CMPE492 - FINAL REPORT

Twitter Account Recommendation System Based on Explicit User Feedback

Supervisor: Assoc. Prof. Ali Taylan Cemgil

Kemal Berk Kocabağlı, 2013400045
kberkkocabagli@gmail.com

Güneykan Özgül, 2012400090
guneykanozgul@gmail.com

January 7, 2018

Contents

1 Introduction

With the rapid development in technology and easy access to information, a community of "makers" emerged. These people manufacture or develop products in local hubs and are open to innovation and collaboration. To facilitate an organisational device of ideas between the makers, traditional manufacturers, stakeholders and accelerators, a conceptual framework called Open Maker is being built. The project targets community builders in Europe and aims to help them create active communities.

As a sub-module of the Watchtower platform under the Open Maker project, we are developing a personalised audience recommendation system for the community builders. The standard approach is to expand the community by chain of nomination but this can be limited in many cases compared to exploiting the potential of social media. Twitter is one of the most well-known social media with a huge amount of user base, which makes it an appealing source to fetch makers for our communities. However, it is not feasible for a community builder to go through so many Twitter profiles to find members for his community. We aim to expedite this process by building a recommendation system.

2 Related Work

The most commonly used recommendation methods are collaborative and content-based filtering.

2.1 Collaborative Filtering

Collaborative filtering systems recommend items preferred by similar users. The similarity between users depends on the similarity between the items they prefer, requiring lots of user-item interactions. For example, in a shopping website, users should buy many items before an algorithm can decide that they have similar tastes.

2.1.1 Example: Movie Recommendation

In the usual scenario with N users and M movies, the recommendation problem can be solved as a matrix factorization problem, where the matrix has N rows (for users) and M columns (for items, or movies). For example, let's say we have $N = 5$, $M = 5$

Users: ['John', 'Sally', 'Peter', 'Jack', 'Rose']
Movies: ['Lord of the Rings Return of the King', 'Harry Potter Deathly Hollows Part II', 'Seven', 'A Beautiful Mind', 'Inception'].

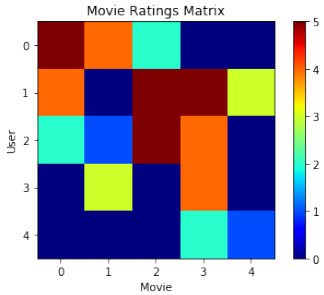

Figure 1: Matrix R

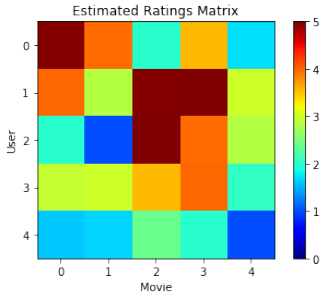

Figure 2: Matrix R'.

To predict the ratings that are zero (unrated), we factorize R into two matrices and then multiply those two to get the matrix R'.

In our example where $N = 5$, the collaborative filtering method will not work very well due to lack of data. Our example was just a demonstration of how the method can be used.

In Watchtower, for each local audience, there is only one or two Watchtower users. Therefore, $N \leq 2$ in most cases, which makes collaborative filtering impossible.

2.2 Content-based filtering

Content-based systems consider features of the items recommended. To illustrate, if an Amazon user buys a camera lens, the website recommends other camera lenses or a lens-hood, since it is related to a camera lens. This approach relies on data coming from a single user.

Watchtower users will not be able to rate sufficient amount of Twitter profiles for content-based filtering to perform very well. Nevertheless, a content-based similarity approach is our only option and given that the data is reasonable, we actually get decent results, as demonstrated below:

2.2.1 Example: Twitter Profile Recommendation based on Tweets

```
user 0: computer architecture and cpu design and embedded systems computer architecture and cpu design and embedded s
ystems computer architecture and cpu design and embedded systems
user 1: rock n roll music
user 2: deep learning and neural networks
user 3: age of technology
user 4: political science
user 5: embedded systems and internet of things
user 6: embbeded design
user 7: jazz music
user 8: convolutional neural networks
user 9: developments in technology
user 10: political debate
user 11: embedded system design
user 12: computer architecture and cpu design and embedded systems
user 13: rock n roll music
user 14: deep learning and neural networks
user 15: age of technology
user 16: political science
user 17: embedded systems and internet of things
user 18: embbeded design
user 19: jazz music
user 20: convolutional neural networks
user 21: developments in technology
user 22: political debate
user 23: political debates relating internet and technology
```

Figure 3: Users and their tweets

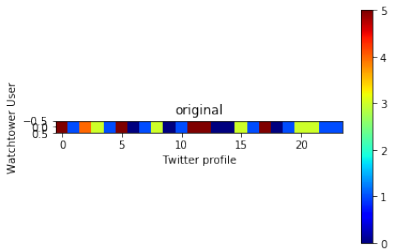

Figure 4: The ratings a Watchtower user has given to the Twitter profiles. Dark blue ones (=zero) are unrated.

