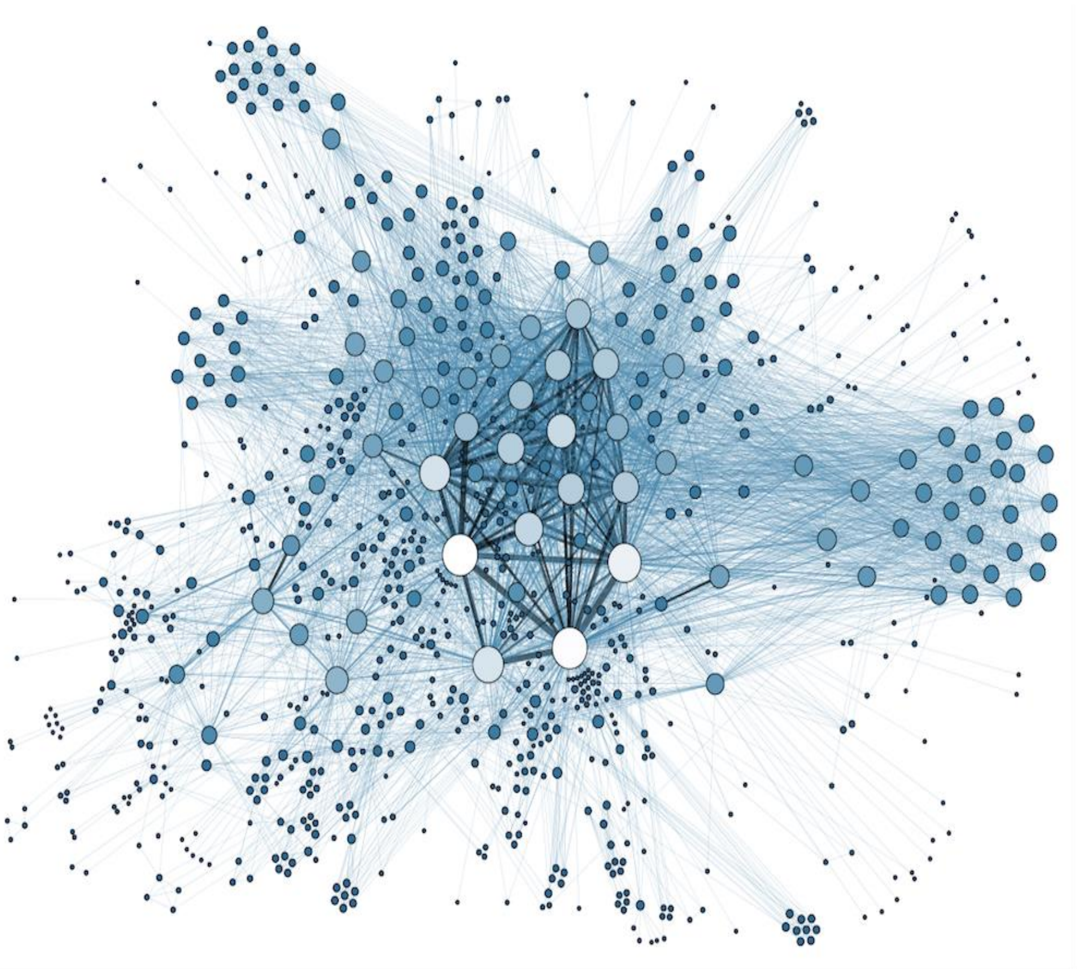
Microsoft Academic Graph



Citation Recommendation

USING TITLE SIMILARITY AND COLLABORATIVE FILTERING BY MATRIX FACTORIZATION

Siddharth Jayasankar (sidjayas), Ganesh Nagarajan (gnagaraj), Neha Bisht (nbisht), Arun Ram (arunshank), Nishant Shah (nishshah)

Z-604 – Big Data Analytics for the Text and Web | May 3, 2016

Contents

Introduction

Citation is recognition of the original work done by the author and people adopting and extending the idea into new horizons. An academic style papers require citations to credit the idea to an author and the more citation a paper gathers in a sense explains the impact of the paper in their respective field. . However with the ever increasing number of papers being published, we find it hard to find, right papers that could help for the research. It would be helpful if we had some tool which could recommend us useful papers to cite. Thus in this project we have proposed an innovative method to provide recommendations to the author based on the paper title and key words. It is hard to provide good quality citations even with full paper context. We have gone beyond the usual and have taken up the challenge to explore the feasibility of providing good quality recommendations with only the key words associated with the paper and paper title. We have used a combination of "title similarity" and "Context Citations" with collaborative filtering by matrix factorization to come up with a citation recommendation system.

Main Idea of the Project

This paper explores hypothesis for citation recommendation, first being if two papers share a similar title, then the papers being cited would also contain a common theme / topic. It is due to this hypothesis that we calculate the similarity of titles and provide similar titled papers as inputs for the collaborative filtering.

The other hypothesis is that titles as Bag of Words representation by itself cannot have enough information to retrieve similar papers. We introduce a concept called "context citations". This concept is similar to word2vec's philosophy "The word is explained by the company that it chooses", we propose "A paper is characterized by the paper that it cites" more than the title similarity. This is also a smart way of avoiding the cold start problem by giving the initial inputs for the recommender system.

