# WSM Project 3

Yueh-Sheng Pan 106306033, Chou-En Chien 106306035, Kuan-Wei Wu 106306070

June 2021

## 1 Abstract

In this project, we participated in a competition called MIND News Recommendation held on Codalab[1]. During the competition, we choose two main strategy to run the whole prediction, PageRank[3] algorithm and SMORe[4]. Besides these two machine learning based prediction technique, we also attempted to utilize the official deep learning model, NRMS[5], which eventually gained the highest score among all the methods we tried.

Therefore, this report will be divided into five parts. We will briefly introduce the competition in the first part, and then describe how we trained NRMS model to gain our best submission score in the second part. In the third part, we will describe how we use PageRank algorithm to make predictions in this competition, and the difficulties we faced when we tried to combine the result of PageRank prediction and NRMS model training. The forth part will be the introduction of how we use SMORe to help us generate embeddings, how we use those embeddings to make predictions and also the difficulties we faced during the calculation within this method. Finally, the fifth part will be the conclusion and findings during this competition, and the thoughts throughout the courses during this semester.

## 2 Introduction of Competition and Dataset

Online information offerings inclusive of Microsoft News have won large recognition for online information analyzing. However, due to the fact huge information articles are posted everyday, customers of online information offerings are dealing with heavy records overload. Therefore, information advice is an critical approach for personalised information offerings to enhance the reading experience and alleviate records overload.

### 2.1 Dataset Introduction

[2] During the period from October 12th to November 22nd, 2019, randomly sample data of 1,000,000 users (the selected user has to click on at least five news articles during this period), and the user ID has been coded as uID. Microsoft collects user behavior logs during this period, and this forms impression logs. An impression logs records whether the user clicks on the article displayed to the user when the user visits the homepage of the website at a specific time. The MIND data set also adds the user's click history ClickHist before impression logs.

1. Impression format: `[ImpID, uID, time, History, Candidate]`

| Column | Content |
|---|---|
| Impression ID | 91 |
| User ID | U397059 |
| Time | 11/15/2019 10:22:32 AM |
| History | N106403 N71977 N97080 N102132 N97212 N121652 |
| Impressions | N129416-0 N26703-1 N120089-1 N53018-0 N89764-0 N91737-0 N29160-0 |

2. News format: `[News ID, Category, SubCategory, Title, Abstract, URL, Title Entities, Abstract Entities]`

3. Division of the MIND data set:
   - Training set: data for the fifth week
   - Test set: data from the last week
   - Validation set: take the last day of the fifth week as the validation set

## 3 PageRank Algorithm

The PageRank algorithm we used is based on the example code provided on June 1st, and the web graph used to calculate the PageRank score is based on `TFIDF_TOP100.txt`, which contains the top 100 related news of each news in the news dataset, whose embedding is counted in TFIDF and the distance is calculated in cosine similarity. Therefore, this section will briefly introduce the example code, describe how we use these example code and files to generate PageRank score and make predictions, and how we handle exception.

### 3.1 Introduction of PageRank Example Project

The example project ranks the whole collection based on the PageRank score, which means that this project sorts the whole collection by PageRank algorithm. To calculate each document's PageRank value, this project receives input file contains the relations among documents. The input file format contains the following information: `[Doc ID] [This doc's Out-link Docs' ID list]`, and all of the documents' IDs are separated by single space. Noted that each of the docs in the latter list should also show their out-links docs' ID list in the input file, which makes this input file be like a adjacent list representing the web graph needed to calculate PageRank score and able to be converted into adjacent matrix to make further calculation.

### 3.2 Changes and Operations based on Example Code

#### 3.2.1 Filtering Out-links

In order to generate the readable input file mentioned in 3.1 , we use the file `TFIDF_TOP100.txt` and `BM25_TOP100.txt` provided by TA. We planned to extract each file's most related files as their out-link file. To make the calculation more precise, we made further inspect into these two files and decided the threshold, only the document whose score is higher than the threshold will be chosen as the out-link document. Therefore, we decided threshold for these two file by running `.describe()` in Python's Pandas package, which generates the following tables:

| | TF-IDF | BM25 |
|---|---|---|
| Count | 11483500 | 11483500 |
| Mean | 62.66089 | 13.80931 |
| Standard Deviation | 66.29406 | 7.076415 |
| Minimum Value | 0.5871458 | 0.09233093 |
| 25% | 26.60357 | 9.214133 |
| 50% | 46.62847 | 12.33702 |
| 75% | 77.36780 | 17.15937 |
| Maximun Value | 504.0482 | 95.25773 |

We decided to use Q3 as our threshold, so the threshold for TFIDF is 77, and for BM25 is 17. The following operations only used TFIDF file as first attempt.

