GitHub Repositories - Analysis and Recommendation

Guided By: Prof. Magdalini Eirinaki

Team Vidicators- Animesh Grover
Mustafa Bandukwala
Venkatesh Devale

Motivation and Dataset

- GitHub - Best place to develop and share software in today's world
- We face challenges in knowing the current trends in programming world

  - what are current trending programming languages?

  - what are trending projects related to projects you have followed?
- Why we chose GitHub?
- Dataset : Size - 3 TB, Used - languages and sample_repos tables
  https://www.kaggle.com/github/github-repos

Use Cases

- Help user with top three programming language recommendation based on user's current choice of language
- Help user with top *k* repositories which user will likely follow based on his starred repositories
- Evaluation with graphical analysis of repository and languages to detect the connection between the repositories

Solution - First Case

- Recommending top three programming languages
  - Collaborative filtering using Gradient Descent Algorithm
  - We build a matrix of repository vs languages, where we assign 1 if user has used that language in his repository else 0
  - The formula we have used is $Y = X\Theta^T$
  - Y is the matrix built in second step, X is the input matrix of repository and languages and we have to find Θ(weight vector) in order to get closer to Y
  - We then use gradient descent algorithm to get Θ

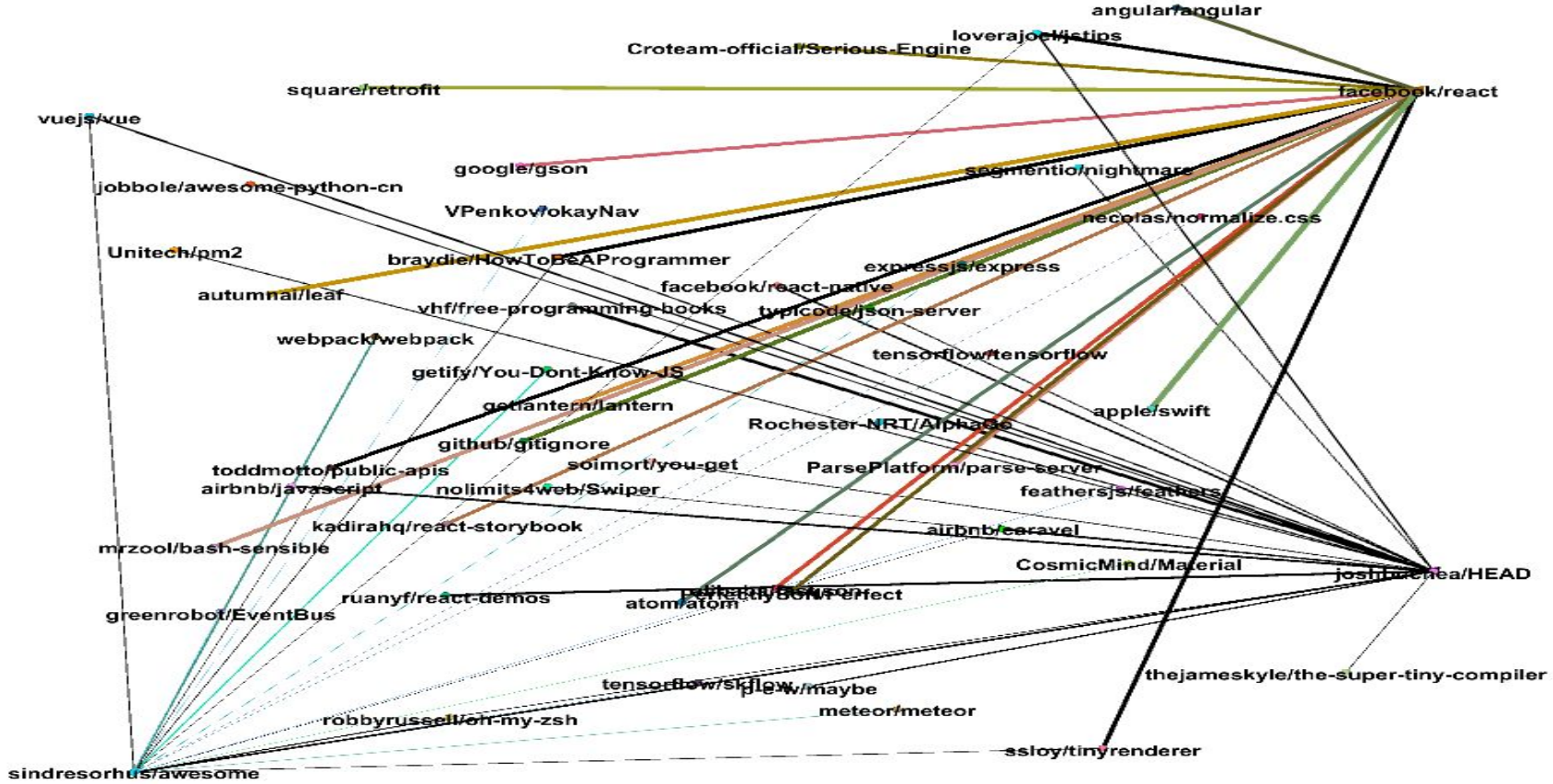
- Why Gradient Descent?
  - We had to update our weight vector in each iteration to approximate Y
  - Gradient descent is used to minimize the difference between Y and our hypothesis i.e $X\Theta^T$
  - This is why we implemented gradient descent using Scipy library's optimize function