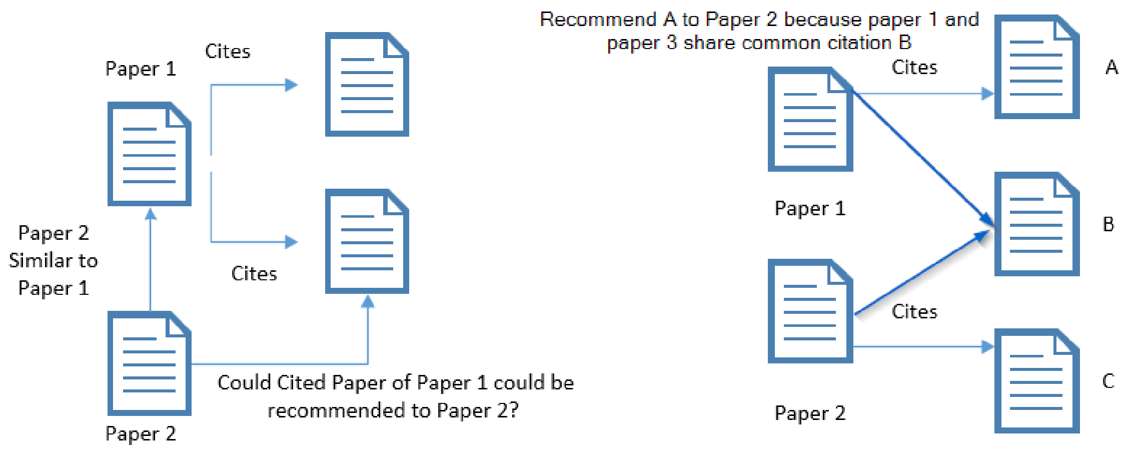

Paper 1
Cites
Paper 2
Similar to
Paper 1
Cites
Paper 2
Could Cited Paper of Paper 1 could be
recommended to Paper 2?
Recommend A to Paper 2 because paper 1 and
paper 3 share common citation B
Cites
A
Paper 1
B
Cites
C
Paper 2

Fig 1. Pictorial representation of Hypothesis 1 and 2

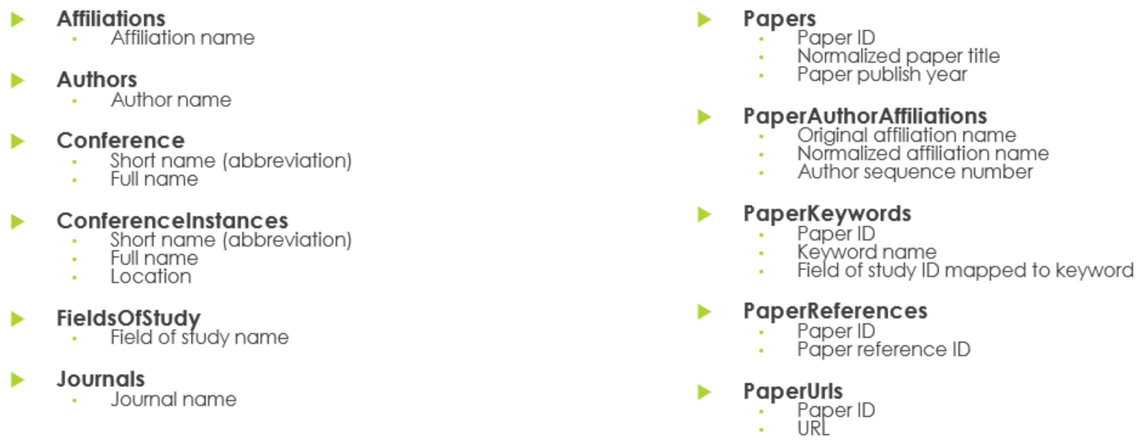
Collaborative Filtering is currently the most used approach to build recommendation systems. The major problem faced by these CF methods is the sparsity of the rating matrix. In this project we have observed the matrix to be sparse. Thus to deal with this sparsity we have adopted the matrix factorization technique.

Data Description

The data that we used is the Microsoft Academic graph dataset which is a heterogeneous graph that had information relating to a collection of scientific publication records and also the relationships between the records in that collection. The major features in the dataset include

- Authors
- Institutions
- Journals and venues
- Field of study

The initial investigation showed that the dataset as of February 2014 had information of up to 39.9 million publications and about 18.8 million authors. Fig 1 shows a description of the dataset. Based on the initial investigation on the dataset, we were able to generate a subset of the papers in the dataset based on their field of study. Information Retrieval and Machine Learning were the two fields of study that we considered for building our citation recommendation system.