#### 3.2.2 Narrow Down the Calculation Scope for each Impression

After filtering the out-links file, we then make these information into dictionary format as `inlinks_dict`: `{Doc ID: [out-links doc IDs list]}`, and calculate each impression's candidate news ranking predictions. As mentioned in 3.1, the PageRank algorithm requires square matrix to make calculation. Therefore, we narrow down the calculation scope for each impression, and read `inlinks_dict` with only candidate news and history news like following representation: `inlinks_dict[cand]` and `inlinks_dict[hist]`.

After reading out the out-links list, we then deleted IDs that neither belongs to candidate nor history. For example, we have a history news and its out-links ID list retrieved by `inlinks_dict[hist]` like following: `hist: [hist can can hist none none hist none ...]`, then we delete the `nones` in the retrieved list. The result of these filtering will generate a adjacent list of a square matrix, containing only the relations among history news and candidate news. We then convert these results into readable format for the example project in 3.1.

### 3.3 Exception Handling and Result

The returned result will be the overall ranks among history and candidate news for each impression, so we extract the candidate order in the overall rank as the prediction rank for the candidate news.

However, some of the impression do not have any history. In this kind of situation, we utilize the whole news' PageRank ranking result to extract these impressions' candidate ranking prediction.

After all of the operation mentioned above, our submission result for PageRank algorithm is 0.5208

## 4 SMORe Algorithm

We use two simple methods combined with embedding from Hpe model in Smore.Smore is `C++` framework for variant weighted network embedding techniques.

### 4.1 Operation

1. We transform the User and History columns of behavior.tsv in the train dataset into the format of `[user] [every news in the history] [1]` ,and save as a file.
2. Take above file as input , we use the Hpe model in Smore and get a 64-dimensions embedding of it.
3. We use two ways to calculate the score between user and every news in the impression column, which are cosine similarity and Euclidean distance.
4. Then we sort the impression in accordance with the score above.

### 4.2 Score

- cosine similarity: 0.4975
- Euclidean distance: 0.4997

### 4.3 Difficulties encountered

In the process of calculation, we found that many news in the impression column has never appeared in the history column.

Thus, we replace them with 0-closed embedding vectors, it also means that they are not be sorted basically. So it cause the problem of cold-start ,and we think this is the reason of why we will have a low accuracy with this method. This method is not content-based, it won't get the property or content of news.

News will be reported everyday and we have totally different news can be read everyday, we won't recommend old news to users. So, in this situation , we should use content-based method to get higher accuracy.

## 5 NRMS

### 5.1 Pure NRMS

NRMS is a neural news recommendation approach with multi-head self-attention. The core of NRMS is a news encoder and a user encoder. In the newsencoder, a multi-head self-attentions is used to learn news representations from news titles by modeling the interactions between words. In the user encoder, it learn representations of users from their browsed news and use multihead self-attention to capture the relatedness between the news. Besides, it apply additive attention to learn more informative news and user representations by selecting important words and news.

#### 5.1.1 Result

We train the NRMS model with different parameters, the result as below:

| | Epoch | Batch Size | Score |
|---|---|---|---|
| Attempt 1 | 1 | 100 | 0.6724 |
| Attempt 2 | 10 | 64 | 0.6815 |

#### 5.1.2 Difficulties Encountered

We expect to find out the influence of different batch size toward accuracy during the training. Because of the limitation of the equipment(The computer configuration in the lab is Nvidia 2080,RAM 16GB), when the batch size was setted too large may induce some unavailable training problem which lead to the crash of the program.

However, we discoverd that more smaller is the batch size, more better result in the training. Besides, even if setting large epoch, the result would not have significant change. In general, because NRMS is content-based, it can solve the cold-start problem efficiently and performed well in this question.

After listened to other groups proposal, we realized we can utilize ensemble and combined with different model to calculate. We hope that we can use this technique to carry on training and ranking as we meet similar question next time.

### 5.2 NRMS with PageRank and the Problems

Besides pure NRMS and pure PageRank, we also attempted to make a combination of both method, tried to use the PageRank calculated result as a new feature for NRMS to train a more precise model. In this section, we dug into the source code of NRMS project, and find out the file that we need to modify when adding new features: `newsrec/io/mind_iterator.py`. At the same time, we planned to use the prediction result generated in section 3, and might choose to use each candidate's PageRank score as feature if the operation is going well.

