Search Project - Yelp

Abhishek Bane
Department of Data Science
Indiana University
Bloomington, IN 47408
`abbane@iu.edu`

Rohit Rokde
Department of Data Science
Indiana University
Bloomington, IN 47408
`rrokde@iu.edu`

Vidit Mohaniya
Department of Data Science
Indiana University
Bloomington, IN 47408
`vimohan@iu.edu`

Github

https://github.iu.edu/vimohan/Search-Project

1 Introduction

The goal of this project is to understand information retrieval and natural language processing better. In particular, the project has two tasks, first to understand how recommendations are personalized and second to classify text into different levels ranging from most negative to most positive. The intent of task 1 is to provide Yelp users with personalized business recommendations using multiple collaborative filtering methods and then evaluate the performance of these methods. Task 2 aims at identifying the stars given by a Yelp user to a business based on their text reviews alone. This task involves processing the text data and exploring methods which best classify the text reviews.

The code for this report is available at IU Github.[1]

2 Team's Contribution

We divided the tasks among us and communicated over zoom. We also shared the resources which we thought will help the other person in their tasks. The task division is as follows:
Abhishek Bane

- Collaborative filtering with SVD and ALS
- Social context with Collaborative filtering
- Model Evaluation

Rohit Rokde

- Task 2: Feature Engineering
- Model Selection & Evaluation
- Compilation of results

Vidit Mohaniya

- Exploratory Data Analysis
- Data Preprocessing
- Task 1: Memory-Based Collaborative Filtering Approaches and their evaluation
- Part of Task 2: Models Development

[1]https://github.iu.edu/vimohan/Search-Project

3 EDA & Data Pre-processing

Business
The state of Arizona has the highest number of businesses in the data. Most common business categories are Restaurants and Food. These categories together have more than 75000 businesses[Figure2].

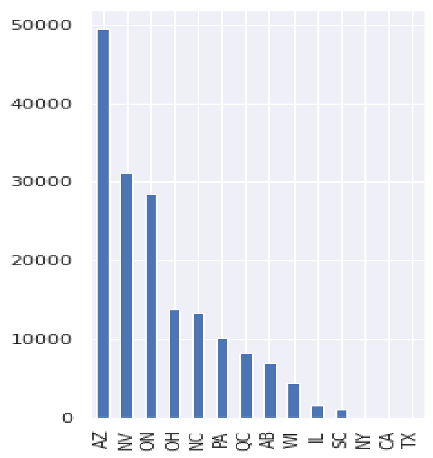

Figure 1: Businesses per State

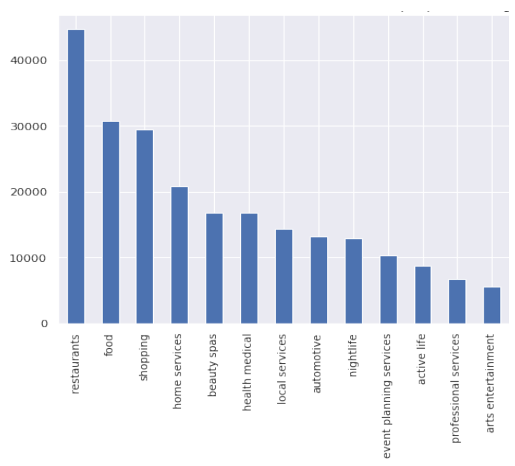

Figure 2: Businesses per Category

Users
The dataset has very few users before the year 2010 and most of the users in the dataset prefer giving a rating of 3 or more stars to the business. For this project, we would consider a rating of 3 stars and above as positive, which makes the dataset imbalanced with a majority of positive ratings.