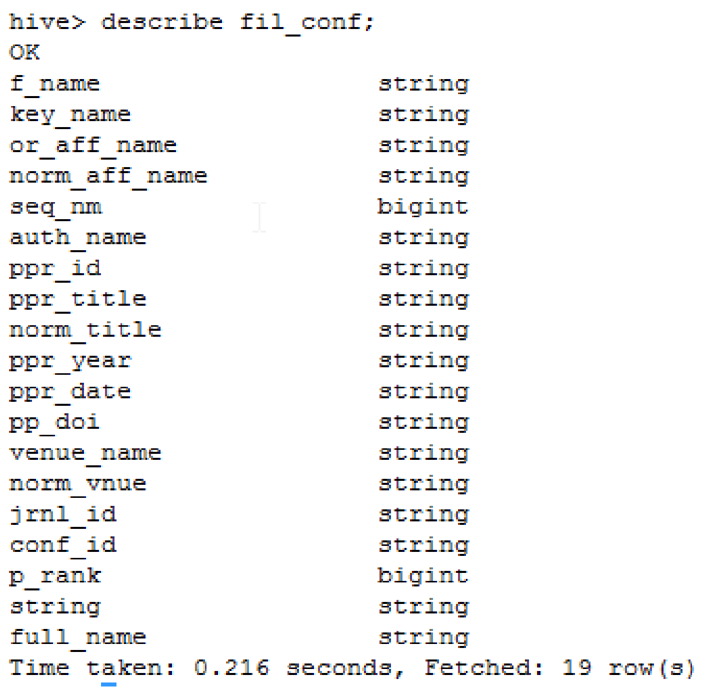
- **Affiliations**
  - Affiliation name
- **Authors**
  - Author name
- **Conference**
  - Short name (abbreviation)
  - Full name
- **ConferenceInstances**
  - Short name (abbreviation)
  - Full name
  - Location
- **FieldsOfStudy**
  - Field of study name
- **Journals**
  - Journal name
- **Papers**
  - Paper ID
  - Normalized paper title
  - Paper publish year
- **PaperAuthorAffiliations**
  - Original affiliation name
  - Normalized affiliation name
  - Author sequence number
- **PaperKeywords**
  - Paper ID
  - Keyword name
  - Field of study ID mapped to keyword
- **PaperReferences**
  - Paper ID
  - Paper reference ID
- **PaperUrls**
  - Paper ID
  - URL

Fig 2. Dataset Description

Sampling of data

Since the data is relational, random sampling or reservoir sampling wouldn't return the whole relationship. Hence we require robust relational databases based joins to retrieve the dataset. The whole 100 GB of data from the Microsoft graph dataset was loaded into Hadoop and subset of all papers from "Information retrieval and Machine Learning" was done using SQL relational queries. This subset shall act as the one point of truth from now on. The original data is then removed from the Hadoop to conserve storage cost.

Following is the subset of the schema that was used in this project. This schema was joined with the paper reference table which is a dual column bridge table joining paper id to the cited paper id. These now represent the papers and their relationship with each other.



```
hive> describe fil_conf;
OK
f_name                  string
key_name                string
or_aff_name             string
norm_aff_name           string
seq_nm                  bigint
auth_name               string
ppr_id                  string
ppr_title               string
norm_title              string
ppr_year                string
ppr_date                string
pp_doi                  string
venue_name              string
norm_vnue               string
jrnl_id                 string
conf_id                 string
p_rank                  bigint
string                  string
full_name               string
Time taken: 0.216 seconds, Fetched: 19 row(s)
```

Fig 3. Format of sampled data

Tools Used

Hadoop – The uncompressed data consisted of eleven files and consisted of more than 110 GB of data. Thus to help us join the required fields and to extract a smaller subset of data to carry out our project we dumped the entire data into Hadoop.

Hive – Hive offers a structured way to query and join these disparate files. To combine different fields from different files and to subset the data into a smaller subset of data which consists of papers related to machine learning and information retrieval, hive was used.

Lucene – Retrieving data from a Lucene index is very fast calculating the similarity between titles could be done in almost real time using Lucene. Thus to calculate title similarity Lucene was used.

Spark - Since we are dealing with big data we decided to use spark to take advantage of parallel processing capability and also to utilize the machine learning package which has collaborative filtering implementation.

Scala – Due to easy integration with Spark we chose to work with Scala. Also functional programming in our opinion fits more to the map reduce paradigm than declarative programming.

Neo4j – The page rank is used to order the papers based on their popularity. This can also be used to rank our predictions. Thus find popular papers and to help in raking the predicted result page rank was used.

Python – We code in python to check how good our recommendation system performed. We calculated precision and recall using a python code.

