content: Politics" and "Geography: US".

We then trained the two Naïve-Bayes models and deployed them into production (Google Cloud). After deployment, on an hourly basis, we scraped articles from the news sources Reuters and China Daily, pipelined the articles through our model to predict the two-labels for every article, and then insert these labelled articles into our elastic search engine.

We believe that for each label, user preferences are composed of a distribution across

the classes, rather than one discrete class [8]. Therefore, for every article, we use the two models to predict a list of probabilities for the classes in "content" and another list of probabilities for the classes in "geography". Then, as an important input to the overall ranking function, we compute the cosine similarity of our users "content" probability distribution and "geography" probability distribution with that of the articles, and combine these 2 similarities with other factors into a final score, as defined by functions 1 and 2.

We also re-trained the 2 Naïve-Bayes models after every hour, so that we take into account all the partially and fully-labelled news articles scraped at every hour into an archive folder. In a production setting, this makes our models (and search engine) more current and relevant to our users.

3.3 News Recommendation

Our system utilizes a hybrid approach to recommend news to users. As our user base is small we chose to avoid the collaborative filtering method. As such, we hope to avoid the "cold start" that it suffers from and still be able to provide relevant news recommendations in a short period of time. The hybrid system is a combination of content-based filtering and a popularity-based algorithm to provide recommendations that are both relevant in regards to content and popularity.

Each user has a unique topic (content) preference vector with the same keys as the article classes, and is our main tool in classifying content relevance. Each user also has a unique geographical preference vector with the same keys as geographical regions, and is our main tool in classifying geographical relevance. At the initialization of each user, these vectors are uniformly distributed and sum to one such that the model shows no initial bias. Subsequently, these vectors are updated according to function 4 described below.

Each article has two pieces of metadata associated with them that are used by the recommendation system: the total number of clicks and the time it was published. These are the main inputs to our popularity-based algorithm. Their use, together with the content-based filtering described in the preceding paragraph, are shown in the next sections.

We provide two modes of news recommendation: the first is based on a query from the user and the second is based on suggestions we give to the user.

User Query

Queries that are entered by the user are sent to the ElasticSearch server, then matched on both title and text fields where the results are ranked according to function 1, seen below, and sent back to the user.

$$\texttt{TF\_IDF} \cdot \texttt{time\_func} \cdot \log((\texttt{clicks} + 2) \cdot 100) \cdot (\texttt{cos(topics) + cos(geo)}/4) \quad (1)$$

Suggestion

For news suggestions, all news articles are matched and ranked according to function 2, as shown below

$$\texttt{rand(0..1)} \cdot \texttt{time\_func} \cdot \texttt{log}((\texttt{clicks} + 2) \cdot 100) \cdot (\texttt{cos(topics) + cos(geo)}/4) \quad (2)$$

In both cases, the `time_func` refers to an exponential decay function provided by ElasticSearch [9]. The `time_func` returns 0.7 if the article is 4 days old in the case of a user query, and 0.7 if the article is 2 days in the case of news suggestions. The `rand(0..1)` part of function 2 is used to artificially produce variety among the top suggestions.

If a user clicks on a document, the user's topic (content) and geographical preferences vectors are updated by:

$$\alpha * \texttt{user\_content\_preference\_vector} + (1 - \alpha) * \texttt{document\_content\_vector} \quad (3)$$

$$\alpha * \texttt{user\_geography\_preference\_vector} + (1 - \alpha) * \texttt{document\_geography\_vector} \quad (4)$$

which captures and simulates the user's ever-changing preferences. Here, α is set to 0.95.

4 Results and Evaluation

4.1 Evaluation of Results from Classifiers

Classifier	**Content**	**Geography**
*Initial train set size (70%)	128,981	8,990
*Initial test set size (30%)	55,278	3,853
Precision mean	76.7%	93.3%
Precision standard deviation	0.35%	0.59%
Recall mean	74.8%	93.7%
Recall standard deviation	0.44%	0.55%

Table 3: 30 Runs of Random Stratified Splits of Dataset into 70% Train and 30% Test

*Re-training at the end of every hour will take into account incrementally larger datasets from the hourly scraping.

Generally, "content" metrics are worse than "geography" metrics because it is less complex to identify the "geography" label with country or continent words, while it is more challenging to identify the "content" as key words may be shared between different "content" labels; such as the words "trade war" being important in both labels "Business and Finance" and "Politics".