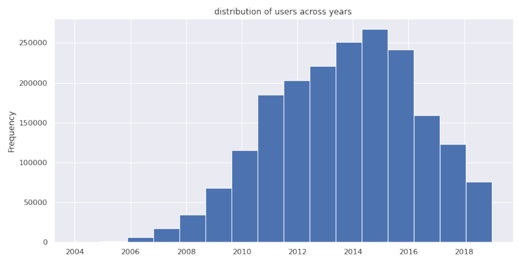

Figure 3: Distribution of Users

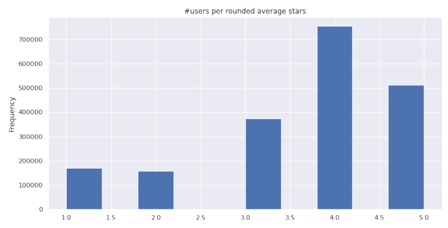

Figure 4: Users per rounded average stars

Data Pre-processing Criteria Based on the exploratory data analysis, the original dataset was filtered to include reviews and ratings from open businesses in the state of Arizona which were under the categories $Restaurants$ and $Food$. The number of users experienced a steady growth after the year 2010, hence all the users before the year 2010 have been excluded from the data as well. Collaborative filtering algorithms often experience cold start problems due to lack of ratings for a user. To avoid this issue, only the user who rated at least 50 businesses were kept in the data. If a user had multiple ratings for a business, the most recent rating was retained since that would most accurately denote a user's current sentiment towards a business.

4 Tasks

4.1 Task 1: Comparing different Collaborative Filtering methods for recommendation

In this task we have compared Memory Based Collaborative Filtering and Model Based Collaborative Filtering using various algorithms and recommend businesses to users.

4.1.1 Memory based methods:

Memory-Based CF approaches are divided into two approaches: user-item filtering and item-item filtering. In user-item filtering we take a particular user and find users that are similar to the selected user based on some sort of distance/similarity metric, and recommend items that those similar users liked.

In item-item filtering, we take an item and find users who liked the selected item and find other items that those users or similar users also liked.

For both the above approaches, we need a distance/similarity metric to measure how similar the two users (in case of user based approach also known as user-item filtering) or two items are (in case of item based approach also known as item-item filtering).
In this project we have compared three different types of similarity metric: MSD (Mean Square Deviation), Cosine and Pearson.

In order to use the above metric to compare similarity between users or items, we have used the rating given to a business by a user. These ratings are on 5 point scale ranging from 1 to 5 with an increment of 1.

User-Based
Below are the three similarity metrics which measures how much similar two users u and v are.

I_{uv} is the set of all businesses which both users u and v have rated.

$$msd\_sim(u, v) = \frac{1}{\frac{1}{|I_{uv}|} \cdot \sum\limits_{i \in I_{uv}} (r_{ui} - r_{vi})^2 + 1} \tag{1}$$

This is inverse of the the Mean Squared Difference of ratings between all pairs of users.

$$cosine\_sim(u, v) = \frac{\sum\limits_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} r_{vi}^2}} \tag{2}$$

Also known as vector-based similarity, in this two users and their ratings are viewed as vectors, and the angle between these vectors defines the similarity between them.

$$pearson\_sim(u, v) = \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - \mu_u) \cdot (r_{vi} - \mu_v)}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} (r_{vi} - \mu_v)^2}} \tag{3}$$

In this, for the set of items rated by a pair of users, we calculate how much their rating on item i deviates from their average ratings. Once we have build the user-item similarity matrix using one of the above distance/similarity metric, we can predict the ratings that were not present in the dataset. For user-based CF, we predict that a user's u rating for item i is given by the weighted sum of k highest users' ratings for item i. We then normalize these ratings.

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} sim(u, v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} sim(u, v)} \tag{4}$$

One inherent problem with the above approach of predicting ratings is that it fails to consider the fact that different users have different ways of giving ratings, i.e, it does not consider the user's bias towards an item.

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} sim(u, v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} sim(u, v)} \tag{5}$$

In this we subtract each user's average rating when summing over similar user's ratings and then add that average back in at the end.

Item-Based
Below are the three similarity metrics which measures how much similar two items i and j are.

U_{ij} is the set of all users who rated both items i and j.

$$msd\_sim(i,j) = \frac{1}{\frac{1}{|U_{ij}|} \cdot \sum\limits_{u \in U_{ij}} (r_{ui} - r_{uj})^2 + 1} \tag{6}$$

It compute the Mean Squared Difference of ratings between all pairs of items.

$$cosine\_sim(i,j) = \frac{\sum\limits_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum\limits_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum\limits_{u \in U_{ij}} r_{uj}^2}} \tag{7}$$

This is similar to cosine similarity in user based approach. Here we think of two items and their ratings as vectors, and then defines the similarity between them as an angle between the vectors

$$pearson\_sim(i,j) = \frac{\sum\limits_{u \in U_{ij}} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum\limits_{u \in U_{ij}} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum\limits_{u \in U_{ij}} (r_{uj} - \mu_j)^2}} \tag{8}$$

This is based on how much the ratings from common users for a pair of items deviates from average rating for those items.