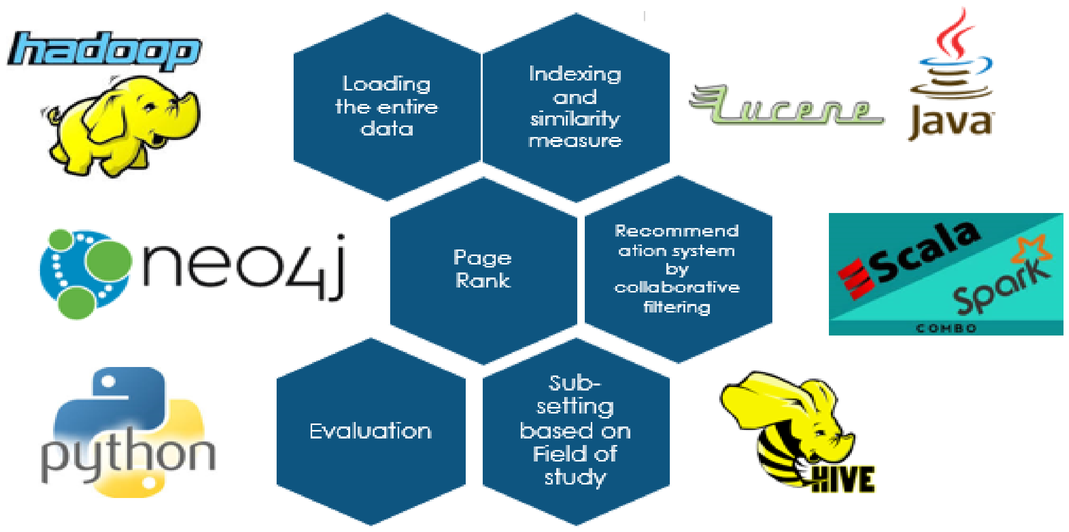

Loading the entire data
Indexing and similarity measure
Page Rank
Recommendation system by collaborative filtering
Evaluation
Sub-setting based on Field of study

Fig 4. Over view of tools used.

Title Similarity using Lucene

Indexing Using Standard analyzer

In order to calculate similarity between titles, we had to first index all the paper titles. For indexing paper titles we used Lucene's standard analyzer. The Paper ID, Paper Title were stored as text fields. The standard analyzer performs the below two tasks before indexing the contents

1. *Stop Word Filtering* - Stop words are small words that appears often inside a text and therefore does not have any title specific relevance. Filtering out those words will lead to a term list that contains proportionally more relevant

terms. Apache Lucene contains a list of stop words with the standard analyzer which it uses to filter stop words

2. *Tokenizing* - Tokenization refer to the process of splitting up a text in to small pieces (terms) which then will be used to match search query and documents against each other. Thus each title is reduced to an array of words which are present in the title.

Similarity scoring using Default similarity

The title similarity is calculated using Lucene's Default similarity. Similarity is based on four components

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t \in q} \{ tf(t \in d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d) \}$$

$$coord(q,d) = \frac{|q \cap d|}{|q|} \qquad queryNorm(q) = \frac{1}{\sqrt{\sum_{j+1}^{|q|} q_j^2}} \qquad tf(t \in d) = \sqrt{freq(t)} \qquad idf(t) = 1 + \log\left(\frac{numDocs}{1 + docfreq}\right)$$

Fig 4. Scoring formula of Lucene's default similarity

1. *Coord(q,d)* – number of terms present in both the titles . Thus if there are 4 words common in the query and the document and if the number of words in the query is 7 then this component would return a value of 0.5714
2. *QueryNorm(q)* – normalized length of the citing paper.
3. *Term frequency(t ϵ d)* – It corresponds to the number of terms that appear in the current document d. Thus d. Documents that have occurrences of a given term receive a higher score. However the *tf(t ϵ q)* is assumed to be one and hence it does not appear in the formula .Thus if a term appears twice in the query there will be 2 calculations for the same term thus consistency of the formula is maintained.
4. *Inverse Document frequency (t)* - This component provides higher scores to the rare words. Thus if a word which appears rare in both the query and the document the score for that query and document will be high. The *idf* value is squared because the term appears in both the query and the document

Once we have scored every query document pair, the top 5 scores are retrieved and these 5 scored documents would give the 5 similar paper titles to the give citing paper title. These 5 similar titled papers are then fed to the collaborative filtering engine to overcome the cold start problem

CF using Matrix Factorization

Collaborative Filtering

Collaborative filtering has become one of the most popular methods used in designing a recommendation system. The most important element of data that is being considered in this technique is the data relating to past user experiences. The technique first collates past user experiences in order to establish a connection between the users and the items. From these recommendations are made to each user based on the opinions of the other users in the list. It is typically based on the presumption that those users who had similar likings in the past will have similar likings in the future.

There are two types of collaborative filtering 1) User based and 2) Item based. The user based collaborative filtering is based on the similarities between the users while the item based collaborative filtering is based on the similarity of items being purchased by the user in a retail system.

With the enormous amount of data that is being processed everyday it has become a challenge to implement collaborative filtering techniques on larger datasets. Hence the technique of matrix factorization is being implemented in these cases. The sparse nature of the rating matrix and the huge amount of data both hamper the performance of collaborative filtering. Matrix factorization helps to eliminate these hurdles that arise due to the enormity of the data. The concept of low rank matrix factorization is being employed in our case to measure the fit between our given data and our approximation data, based on the constraint that our approximation data has a lower rank.

Low Rank Matrix Factorization Model

Matrix decomposition is a standard problem in information retrieval. Even though there are multiple ways to achieve matrix decomposition and dimensionality reduction including PCA, SVD and more recent efforts including neural networks for matrix decomposition, we chose Alternating least squares method for matrix factorization. The implicit problem with the SVD for rank decomposition is it doesn't work well under sparse matrix. If the document is really sparse, the matrix cannot be factorized using Eigen decomposition since the determinant of the matrix becomes zero.