4.2 Evaluation of Scoring Model

User studies are a common form of evaluating recommendation systems [3]. We utilize this through various experiments to evaluate individual pieces of our recommendation system as well as the end result. These user studies were performed by simulated users having different profiles and interests. One profile might be interested in sports, another in science and a third in a mixture of different topics. To both control and simplify this evaluation, we considered a smaller subset of our index articles, including only approximately 350 articles from the BBC. We present our findings below.

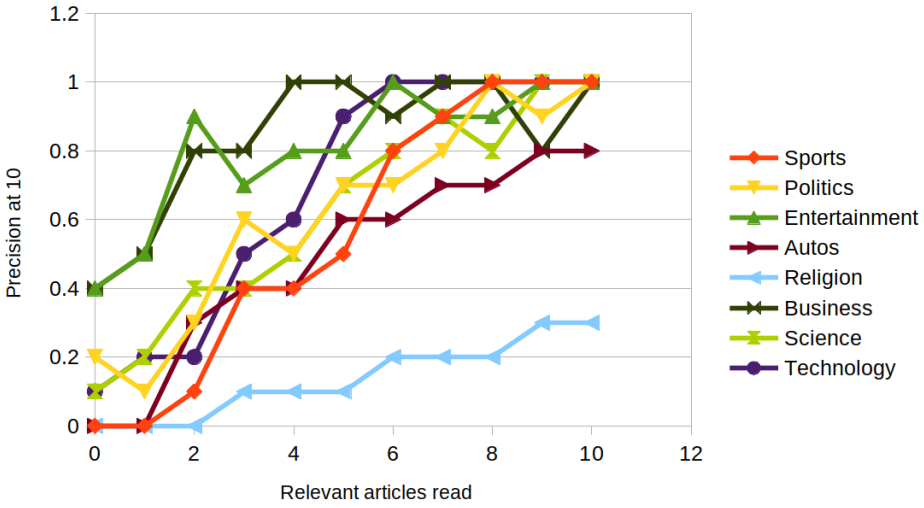

Figure 1: Shows the precision for the top 10 suggested news at varying number of relevant articles that have been read. Relevance is measured subjectively by the user in a binary form as either relevant or not. Popularity metrics were disabled during these measurements. The figure also shows that the news suggestion properly reflects the user's interest. All topics, except for religion, follow the same trend of converging towards 100% precision for the top 10 news suggestions. While the precision for religion also increases, it is at a slower pace than other interests.

To explore this further, we considered three sample users with the following topic preferences:

Category	User 1	User 2	User 3
Autos	5%	5%	2.86%
Business and Finance	5%	5%	40%
Entertainment and Lifestyle	5%	60%	2.86%
Religion	5%	5%	2.86%
Politics	5%	5%	40%
Science	5%	5%	2.86%
Sports	60%	5%	2.86%
Technology	5%	5%	2.86%
Other_content	5%	5%	2.86%

Table 4: Our sample users for evaluation

We looked at the top 30 results that were output by our news recommender engine and ranked them on a scale of 0-2, where 0 is least relevant to the user and 2 is most relevant to the user. We then calculated the normalized discounted cumulative gain for each user and plotted a precision-recall graph. These were the results:

	User 1	User 2	User 3
nDCG	0.994530307	0.977810373	0.909624472

Table 5: Normalized discounted cumulative gain for the sample users

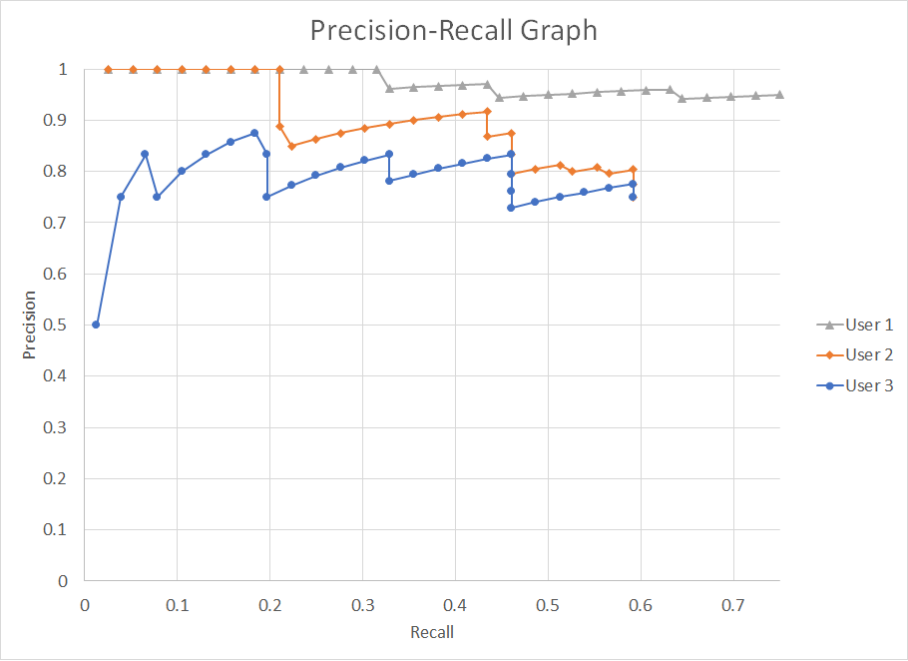

Figure 2: Shows the precision-recall curves for the first thirty articles displayed for each user. Note that the recall values were calculated assuming approximately 40 articles per category.

From these results we can see that certain categories like sports tend to fare better in our recommendation engine than categories like entertainment. It is also suggested that interest in a greater number of categories will reduce the relevance of the recommendations when compared to users who are interested in only a single category.

5 Discussion and Conclusion

Our news recommendation solution is a hybrid approach utilizing content-based filtering and a popularity-based algorithm. The content-based filtering is based on a cosine similarity between an articles content and geography vectors and the user's corresponding vectors. In the case of a user query, the content-based filtering is additionally dependent on the `TF_IDF` score of the document. Because of this, our solution is highly dependent on successful classification by the models we have created and trained. As can be seen in section 4.1, we are mostly successful in the content classification and have a high classification accuracy for the geography classification indicating that our solution has a potential to work. The results can be seen in section 4.2 where we show that our solution shows nice properties regarding appropriate convergence toward the user's interests in most cases. The topic "Religion" deviates from the norm by displaying poor convergence. This can likely be attributed to several things, but most likely is that there are few articles regarding religion that causes the classifier to struggle with understanding this particular topic.

The popularity-based part of our solution is dependent on the timestamp of the article and the number of acquired clicks, as shown in functions 1 and 2. The parameters for the timestamp and clicks would most likely need to be tweaked given a larger user base and a news text corpora with older articles. As of now, they fulfill their use of boosting articles that other users are interested in. The effect of using the clicks as a popularity metric boosted popular articles as expected given our limited number of users. The use of the click multiplier within a log function smooths the difference between no clicks and just a few clicks, such that the articles with a few clicks are reasonably not boosted too much.

5.1 Future Work

In our search engine, we have used the Naïve Bayes model on a multi-label (two-label) output. At the same time, we are aware that alternative news classification features and models could be explored. To us, some exciting directions are:

1. Explore different word features as inputs to the classifier, such as BERT, GloVe or Word2vec features instead of TF-IDF features. [10][11][12]

2. Explore the use of a single model to directly predict the multi-labels output, such as a neural network or random forest.

3. Use an unsupervised generative model, such as Latent Dirichlet Allocation, for classification into groups. [8]

References

[1] A. Lommatzsch, N. Johannes, J. Meiners, L. Helmers, and J. Domann, "Recommender ensembles for news articles based on most-popular strategies," 09 2016.

[2] N. Jonnalagedda, S. Gauch, K. Labille, and S. Alfarhood, "Incorporating popularity in a personalized news recommender system," *PeerJ Computer Science*, vol. 2, p. e63, 06 2016.

[3] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*, 2011.

[4] B. Schafer, B. J, D. Frankowski, D. , H. , J. , S. , and S. Sen, "Collaborative filtering recommender systems," 01 2007.

[5] Z.-K. Zhang, C. Liu, Y.-C. Zhang, and T. Zhou, "Solving the cold-start problem in recommender systems with social tags," *Europhysics Letters (epl)*, vol. 92, 04 2010.

[6] W. Ben Towne, C. P. Rosé, and J. D. Herbsleb, "Measuring similarity similarly: Lda and human perception," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, 09 2016.

[7] "Scrapy," 2019.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[9] ElasticSearch, "Elasticsearch decay function," 2019.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013.

KTH Github Link: https://gits-15.sys.kth.se/rytan/news-recommendation/
Access granted to user "jboye".