We first generated the train dataset's PageRank prediction for training the NRMS+PageRank model. However, due to the difference between the news dataset between valid's and train+test's, we have to generate our own `TFIDF_TOP100.txt` for valid dataset. We found this problem one day before the presentation, therefore we did not apply this method successfully.

### 5.3 NRMS with SMORe

NRMS-SMORe Embedding In this part, we try to use the embedding generated by the SMORe model as a new feature, hoping to use the learned high-dimensional expression of Embedding and put it into the NRMS model for training to increase the accuracy of the NRMS inference results. We chose HPE (Heterogeneous Preference Embedding) in SMORe for calculation. This model will generate the Embedding value of News and User based on the user's preference score provided by the input value. Each value has 64 dimensions.

| | Dev | Test |
|---|---|---|
| New User | 15.3% | 22.1% |
| New Document | 32.3% | **87.5%** |

According to the above table, it can be found that up to 87.5% of the news in the Test data set has not appeared before, and this is because the Training Set of the MIND data set collects five weeks of click data, and removes the last day as Dev Set, and the click information for the sixth week is Testing Set. Due to the high timeliness of news, most of the news in the Test data set has not appeared before, so we decided to use User's Embedding as the new feature value to ensure that the training and test values can be as consistent as possible.

Conversion steps First, convert the behavior.tsv in the Train, Dev, and Test folder that was originally downloaded into a file format that can be read by the SMORe model. The converted data includes the History and Impression click records in the file. After the conversion is completed, HPE model training is performed on the three data sets, and the Embedding values of News and User are obtained. After the conversion is completed, put the corresponding User Embedding in the new field according to the User ID, and add new fields in MIND_iterator.py and nrms.py respectively to start training.

During the training, we found that because we did not fully understand the self-attention mechanism, all the code could not be changed smoothly, resulting in an error message when the model was halfway through training.

## 6 Conclusion and Impressions

The summary of our submission result is shown as following table:

| Methods. | Score |
|---|---|
| Sort Directly | 0.5 |
| PageRank | 0.5208 |
| SMORe | 0.4965 |
| Pure NRMS with epoch 1 | 0.6724 |
| Pure NRMS with epoch 10 | 0.6815 |

We found that the deep learning based methods performs much better than the machine learning based methods. The probable reason might be that we were using the linear calculation methods to predict the non-linear problem. Besides, the population also contains some predicament like cold-start problem mentioned in 4.3, causing the previous 3 methods' score much lower than the score of NRMS'.

Yueh-Sheng Pan: In this class, I learned a lot of information retrieval knowledge, whether it is the content of the class, homework, and projects, there is a lot of learning, and these are not only the theories in the textbook, but also how to practice. For example, use the code and open source package to achieve the goal. This class taught me the importance of information retrieval and important developments in related fields. I believe these practical research methods and concepts can continue to be applied in different fields.

Chou-En Chien: After learning all the content throughout this semester, I think that it is very helpful for me to choose our future thesis topic in graduate school, and also makes me learn a lot in the field of information retrieval and natural language processing. Besides the theories, I also found that the programming technique is equivalently important, especially for tracing code, and I did practice a lot during this project. In conclusion, this course makes me want to do further research in the recommender/NLP field, and I really appreciate the teachings throughtout this semester.

Kuan-Wei Wu: In this course, I learn a lot of Information retrieval ,and what the search engine actually did after we sent a query. The most attractive part for me is that this class is not focus on theory too much, but make us have a lot of chance to practice by ourselves. I love the whole design of this course because it let us lay a solid foundation in the field of Information retrieval. I think it will help a lot when i become a graduate student in the future.

## References

[1] MIND News Recommendation Competition, `https://competitions.codalab.org/competitions/24122`, 20 June, 2020.

[2] Introduction to MIND and MIND-small datasets `https://github.com/msnews/msnews.github.io/blob/master/assets/doc/introduction.md`, 27 Jul, 2020.

[3] Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry. *The PageRank citation ranking: Bringing order to the web.*. Stanford InfoLab, 1999.

[4] SMORe: Modularize Graph Embedding for Recommendation, `https://github.com/cnclabs/smore`, 2 Nov, 2020.

[5] Microsoft Recommenders, `https://github.com/microsoft/recommenders`, 21 June, 2021.