ALS method tries to minimize the mean squared error of the known implicit ratings. The term Low-Rank matrix factorization implies that we are trying to reduce the dimensionality of the matrix from items x products to two matrix of items x k and k x products where k is the hidden or latent set of variables. ALS is an optimization problem such that the following equation holds for a Matrix A ∈ R^{mxn} and a desired rank k such that k << min(m,n), low rank approximation of A is,

$$A \approx WH \; st \; W, H \geq 0$$

$$\min f(w,h)_{WH} = \frac{1}{2} \|A - WH\|^2_F \; such \; that \; WH \geq 0$$

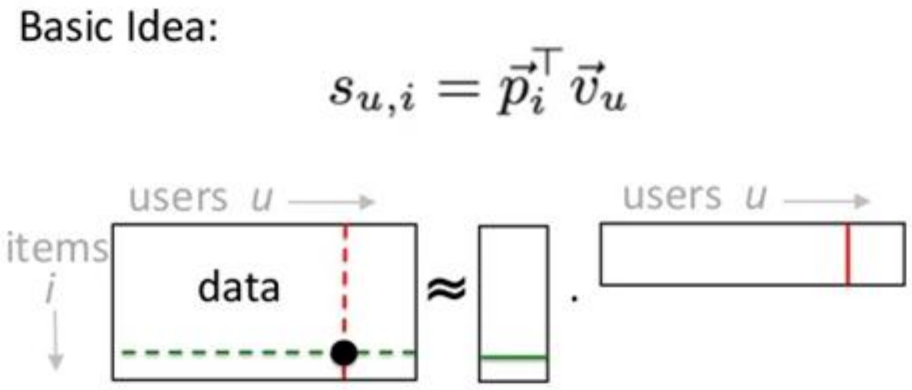


Fig 5. Low Rank Matrix Factorization Model

Our program consisted of an iterative way to search across different parameters, our search space was Rank of the matrix from 5-50, the regularization parameter 0.1-1 in steps of 0.05 and alpha, the generalization parameter in steps of 50 from 50 to 3000. This is a significantly large search space to search for optimized parameters, however because of the parallel framework of spark and multiple worker nodes, we were able to zero in on a model in about 3 hours using Intel i5 laptop with 4GB RAM and 3 parallel workers.

The final model has the MSE of 0.06, considering the rank of 30, Trained in 25 Iterations using alpha values as 3000 and lambda as 0.1. This model is then used for the recommender using the recommend products function.

Implementation

We have incorporated two methods using the above mentions concepts of title similarity and collaborative filtering using low rank matrix factorization. The algorithm and the evaluation of these two approaches are described in this section.

Method 1

ALGORITHM:

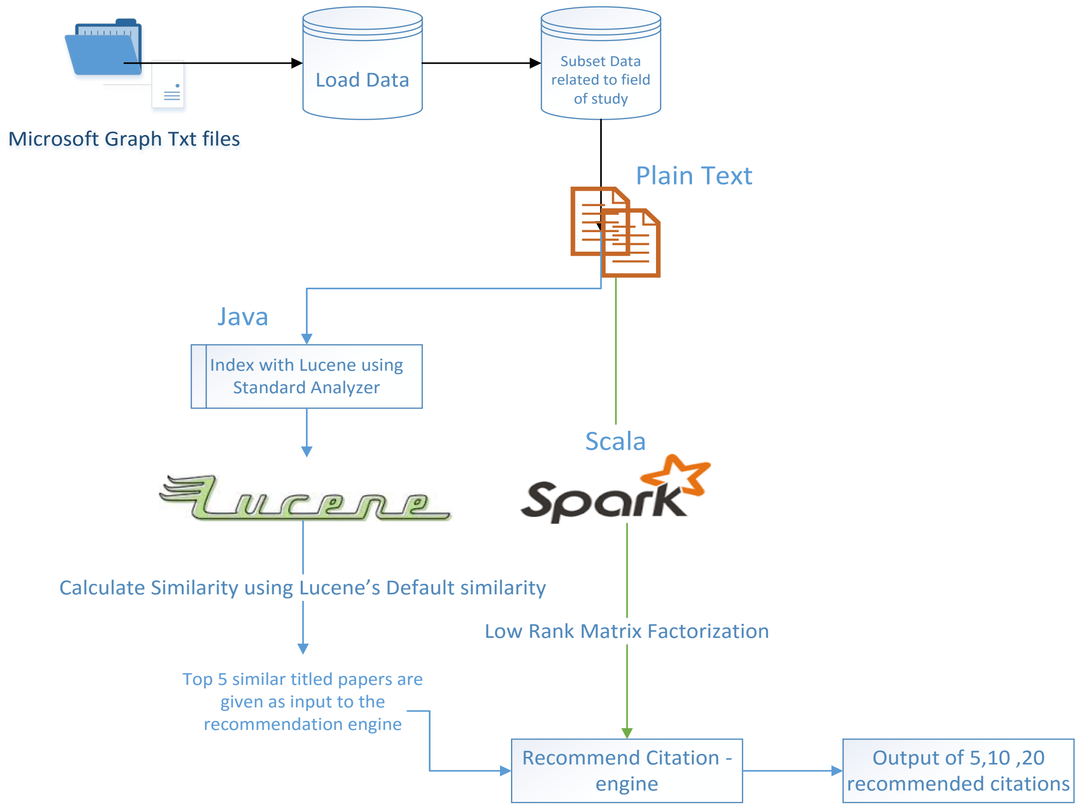
Step 1 > Using Lucene's standard analyzer index all the papers

Step 2 > Using Lucene's default similarity score all the cited papers for each citing paper