Just like User-based approach, here also we have to ways to predict the ratings, one which does not consider the users bias and the one which does.

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} sim(i,j) \cdot r_{uj}}{\sum\limits_{j \in N_u^k(i)} sim(i,j)} \tag{9}$$

$$\hat{r}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} sim(i,j)} \tag{10}$$

4.1.2 Model based methods:

Model based collaborative filtering makes a model of existing user-business interactions and then provides business recommendations. The process of building a model is performed by a simple rule based process, a machine learning algorithm, factorizing user business matrix or by probabilistic models. In simple rule based processes, association between a user and already interacted businesses can be calculated and new businesses can be recommended based on the strength of association between businesses. The machine learning methods can include clustering similar users together who rate a business similarly and then estimating the probability of a user being part of that cluster. However, the most well known method for model based collaborative filtering is Matrix Factorization

Collaborative Filtering with Matrix Factorization: The assumption while using matrix factorization is that there are few features/factors that determine the strength of the interaction between a user and a business. In case of restaurants those might be the type of food served, price of a meal, ambience etc. These can be called the latent factors of a business or a user and we do not have data for these factors. We try to estimate these user and business latent factors using the user-business interaction matrix.
The user-business interaction matrix can be populated with either implicit or explicit interactions and is usually sparse. Explicit interactions are those where a user explicitly likes or dislikes a certain business. An example of explicit interaction is the user rating on Yelp which indicates user's feedback

for a business in the form of stars. Implicit interactions are user's actions which may indicate the user's feedback about that business. For example, a user visits a restaurant a lot but has never rated it explicitly on Yelp. User checking in a restaurant multiple times might indicate that he likes that restaurant.
The user business matrix is factorized into two smaller matrices with user and business latent factors. These new matrices with user and business latent factors are much smaller than the original user business matrix because we only retain **k** latent factors where **k** is much smaller than the number of users or businesses.

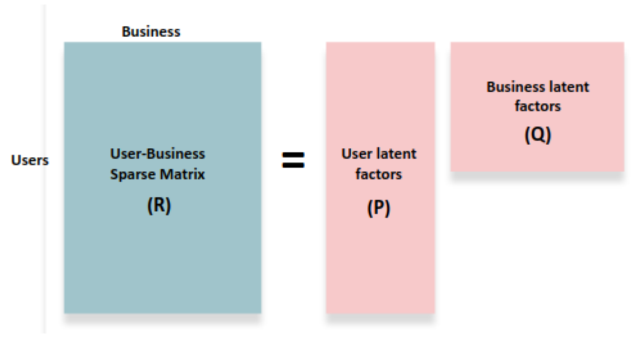

Figure 5: Matrix Factorization

We try to estimate the user and business latent factors matrices (**P** and **Q**) such that their product is as close to the original user-business matrix **R** and we use *root mean squared error* as the cost function.

$$RMSE = \sqrt{\sum (r_{ui} - \hat{r}_{ui})^2}$$

where $\hat{r}_{ui}$ are the predicted user-business interactions for a user u and a business i. RMSE is the squared difference between the actual and predicted values of user-business interactions. To find the optimal P and Q matrices, we used two different methods - Singular Value Decomposition (SVD) and Alternating Least Squares.

SVD with Stochastic Gradient Descent: While using Stochastic Gradient Descent (SGD) to factorize the user-business matrix, the predicted interaction $\hat{r_{ui}}$ is set as $\hat{r_{ui}} = b_u + b_i + p_u^T q_i$ where b_u is the bias for the user u and b_i is the bias for business i. The cost function which is minimized using stochastic gradient descent is:

$$CostFunction = \sum ((r_{ui} - \hat{r_{ui}})^2 + \lambda(\sum ||p_u||^2 + \sum ||q_i||^2))$$