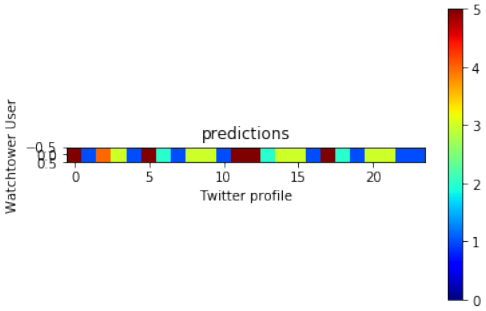

Figure 5: The predicted ratings. To predict the ratings that are zero (unrated), we first concatanate the ratings row with the TF-IDF tweet matrix and get the data matrix D. Then, we factorize D into two matrices and multiply those to get the estimated version of D: $\widetilde{D}$. The $0th$ row of $\widetilde{D}$ carries the predicted ratings.

3 Data

3.1 Collection

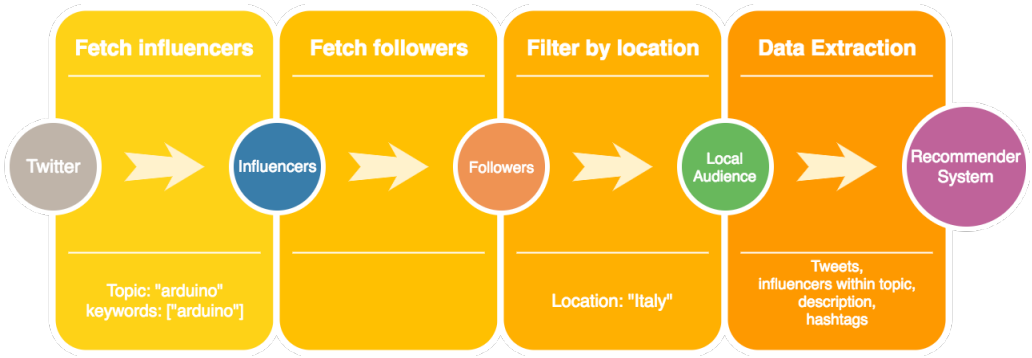

Figure 6: Data fetching pipeline

So far, we fetched **around 10 million** Twitter profiles, spanning 19 topics, which include "deep learning", "data science", "artificial intelligence", "arduino" and so on. To retrieve these profiles, we first queried the Twitter API for all keywords in each topic and fetched 20 influencers per keyword. A topic then would have a maximum influencer count of 20*(number of keywords).

After that, to find the audience for each topic, we fetched all followers of all influencers of that topic. We applied a naive location filtering, taking only profiles whose location strings contain the string corresponding to the location parameter in numerous languages such as English, Italian, Turkish etc. into audience samples.

Currently, there are 19*5 = 95 local audiences since we have 5 pre-defined locations: ['italy', 'spain', 'uk', 'slovakia', 'turkey'].

The system is deployed on the Watchtower staging server and live at http://138.68.92.181:8484. We collected ratings from Watchtower specifically for the local audience bounded by the topic: "arduino" and the location: "italy". At the moment, we have 75 rated Twitter profiles in total.