Step 3 > Based on the scores, retrieve top 5 similar cited papers for each citing paper

Step 4 >Feed these 5 similar titled cited papers as input to the recommendation engine

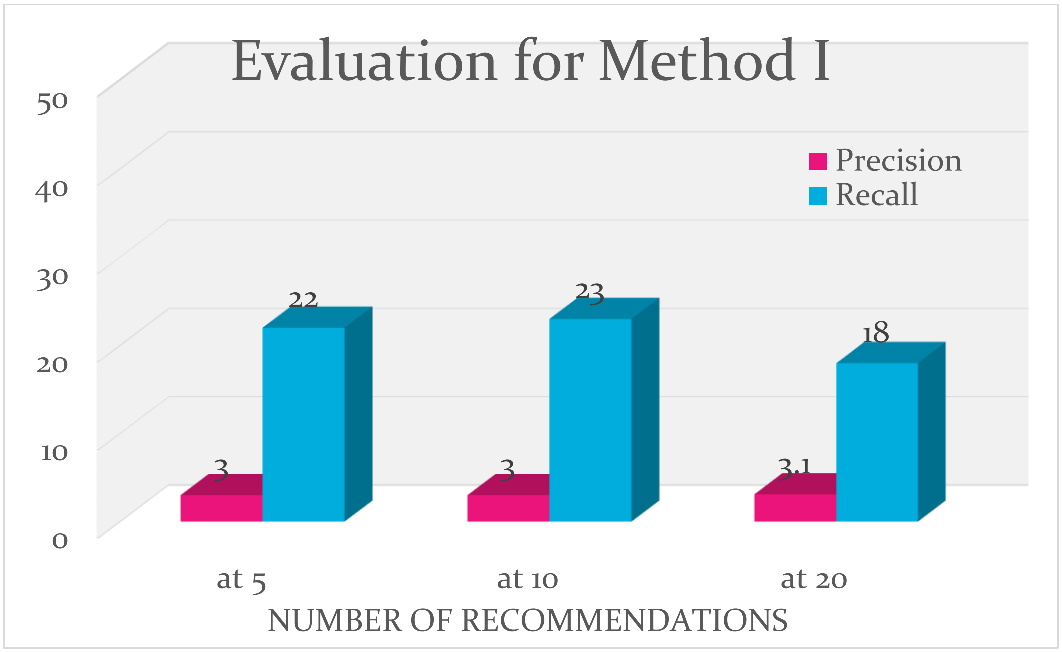
Step 5>For these inputs using collaborative filtering provide 5, 10, 20 recommendations for each citing paper



Load Data
Subset Data related to field of study
Microsoft Graph Txt files
Plain Text
Java
Index with Lucene using Standard Analyzer
Scala
Calculate Similarity using Lucene's Default similarity
Low Rank Matrix Factorization
Top 5 similar titled papers are given as input to the recommendation engine
Recommend Citation - engine
Output of 5,10 ,20 recommended citations

Fig 7. Flow chart for Method 1

EVALUATION



For this method we have a constant precision of 3% but the recall is decent at around 21% .

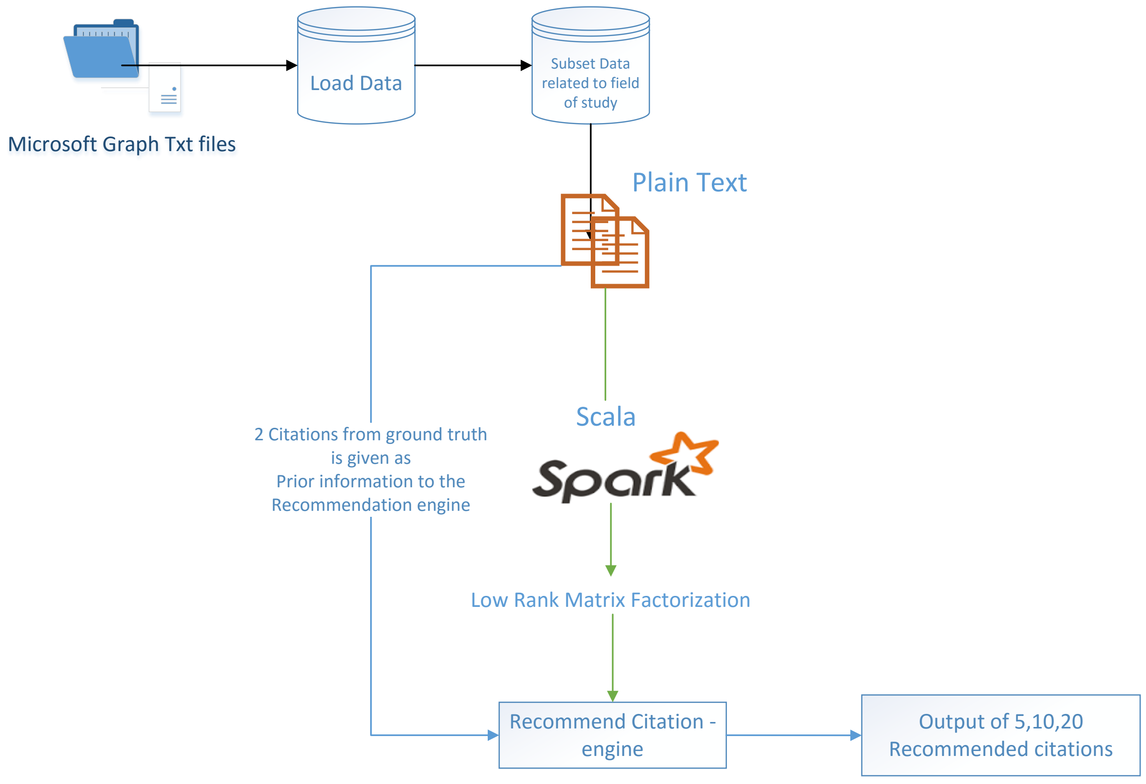
The results of this experiments are quite interesting. We have ran the proposed algorithm thrice for every values and the averaged results are presented in the above graph. It can be seen that the recall is almost similar over multiple values, however there seems to be change in recall. We have a recall of 22% at 5, 23 % at 10 and 18% at 20. 10 seems to be the best balance while trying to recommend citations to the user.