SGD starts from a random point and tries to minimize the cost function based on the errors from the previously predicted interaction. The new values are calculated by adding the previous values to the errors (e_{ui}) between actual and predicted interactions. The errors are also multiplied by a learning rate so that the which is small to find the optimal parameters. The learning rate decides how fast or slow convergence would be achieved. Following equtions are used for minimization:
$b_u = b_u + \gamma(e_{ui} - \lambda b_u)$
$b_i = b_i + \gamma(e_{ui} - \lambda b_i)$
$p_u = p_u + \gamma(e_{ui}.q_i - \lambda p_u)$
$q_i = q_i + \gamma(e_{ui}.p_u - \lambda q_i)$
The hyperparameters to be tuned for SGD are number of epochs, learning rate γ and the regularization parameter λ.

Alternating Least Squares (ALS): The matrices **P** and **Q** are unknown and optimizing for both the matrices together might lead to non-convex optimization. So, the ALS algorithm fixes one of the matrices and solves for the other matrix. Since one matrix is fixed the new optimization would now be convex and quadratic which can be solved using the least squares solution.
Similar to SVD, the cost function for ALS also includes a two terms. The first term is the squared difference between actual and predicted interactions and the second term is a regularization term which prevents the estimates from taking very large values. However, the bias terms for predicted interactions are missing.

$$CostFunction = \sum((r_{ui} - p_u^T q_i)^2 + \lambda(\sum ||p_u||^2 + \sum ||q_i||^2))$$

ALS algorithm:

- Initialize user vector **P** (can be random)
- For each restaurant, compute $q_i = (P^T P + \lambda I)^{-1} P^T r_i$ where r_i is the vector of ratings for a restaurant
- For each user, compute $p_u = (Q^T Q + \lambda I)^{-1} Q^T r_u$ where r_u is the vector of ratings of a user
- Repeat for **N** iterations till convergence

The equations for q_i and p_u are the least squares solutions.

Adding Social context to ALS: Not every user who visits a restaurant rates it on the Yelp website. So the user-business explicit interactions are very limited [1]. On the other hand, implicit interactions are more commonly found between users and businesses. So, in this project we decided to include interactions between user's friends and businesses as implicit interactions. The hypothesis here is that if most of the user's friends rate a business positively ie. give the business 3 or more stars, then there is a high probability that the user might like it as well. So, the new user-business matrix would include the number positive ratings for a business from a user's friends. A higher number of positive interactions from friends would mean that the user would also rate that business positively. Then, this user-business matrix is factorized using Alternating Least Squares to get recommendations.
To implement this implicit interaction using ALS, a confidence term is added which determines how reliable the interactions are.

$$CostFunction = \sum(C_{ui}(r_{ij} - p_u^T q_i)^2 + \lambda(\sum ||p_u||^2 + \sum ||q_i||^2))$$

4.1.3 Evaluation and Results of Collaborative Filtering

The collaborative algorithms were evaluated by splitting the original dataset into training data and test data. This split of train and test data was created for every user. Some percentage of ratings of every user was kept aside as test ratings and these test ratings were removed from the training dataset. So, if a user u has rated 10 businesses and test percentage is 20, then 2 ratings of this user will be used as test data and 8 ratings as train. The metrics used for evaluating recommendations were **Mean Precision@k** and **Mean Reciprocal Rank(MRR)@k** are also commonly used in literature [2] . Mean Precision gives the average number of positively rated (more than 3 stars) restaurants in top k recommendations. MRR gives the reciprocal of position at which the first positively rated restaurant is recommended. Higher values for both the metrics are desired.

For the Memory Based Approach, it can be seen that cosine is either performing similar or tends to perform better than others in both with or without user bias adjusted prediction. If we just look at the User Biases then we can see that Bias tends to perform similar or better than the models which do not consider the user's bias. Between User-Based and Item-Based, user bias adjusted cosine similarity takes a small lead when compare to item-based cosine bias adjusted similarity. performs equally good where user-based approach tends to perform better than the Item-Based approach. One thing to observe is that while predicting the rating for a business we only sum over 40 highest ratings. We ran a grid search to find the best hyper parameter, details of which can be found in the git. However the parameters we found were: user based cosine similarity summing over 40 highest neighbors.