Solution - Second Case

- Recommending top *k* repositories based on ‘stars’
  - Collaborative filtering using Matrix Factorization.
  - Here we found that, our user - repository matrix is a very sparse matrix.
  - That is why we decided to go for matrix factorization in this use case.
  - Prepared User - Repository matrix with stars as the connecting values.
  - Used SVD from Surprise Library for getting the predictions of stars and the likelihood of user following the recommended repository in future

- Why SVD?
  - We compared different models and SVD gave us the best diverse set of repositories with highest accuracy

Evaluations - Graphical Analysis

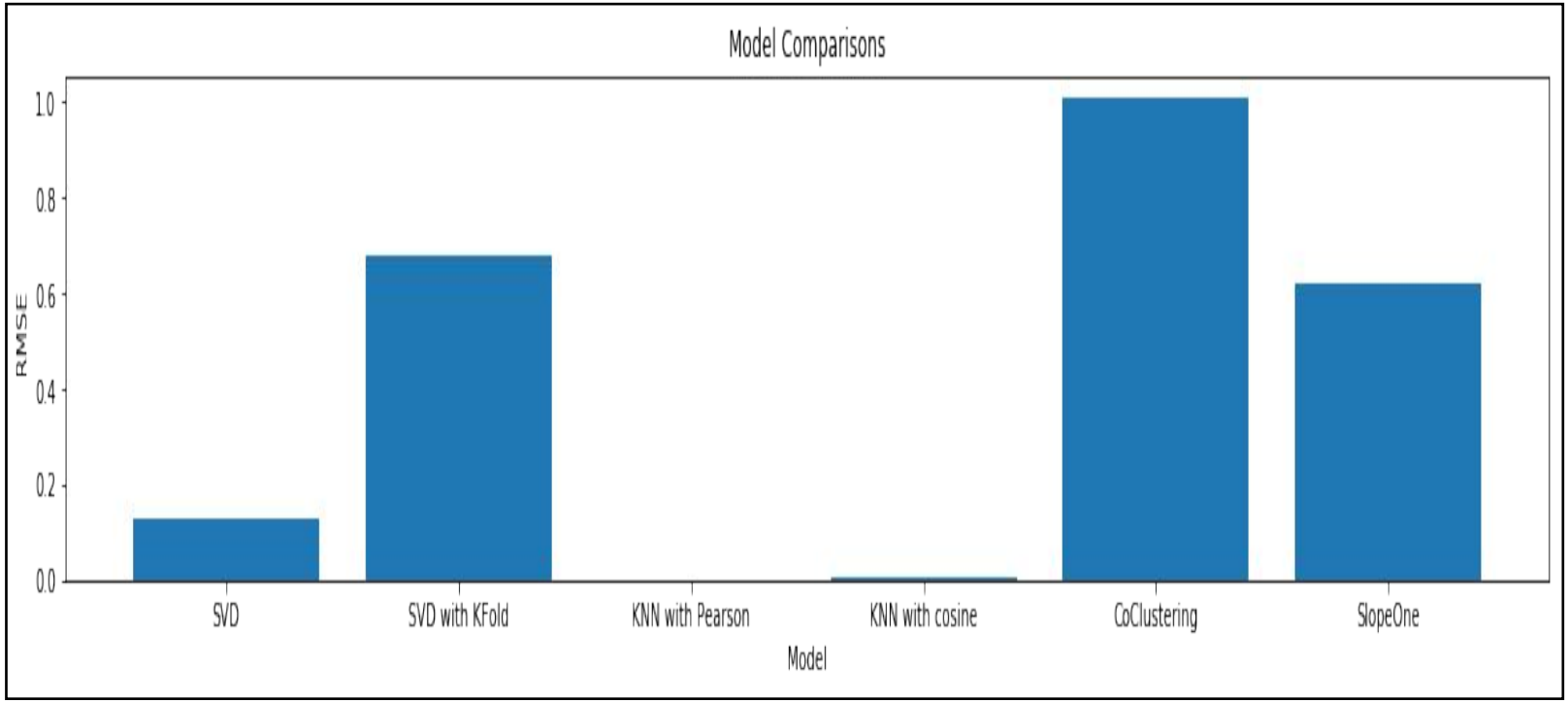


Data Preprocessing

- Data Collection:
  - Used Google’s BigQuery for data collection

- Data Preprocessing:
  - Removed repositories with single language
  - Removed an outlier repository which had the largest number of stars
  - Further we have considered only those repositories which have more than two stars
  - The range of stars was from 1000 - 13000, we have used ‘Divide by Mean’ normalization to get the data in range 0 - 6

Evaluations



How things worked

- Things that worked:
  - We designed the system using Google's BigQuery so we did receive the updated data every time we queried, our helper functions implementation is very robust as the implementation uses new updated data each time hence this worked very well for us.
  - The matrix factorization implementation using SVD worked pretty well with a very low RMSE compared to Slope One and Co Clustering recommendation algorithms.

- Things that did not worked well:
  - We aimed to get the current data using GitHub's API but could not get it due to the GitHub APIs restricted access, so this approach did not worked for us
  - KNN implementation with 'cosine' and 'pearson' similarities gave us same repository recommendations for any user you pick, so it did not worked as we expected.

Conclusion

- We got to learn that large data and large computational power are two required aspects in building great recommender systems.
- Matrix factorization works pretty well when you have user, item and rating combination, but when you have metadata related to repository or user you have to explore the data a lot and do a lot of data preprocessing, and then feature engineering with techniques like feature scaling, extraction etc.
- From our study we also found model selection and prediction is comparatively easy part compared to data gathering, exploration, preprocessing and extraction.

Thank You, Questions Please?