Method 2

ALGORITHM:

Step 1 > For each citing paper extract 2 actual citations from the ground truth and feed it to the recommendation engine

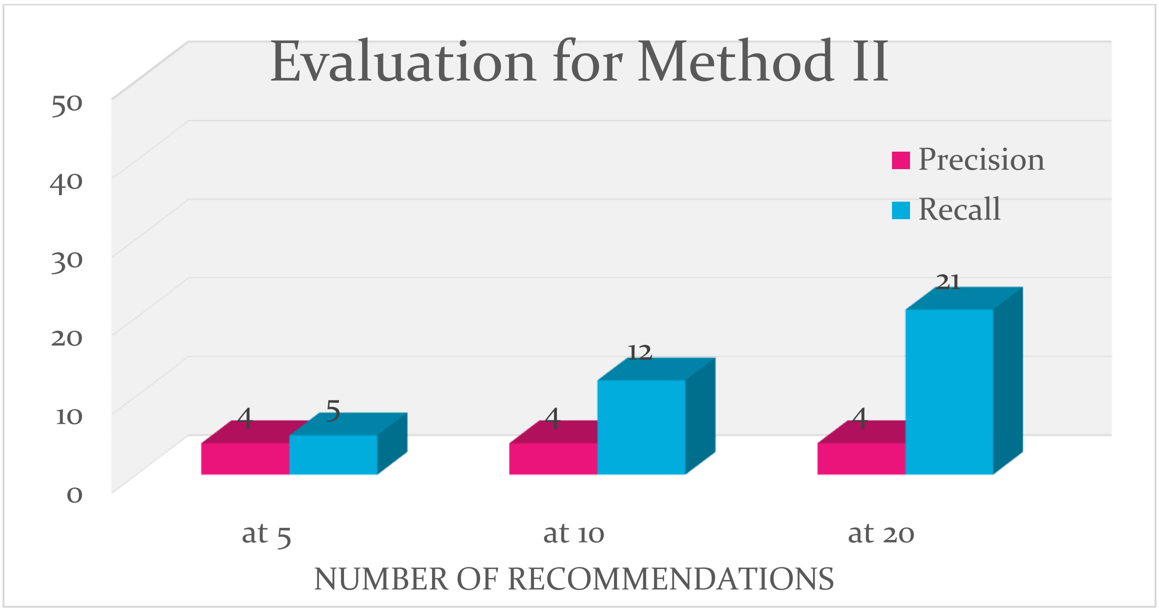
Step 2 > Using these ground truths as inputs provide recommendations of 5, 10 ,20 papers for each citing paper



Microsoft Graph Txt files
Load Data
Subset Data related to field of study
Plain Text
Scala
Spark
2 Citations from ground truth is given as Prior information to the Recommendation engine
Low Rank Matrix Factorization
Recommend Citation - engine
Output of 5,10,20 Recommended citations

Fig 7. Flow chart for Method 2

EVALUATION



This method we have a better precision than the previous method discussed, however there is a big drop in the recall. The recall at 5 recommendation is 5%, 12%

for 10 and 21% for 20. Thus contradicting to our choice of the first algorithm, more the predictions in method two offers better performance.

However, this method we did not get a great deal of increase in precision. Also the recall has dropped considerably suggesting that the hypothesis of method 1 (that similar titled papers provides good information about the citing paper) is valid.

Method 3 (proposed)

ALGORITHM:

Step 1 > Load the data into neo4j and calculate the page rank of all the cited papers

Step 2 > For each citing paper retrieve the top 2 cited paper from the ground truth

Step 3 > Using these top 2 ranked ground truths as inputs recommend 5, 10, 20 papers for each citing paper