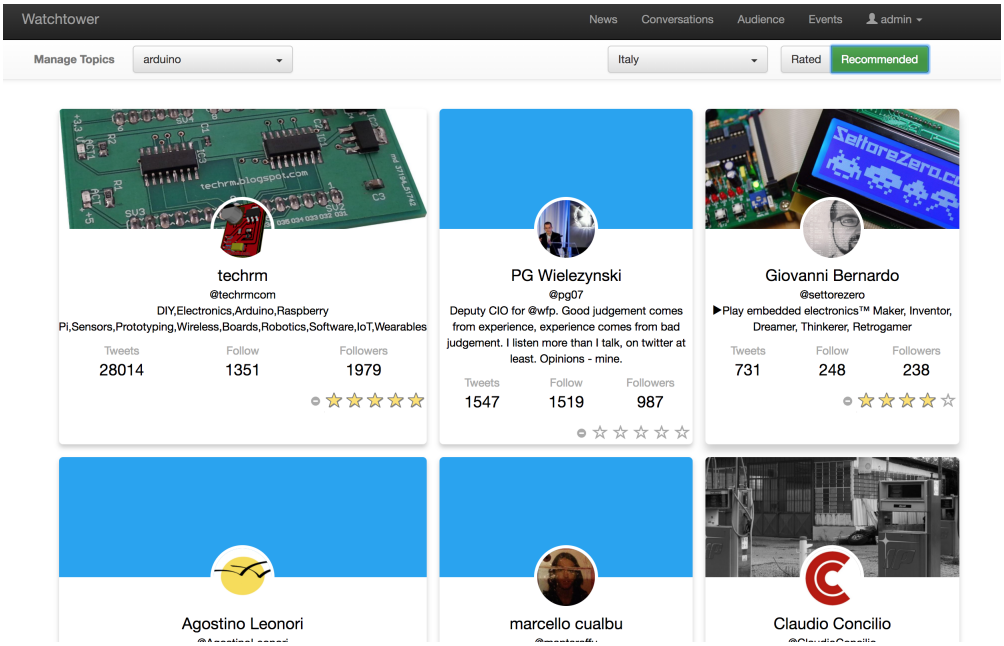

Figure 7: Recommended audience in "italy" for "arduino", with ratings

3.2 Description

Raw data from Twitter about a profile:

1. **Word corpus in last 50 tweets or retweets excluding stop words**
2. **Word corpus of hashtags in the last 50 tweets or retweets**
3. **Word corpus in profile description excluding stop words**

Extra information in our system that we will utilize:

1. Influencers the profile follows within the specific topic

3.3 Preprocessing

3.3.1 Cleaning up Raw Tweets

Tweets contain all sorts of characters, including non-alphanumeric ones and emojis. Also, because we also include retweets, all of those have "RT" in the beginning. Therefore, we used regular expressions to get rid of unnecessary parts of the text to end up with cleaner corpora (for tweets, description and hashtags).

3.3.2 Language Detection and Translation

Given that we work with audiences from countries where the native language is not English, to make the data more homogenous, we first detect the language of the tweets, hashtags and description and then use a library called "TextBlob" to translate the text to English. "TextBlob" internally uses the Google Translate API.

After preprocessing, the data for a Twitter profile looks like:

```
Topic: arduino
Location: italy
{'description': 'Chief Innovation Officer at NTT DATA Italia , proud father of '
                'two beautiful girls, a professor, a technology evangelist, an '
                'holistic thinker and a gentleman.',
 'ground_truth_rating': 1.0,
 'hashtags': 'Milano CheTempoFa Milano CheTempoFa foi13 Cefriel fvw2013 '
             'fvw2013 fvw2013 storytelling Vajont Milano CheTempoFa '
             'StartupWeekend GrandC4Picasso makerfairerome MakerFaireRome '
             'GrandC4Picasso GrandC4Picasso',
 'influencers': '266400754 84094835 767285',
 'location': 'Milan, Italy',
 'screen_name': 'funkysurfer',
 'tweets': 'at Cascina Matiot Disorders of Con Edi Touch is easier than yours '
           'via Startup weekend 3 days at Milan Decoded Milan capital of the '
           'street October 2013 Rain Massima Minima The new science is an open '
           'narration thanks to This project was born via Milan handmade with '
           'Craft Camp of a Pisa ItCup A Roncade in Veneto via October 2013 PM '
           'Showers Maximima Minima at Frijenno Magnanno Want to come to check '
           'out the offer that has been reserved for Makers vs. Todd Blatt is '
           'a mechanical engineer from Baltimore who is certainly interested '
           'in via General Meeting Classroom Osvaldo De Big the quality is '
           'more important than the Linkedin has one of the best team of via '
           'FabLab at school for the future will be talked about the future '
           'via startups Future Pills on the Startups and via To Facebook we '
           'proceed to Stanford for Living with the net wants away Today a '
           'lady approached me and she will Be the resemblance I drink only in '
           'the days beginning with The translation for CHEEKY is Sfaccimme '
           'dixit at on stage here at talking about Pirelli social life Elf '
           'Puccini Theater 7 Smart City a workplace and From Boston to via '
           'access is an Or it is the peer access is a way the Incubator '
           'Network via Festival via the first steps All our way Because '
           'artisans and makers find it hard to talk about what to do After '
           'the success away tells us and Qesse Academy - when it educates and '
           'emancipates Today 50 years of tragedy of the visit the memorial 1 '
           'card x each of the 1910 E counterclaimed The city of tomorrow in '
           'the race at Solar Decathlon Density convenience via Sapienza focus '
           'on the production of novelties and is a way October 2013 Rain '
           'Maxima Minima Thinking about my friend Marco - and what the future '
           'will be A Google with Hal Then at the research site never forget '
           'via Italian tour diary in Silicon The lucky ones have slept three '
           'way For our Coach is ready to help you to realize your Korean info '
           'to the good use of Yuhyun Park has 38 a way Social Finance and '
           'today Uman Foundation to the To get out of the crisis via At the '
           'fair of two Arduino the maker at the Liceo scientifico Filippo '
           'Lussana Brand and video 7 tips for a distribution in Italy if you '
           'want the winner of TechCrunch When a few days in the grand finale '
           'of the media and this morning the point of the We turned to see '
           'you at the next Maker Faire at Milano Malpensa Airport of Nuove - '
           'Somma 24 others Thank you for telling us about your 35 thousand '
           'people at Grazie a Still five hours before you had fun and changed '
```

```
'the two of them to thank you parked by CC one can use the The odd '
'couple I am no longer able to drive with the manual gearbox and '
'have survived the driving of'}
```