For the model Based Approaches, the baseline model is based on biases of users and businesses ie. how different are the users rating for a business than the average rating. SVD with Gradient Descent had the best performance among the model based methods having the highest Mean Precision and MRR. Improvement was also seen in ALS when it was used with social data (implicit interactions).

Table 1: User Based Memory CF Results

	User Based					
	Without User Bias			With Bias		
	MSD	Cosine	Pearson	MSD	Cosine	Pearson
RMSE	1.10	1.11	1.16	1.05	1.05	1.10
MAE	0.85	0.85	0.89	0.80	0.80	0.84
Mean Precision@5	0.91	0.91	0.90	0.90	0.91	0.90
MRR@5	0.95	0.95	0.94	0.95	0.96	0.94

Table 2: Item Based Memory CF Results

	Item Based					
	Without User Bias			With Bias		
	MSD	Cosine	Pearson	MSD	Cosine	Pearson
RMSE	1.07	1.08	1.14	1.05	1.11	1.11
MAE	0.83	0.84	0.87	0.81	0.80	0.84
Mean Precision@5	0.86	0.85	0.86	0.91	0.91	0.90
MRR@5	0.91	0.91	0.91	0.95	0.95	0.95

Table 3: Model Based CF Results

	Baseline	SVD with Gradient Descent	ALS	ALS with social context
Mean Precision@5	0.91	0.91	0.87	0.88
MRR@5	0.96	0.97	0.94	0.95
RMSE	1.11	1.02	1.03	-

4.2 Task 2: Can we predict the review's ratings of a user given to a business from the review text alone?

One of the features of Yelp platform is the ability of a user to post a review about a business on the website. These reviews consists of three components viz. review rating, review text and photos. For our purposes, we are using a data set with only review rating, and review text as a part of those reviews. Review ratings can take up 5 different values viz. 1 star, 2 stars, 3 stars, 4 stars, and 5 stars, and a user can click an image with as many number of stars as he/she wants. Internally these values are stored as integers for the system to store and interpret this data. The review text has a maximum character limit of 5000 which roughly translates to 2 pages; which allows sufficient freedom to an average user to write a review.

Here the question we ask is, can we figure out the value of a review rating by looking at its text alone. The premise behind this question is that, since reviews with lower ratings should correspond to review text that are statistically significant to be differentiated among themselves. So, a lower rating should correspond to a review text that contains features such as choice of words that are overly criticizing in tone, has an overall negative sentiment, etc. A higher rating should correspond to a review text that contains features such as choice of words that praise the business, has an overall positive sentiment and so on.

4.2.1 Method

The total number of reviews in this data set is humongous (fig. 6c), necessitating a cut down in the number of data-samples. To do that we have selected only those businesses which are reported as Restaurants and are located in the State of Arizona. Further, we have down sampled those classes from the data set whose counts exceed the minimum count in one of those classes (Fig. 7). The class with 2 stars has the lowest number of samples, so we made sure that all other classes had the same number of samples.

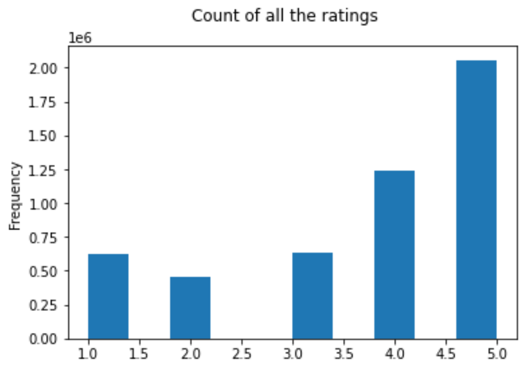

Figure 6: Distribution of ratings

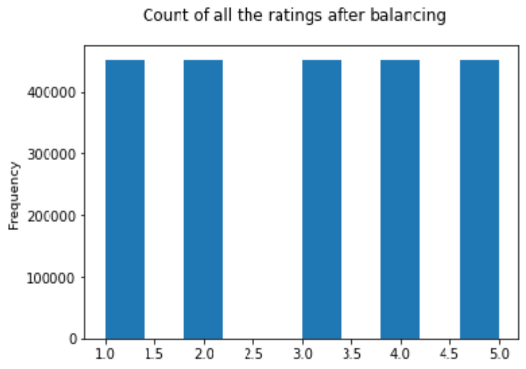

Figure 7: Dist. of ratings after downsampling

4.2.2 Feature engineering

In our experiment, we need to predict the ratings of a review from its original text features. Since Machine Learning models need numerical features to work on, we had to come up with procedures to convert these text features into numerical features. We used two methods to generate numerical features from the text data.