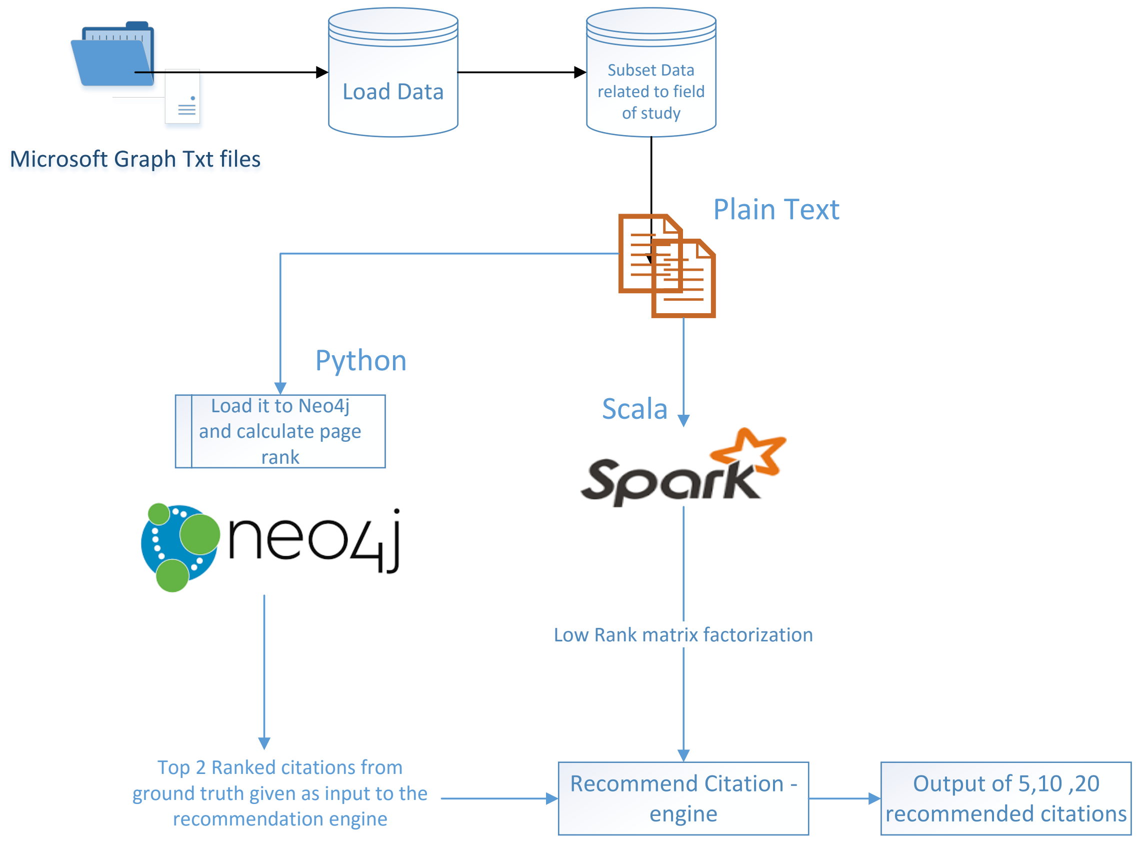

Load Data
Subset Data related to field of study
Microsoft Graph Txt files
Plain Text
Python
Load it to Neo4j and calculate page rank
Scala
Spark
neo4j
Low Rank matrix factorization
Top 2 Ranked citations from ground truth given as input to the recommendation engine
Recommend Citation - engine
Output of 5,10 ,20 recommended citations

Fig 8. Flow chart for method 3

Selected Results

Query : an ensemble of grapheme and phoneme for machine transliteration (language)
Context 1: english to korean statistical transliteration for information retrieval
Context 2: automatic identification and back transliteration of foreign words for information retrieval
Context 3: automatic transliteration and back transliteration by decision tree learning
Result 1: japanese english cross language information retrieval exploration of query translation and transliteration
Result 2: an english to korean transliteration model of extended markov window
Result 3: an ensemble of transliteration models for information retrieval
Result 4: mining the web to create a language model for mapping between english names and phrases and Japanese
Result 5: backward transliteration for thai document retrieval
Result 6: uci repository of machine learning databases
Result 7: generating phonetic cognates to handle named entities in english chinese cross language spoken document retrieval

It can be seen that the recommendations suggested by the algorithm is indeed relevant and on the same topic. An interesting observation was that more unique the "Context citation" or the "Similarity Title", more precise the recommendations. E.g., a title like Application of n-gram model for information retrieval in Asian languages had better recommendations than decision trees for information retrieval.

Conclusion

Providing the similar paper titles using Lucene(Method 1) to recommendation system performed better results when compared with other method where 2 ground truth papers where provided to the recommendation system (Method 2) for doing the prediction. This result clearly proves the hypothesis that similar papers with similar titles can be used to predict useful citations.

However using only the titles of the paper alone has limitations as the information conveyed by the title is very less as compared to the abstract / full text of the paper.

Since we were able to achieve a decent recall with just the title of the paper we can confirm that the method we proposed is sound and this frame work is bound to do better if provided with paper abstract / full paper texts.

Future work

We intend to implement method 3 and compare it with performance of the other 2 methods. We also would like to rank the recommendations and use NGCD as an evaluation metric to see how good our ranked results are.

References

- http://www.cse.psu.edu/~duk17/papers/citationrecommendation.pdf
- http://www.lucenetutorial.com/advanced-topics/scoring.html
- http://ipl.cs.aueb.gr/stougiannis/default.html
- https://en.wikipedia.org/wiki/Low-rank_approximation
- http://www.slideshare.net/haraldsteck/gaussian-ranking-by-matrix-factorization-acm-recsys-conference-2015
- http://www.sciencedirect.com/science/article/pii/S1877050915007462
- http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html