4 Methods

4.1 TF-IDF (term frequency-inverse document frequency)

TF-IDF is a widely used statistical method to weigh the importance of words in a corpus in tasks like information retrieval and user modelling. TF, or term frequency, aims to increase the weight of a word that appears a lot within a document. IDF, or inverse document frequency, aims to decrease the weight of a word that appears in many documents.

$$\begin{aligned} \texttt{TF}(t,d) &= \texttt{\# of appearances of } t \texttt{ in document } d \\ \texttt{IDF}(t) &= \log \frac{n_d}{1+\texttt{DF}(t)} \end{aligned}$$

where n_d is the total number of documents and $\texttt{DF}(t)$ is the document frequency of t

$$\texttt{TF-IDF}(t,d) = \texttt{TF}(t,d) * \texttt{IDF}(t)$$

In our case, we regard each Twitter profile as a document and observe the terms in the corpus of their tweets, description, hashtags, and influencers. Applying TF-IDF to words within the corpora will weigh words used by many Twitter profiles less and domain-specific, unique words more through IDF, while also giving more importance to frequently used words by a single Twitter profile through TF. (In the influencer case, it will weigh influencers followed by many Twitter profiles less and those followed by fewer Twitter profiles more.)

As an illustration, we cannot eliminate the word "Christmas" with the stop-word filter, but it is not related to the maker culture and needs to be discarded. On the other hand, if we are in the topic: "deep-learning", a word like "Tensorflow" is very distinguishing and important. Thanks to TF-IDF, we are able to handle those cases.

4.2 Profiling

Profiling is a method to construct a virtual vector that represents a user depending on his interactions with items. To be able to calculate similarity scores, we need to find a common vector representation for our Watchtower user and Twitter profiles.

Let a Watchtower user profile be $\mathbf{p_i}$. The profile vectors are calculated as follows:

$$\mathbf{p_i} = \sum_j r_{ij} * \mathbf{tp_j}$$

where r_{ij} is the rating that the Watchtower user i gave to the Twitter profile j
and $\mathbf{tp_j}$ is the TF-IDF vector representation of Twitter profile j.

We created a linear model when creating the user profiles and measuring the similarities since it is proved to perform well in content-based recommendation.

Note: A different Watchtower user profile is constructed from each different piece of information: influencers, tweets, description, hashtags. Therefore, there are four user profiles per Watchtower user.

4.3 Cosine Similarity

Cosine similarity measures how much a vector projects onto another.