- TF-IDF Vectorizer
- Word Embeddings

The TF-IDF Vectorizer from the python package sklearn allows us to calculate the Term-Frequency Inverse Document Frequency values of the each of the documents. The package essentially converts the raw documents to a matrix of IF-IDF features. In our case, a document is one sample of review text. Using the TF-IDF features, we were able to achieve an accuracy of 58% using Linear Logistic Regression model to predict the review ratings. We were also able to achieve an accuracy of 65% using Long Short Term Memory (LSTM) network using word embeddings. For the purposes of this experiment, we used the "Fast Text" for Word Representation which is a kind of word embeddings provided by Facebook Open Source. The specific word embedding we have used here is the "wiki-news-300d-1M" embedding which is a 300 dimensional embedding trained on 16 Billion words consisting 1 Million words in its vocabulary.

4.2.3 Results

Table 4: Accuracies obtained using Traditional Machine Learning approaches

	Multinomial Naive Bayes	**Logistic Regression**	**XGBoost**
Token counts (CountVectorizer)	54.47%	60.88%	59.67%
TF-IDF (Word level)	54.84%	62.30%	59.79%
TF-IDF (N-gram)	55.49%	60.48%	-
TF-IDF (Character level)	53.54%	60.80%	-

So it turns out that rating prediction for the rating classes 1 and 5, that is, the predictions for the most negative class and the most positive class have the higher precision and recall score. And the intuitive reason behind it is that these texts contain an unambiguous language that clearly stands on it's own. Either the review author disapproves the business completely or approves of it completely, making it easier for humans or algorithms to identify the category with a high degree of confidence. Such can not be said of the review texts that belong to the categories of 2, 3 and 4 that might contain a mix of words making the overall sentiment ambiguous to judge.

Table 5: Accuracies obtained using Deep Learning approaches

	Word Embeddings
Convolutional Neural Network	62.27%
Long Short-Term Memory	65.28%
Gated Recurrent Units	64.65%
Gated Recurrent Units (Bidirectional)	65.11%
Gated Recurrent Units (Bidirectional) + CNN	62.17%
Long Short-Term Memory ((Bidirectional)) x 2 [Deep LSTM]	65.53%

Table 6: Classification report using Deep LSTM

	Precision	**Recall**	**F1-score**	**Support**
1	0.74	0.78	0.76	104516
2	0.57	0.59	0.58	104670
3	0.61	0.54	0.57	105104
4	0.61	0.59	0.60	105439
5	0.75	0.79	0.77	104908
Macro avg	0.65	0.66	0.66	524637
Weighted avg	0.65	0.66	0.66	524637

5 Conclusion

Memory-based models are an easy to use models as they do not require any training or optimization. They are easy to interpret and analyze however, their performance decreases as the sparsity of the data increases which in turns hinders scalability of these models in real world.

With respect to the prediction of ratings from review text alone, the recent advances in NLP using Deep Learning indicate promising results, and it is expected that word embedding based neural networks will improve upon the achievable accuracy. From our experiments, we were able to achieve accuracy of around 65% using two Bidirectional LSTM layers and word embeddings as our features.

6 Future Work

Task 1

- Combining Model based and Memory based approaches.
- Exploring more variables for implicit interactions

Task 2

- Improvements in word embeddings: Glove, fast-text, and many others to choose from..
- Combining TF-IDF features with word embeddings.

References

[1] Yehuda Koren *Factor in the Neighbors: Scalable and Accurate Collaborative Filtering.* 2010

[2] Michael Reusens, Wilfried Lemahieu, Bart Baesens, Luc Sels *Evaluating recommendation and search in the labor market.* 2018

[3] ttps://fasttext.cc/docs/en/english-vectors.html

[4] ttps://surprise.readthedocs.io/en/stable/knn_inspired.html?highlight=KNNBasic#actual-k-note

[5] ttps://surprise.readthedocs.io/en/stable/similarities.html?highlight=cosine#surprise.similarities.cosine