Our logic is that an unrated Twitter profile is likely to get a high rating if it is similar to high-rated Twitter profiles. Since each Twitter profile is weighted by its rating, the high-rated Twitter profiles affect the Watchtower user profile $\mathbf{p_i}$ more than those that are rated low. Hence, in the end, when we find the cosine similarity between $\mathbf{p_i}$ and an unrated Twitter profile, it will be higher if the unrated Twitter profile is similar to the high-rated Twitter profiles.

This similarity can be found using different types of data coming from a Twitter profile. In our scenario, we consider four:

- Most recent K (tweets + retweets)
- Followed influencers within the specific topic
- Profile description
- Hashtags used in the most recent K (tweets + retweets)

Let a Watchtower user profile
generated with the (tweet + retweet) text be $\mathbf{p}_i^t$
generated with the followed influencers information be $\mathbf{p}_i^f$.
generated with the description information be $\mathbf{p}_i^d$
generated with the hashtags information be $\mathbf{p}_i^h$

Let's show the calculation of similarity scores in terms of one of the four pieces of information: $\mathbf{p}_i^t$

$$\mathbf{p_i^t} = \sum_j r_{ij} * \mathbf{tp_j^t}$$

where $tp_{jk}^t = \texttt{TF-IDF}(k, j)$ for k^{th} term in the tweets of j^{th} Twitter profile
and

$$CosSim(\mathbf{p_i^t}, \mathbf{tp_j^t}) = \frac{\mathbf{p_i^t} \cdot \mathbf{tp_j^t}}{||\mathbf{p_i^t}|| * ||\mathbf{tp_j^t}||}$$

The cosine similarity for $\mathbf{p}_i^f$, $\mathbf{p}_i^d$ and $\mathbf{p}_i^h$ is found in the same way.
For $\mathbf{p}_i^f$, $tp_{jk}^f = \texttt{TF-IDF}(k, j)$ for k^{th} influencer id that j^{th} Twitter profile follows
For $\mathbf{p}_i^d$, $tp_{jk}^d = \texttt{TF-IDF}(k, j)$ for k^{th} term in the description of the j^{th} Twitter profile
For $\mathbf{p}_i^h$, $tp_{jk}^h = \texttt{TF-IDF}(k, j)$ for k^{th} term in the hashtags of the j^{th} Twitter profile

4.4 Multiclass Logistic Regression

Multiclass Logistic Regression is a classification method used to divide the feature space into N classes by fitting a probability distribution. The conventional approach is then picking the class with the highest probability.

The cosine similarities we found will be used as input features to a multiclass logistic regressor. We consider four similarity scores to be fed into the regressor:

- Tweet Similarity Score
- Influencer Similarity Score
- Description Similarity Score
- Hashtags Similarity Score

4.5 Non-negative Matrix Factorization (NMF)

NMF is a widely used method in recommendation systems. Mathematically, let D be a NxM matrix,

$$D \approx \widetilde{\mathcal{D}}_{ij} = W \cdot H$$

$$\begin{aligned} e_{ij} &= D_{ij} - \widetilde{\mathcal{D}}_{ij} \\ &= D_{ij} - \sum_k W_{ik} H_{kj} \end{aligned}$$

$$\frac{\partial e_{ij}^2}{\partial W_{ik}} = -2 e_{ij} H_{kj}$$

$$\frac{\partial e_{ij}^2}{\partial H_{ik}} = -2 e_{ij} W_{kj}$$

Update Rules become,

$$W_{ik} \leftarrow W_{ik} + \alpha 2 e_{ij} H_{kj}$$

$$H_{ik} \leftarrow H_{ik} + \alpha 2 e_{ij} W_{kj}$$

4.5.1 Regularization

To avoid overfitting, we modify error function as follows:

$$eij^2 = (D_{ij} - \widetilde{\mathcal{D}}_{ij})^2 + \frac{\beta}{2}(||W||^2 + ||H||^2)$$

Then the derivatives become,

$$\frac{\partial e_{ij}^2}{\partial W_{ik}} = -2 e_{ij} H_{kj} + \beta W_{ik}$$

$$\frac{\partial e_{ij}^2}{\partial H_{ik}} = -2 e_{ij} W_{kj} + \beta H_{ik}$$

Finally, update rules can be written as follows:

$$W_{ik} \leftarrow W_{ik} + \alpha(2 e_{ij} H_{kj} - \beta W_{ik})$$

$$H_{ik} \leftarrow H_{ik} + \alpha(2 e_{ij} W_{kj} - \beta H_{ik})$$

5 Algorithms

5.1 SimLogReg

Algorithm 1 SimLogReg

```
1:  procedure PREDICTRATINGS(Watchtower_user_profile)
2:      for tp in tfidf_tweet_vectors do
3:          tweetSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
4:      for tp in tfidf_influencer_vectors do
5:          influencerSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
6:      for tp in tfidf_description_vectors do
7:          descriptionSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
8:      for tp in tfidf_hashtag_vectors do
9:          hashtagSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
10:     Data.concatanate(tweetSimilarities)
11:     Data.concatanate(influencerSimilarities)
12:     Data.concatanate(descriptionSimilarities)
13:     Data.concatanate(hashtagSimilarities)
14:     trainingData ← Data.get_training
15:     multiclassLogReg ← new MLR(params)
16:     multiclassLogReg.fit(trainingData, ratings)
17:     return multiclassLogReg.predict(testingData)
```

5.2 NMF + Document Clustering

Algorithm 2 NMFDocClust

```
1:  procedure PREDICTRATINGS
2:      Data.concatanate(tfidf_tweet_vectors)
3:      Data.concatanate(tfidf_influencer_vectors)
4:      Data.concatanate(tfidf_description_vectors)
5:      Data.concatanate(tfidf_hashtag_vectors)
6:      W, H ← factorizeMatrix(Data, rank=10)
7:      clusterNumbers ← argmax(H, axis=0)
8:      clusterRatings ← zeros
9:      for i in range(clusterNumbers.length) do
10:         clusterNo ← clusterNumbers[i]
11:         clusterRatings[clusterNo] ← clusterRatings[clusterNo] + ratings[i]
12:     for i in range(clusterNumbers.length) do
13:         clusterSize ← 0
14:         for j in range(clusterNumbers.length) do
15:             if j == i then
16:                 clusterSize ← clusterSize + 1
17:         if clusterSize! = 0 then
18:             clusterRatings[i] ← clusterRatings[i] / clusterSize
19:     for i in range(clusterNumbers.length) do
20:         clusterNo ← clusterNumbers[i]
21:         predictions[i] ← around(clusterRatings[clusterNo])
22:     return predictions
```

5.3 NMF with the Ratings Row

Algorithm 3 NMFwithRatings

```
1: procedure PREDICTRATINGS
2:     Data.concatanate(tfidf_tweet_vectors)
3:     Data.concatanate(tfidf_influencer_vectors)
4:     Data.concatanate(tfidf_description_vectors)
5:     Data.concatanate(tfidf_hashtag_vectors)
6:     W, H ← factorizeMatrixTaylanHoca(ratings.concatanate(Data))
7:     E ← (W · H)
8:     predictions ← E[0]
9:     return around((10 ∗ predictions)/max(predictions))
```

5.4 NMF with the Ratings Row Regularized

Algorithm 4 NMFwithRatingsRegularized

```
1: procedure PREDICTRATINGS
2:     Data.concatanate(tfidf_tweet_vectors)
3:     Data.concatanate(tfidf_influencer_vectors)
4:     Data.concatanate(tfidf_description_vectors)
5:     Data.concatanate(tfidf_hashtag_vectors)
6:     W, H ← factorizeMatrixOurs(ratings.concatanate(Data), regularized=True)
7:     E ← (W · H)
8:     predictions ← E[0]
9:     return around(predictions)
```

5.5 SimNMF Regularized

Algorithm 5 SimNMFRegularized

```
1: procedure PREDICTRATINGS(Watchtower_user_profile)
2:     for tp in tfidf_tweet_vectors do
3:         tweetSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
4:     for tp in tfidf_influencer_vectors do
5:         influencerSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
6:     for tp in tfidf_description_vectors do
7:         descriptionSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
8:     for tp in tfidf_hashtag_vectors do
9:         hashtagSimilarities.append(cosineSimilarity(Watchtower_user_profile, tp))
10:    Data.concatanate(tweetSimilarities)
11:    Data.concatanate(influencerSimilarities)
12:    Data.concatanate(descriptionSimilarities)
13:    Data.concatanate(hashtagSimilarities)
14:    W, H ← factorizeMatrixOurs(ratings.concatanate(Data), regularized=True)
15:    E ← (W · H)
16:    predictions ← E[0]
17:    return around(predictions)
```

6 Results

6.1 Evaluation

For evaluation, we only used Twitter profiles that have been rated. We knew the ground truth for all the profiles and made the rating 0 for those we wanted to include in the testing set. In the end, we calculated the mean square error, where observations were the predictions and expected results were the ground truth.

To be able to decrease the effect of luck, we ran each algorithm 20 times and looked at the mean and standard deviation of the mean square errors to assess performance.

Subsampling Count=20	**Average Training MSE $\pm$ stdev**	**Average Testing MSE $\pm$ stdev**
SimLogReg	4.96 ± 1.36	13.51 ± 5.35
NMF + Document Clustering	6.72 ± 0.70	9.92 ± 3.90
NMF with the Ratings Row	7.79 ± 1.91	8.23 ± 2.93
NMF with the Ratings Row Regularized	0.04 ± 0.01	9.29 ± 3.77
SimNMF Regularized	0.0 ± 0.0	7.56 ± 3.07

Table 1: Results for 5 different recommendation algorithms. Training set size = 40, Testing set size = 10. Rating scale = 1-10

6.2 Conclusion

Among the five different approaches, the one with the best performance was **SimNMF regularized**. The reason for that could be the fact that the data matrix was not sparse as we used similarities instead of raw data. Furthermore, the values in the matrix were not so small. We also observed that increasing the rank of factorization resulted in better performance up to a certain point and fixed the rank at 16.

Regularization is used to control the magnitude of the values in the matrix and prevents the values to be too big.

One may notice that the mean and the standard deviation of the training MSE is 0.0 and think that the algorithm overfits. Our idea is that because there is too little training data, the similarities are biased in favor of the Twitter profiles in the training set. The probability of finding a Twitter profile with a high rating in the training set similar to one in the testing set which should also get a high rating is low.

7 Challenges

Our setting was fundamentally different and constrained compared to those of traditional recommender systems in various ways:

1. We only have a single or a handful of Watchtower users that rate the Twitter profiles. Therefore, **collaborative filtering is not possible**.

2. Each Watchtower user gives a little amount of ratings per local audience, which is a huge **data constraint**.

3. The items to recommend in our setting are Twitter profiles, which have very **heterogenous data**

4. The Twitter profile data is also **dynamic** (not static like in the case of movie recommendations) - especially tweets

2, 3 and 4 **make content-based filtering very difficult.**

Additionally, there is bias in the ratings. A rating might fluctuate 0.5-1.5 out of 5, depending on the rater. The psychological aspect of different rating settings (whether the scale should be 0-1, 1-5, 1-10 etc.) could be a topic for investigation by itself.

8 Future Work

1. To improve the quality of the data, one can either predetermine a set of keywords for each topic and eliminate all other words from the tweets corpus, or have a script to decide if a tweet is relevant to the current topic and discard irrelevant tweets. This would also decrease the sparsity of the data matrix, resulting in better predictions. We created a toy example where the tweets corpus contains only topic related words and observed that especially regularized NMF performs really well for such a case.

2. The temporal dynamics of a Twitter profile is also important. After all, the tastes of a person change over time. If a Twitter profile A is rated high for "deep-learning" today, it does not mean that he should have a high rating one year from now. His interests might shift. Therefore, one could weigh the recent tweets more than the past tweets. Also, the raters should rate the profiles regularly - once per month or so.

3. One could experiment with different rating scales.

9 Acknowledgements

We thank

- Prof. Ali Taylan Cemgil for his guidance in this project.
- Arman Boyacı for his discussions and support.
- the Open Maker team for the intro video that we used in our project video.
- all other people working in the Open Maker project to make the website real.
- the people who provided the Twitter profile ratings.

References

[1] Cemgil, Ali Taylan.
MatrixFactorization. https://github.com/atcemgil/notes/blob/master/MatrixFactorization.ipynb.

[2] Koren Y., Bell R. and Volinsky C., "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009. doi: 10.1109/MC.2009.263

[3] Rajaraman, A., & Ullman, J. (2011). Recommendation Systems. In Mining of Massive Datasets (pp. 277-309). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139058452.010

[4] Shani G., Gunawardana A. (2011) Evaluating Recommendation Systems. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA

[5] Xu, Wei, Xin Liu, and Yihong Gong. "Document clustering based on non-negative matrix factorization." Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval. ACM, 2003.

[6] Yeung, Albert Au. "Matrix Factorization: A Simple Tutorial and Implementation in Python." Matrix Factorization: A Simple Tutorial and Implementation in Python, www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/.