* Copy all the input and output files from the “TF\_IDF (Python)” folder to your Neo4j project import folder.
* You can find the files needed for Neo4j in the “4. Recommendation (Neo4j)/import” folder.
* You can find the findal results of different output in the xlsx files in the “4. Recommendation (Neo4j)/results” folder.

### Loading Data ###

1. Load user data in “Users” nodes.

LOAD CSV WITH HEADERS FROM 'file:/user.csv' AS row

FIELDTERMINATOR '\t'

CREATE (:Users {user\_id: row.user\_id, cnt: row.cnt, attr: row.attr})

1. Load book data “Books” nodes.

LOAD CSV WITH HEADERS FROM 'file:/book.csv' AS row

FIELDTERMINATOR '\t'

CREATE (:Books {book\_id:row.book\_id, authors:row.authors, year:row.year, title:row.title, language:row.language})

1. Load rating data in “Rated” relationship. [Users-Rated->Books]

LOAD CSV WITH HEADERS FROM 'file:/rating.csv' AS row

FIELDTERMINATOR '\t'

MATCH (u:Users {user\_id: row.user\_id})

MATCH (b:Books {book\_id: row.book\_id})

MERGE (u)-[r:RATED]->(b)

ON CREATE SET r.rating = toInteger(row.rating)

1. Load IDF values for all book features in “IDF\_title, IDF\_authors, IDF\_language, IDF\_year” nodes.

LOAD CSV WITH HEADERS FROM 'file:/IDF\_title.csv' AS row

FIELDTERMINATOR '\t'

CREATE (:IDF\_title {word: row.word, IDF: row.IDF\_val})

LOAD CSV WITH HEADERS FROM 'file:/IDF\_authors.csv' AS row

FIELDTERMINATOR '\t'

CREATE (:IDF\_authors {author: row.author, IDF: row.IDF\_val})

LOAD CSV WITH HEADERS FROM 'file:/IDF\_language.csv' AS row

FIELDTERMINATOR '\t'

CREATE (:IDF\_language {language: row.language, IDF: row.IDF\_val})

LOAD CSV WITH HEADERS FROM 'file:/IDF\_year.csv' AS row

FIELDTERMINATOR '\t'

CREATE (:IDF\_year {year: row.year, IDF: row.IDF\_val})

1. Load TF values for all book features in “TF\_title, TF\_authors, TF\_language, TF\_year” relationship [Books-TF…->IDF…].

LOAD CSV WITH HEADERS FROM 'file:/TF\_title.csv' AS row

FIELDTERMINATOR '\t'

MATCH (b:Books {book\_id: row.book\_id})

MATCH (i:IDF\_title {word: row.word})

MERGE (b)-[v:TF\_title]->(i)

ON CREATE SET v.val = toFloat(row.TF\_title\_val)

LOAD CSV WITH HEADERS FROM 'file:/TF\_authors.csv' AS row

FIELDTERMINATOR '\t'

MATCH (b:Books {book\_id: row.book\_id})

MATCH (i:IDF\_authors {author: row.author})

MERGE (b)-[v:TF\_authors]->(i)

ON CREATE SET v.val = toFloat(row.TF\_author\_val)

LOAD CSV WITH HEADERS FROM 'file:/TF\_language.csv' AS row

FIELDTERMINATOR '\t'

MATCH (b:Books {book\_id: row.book\_id})

MATCH (i:IDF\_language {language: row.language})

MERGE (b)-[v:TF\_language]->(i)

ON CREATE SET v.val = toFloat(row.TF\_language\_val)

LOAD CSV WITH HEADERS FROM 'file:/TF\_year.csv' AS row

FIELDTERMINATOR '\t'

MATCH (b:Books {book\_id: row.book\_id})

MATCH (i:IDF\_year {year: row.year})

MERGE (b)-[v:TF\_year]->(i)

ON CREATE SET v.val = toFloat(row.TF\_year\_val)

1. Load User profiles for all book features in “UA\_title, UA\_authors, UA\_language, UA\_year” relationship [Users->UA… ->IDF…] .

LOAD CSV WITH HEADERS FROM 'file:/UA\_title.csv' AS row

FIELDTERMINATOR '\t'

MATCH (u:Users {user\_id: row.user\_id})

MATCH (i:IDF\_title {word: row.word})

MERGE (u)-[v:UA\_title]->(i)

ON CREATE SET v.val = toFloat(row.UA\_title\_val)

LOAD CSV WITH HEADERS FROM 'file:/UA\_authors.csv' AS row

FIELDTERMINATOR '\t'

MATCH (u:Users {user\_id: row.user\_id})

MATCH (i:IDF\_authors {author: row.author})

MERGE (u)-[v:UA\_authors]->(i)

ON CREATE SET v.val = toFloat(row.UA\_author\_val)

LOAD CSV WITH HEADERS FROM 'file:/UA\_language.csv' AS row

FIELDTERMINATOR '\t'

MATCH (u:Users {user\_id: row.user\_id})

MATCH (i:IDF\_language {language: row.language})

MERGE (u)-[v:UA\_language]->(i)

ON CREATE SET v.val = toFloat(row.UA\_language\_val)

LOAD CSV WITH HEADERS FROM 'file:/UA\_year.csv' AS row

FIELDTERMINATOR '\t'

MATCH (u:Users {user\_id: row.user\_id})

MATCH (i:IDF\_year {year: row.year})

MERGE (u)-[v:UA\_year]->(i)

ON CREATE SET v.val = toFloat(row.UA\_year\_val)

### Recommend Books for Registry to the couple ###

1. Store average rating by each user of type ‘Couple’ and ‘Married’ [Needed for Pearson Similarity] in “avg\_rating” property of Users

MATCH (u:Users)-[r:RATED]->(b:Books)

WHERE u.attr = 'M' OR u.attr = 'C'

WITH u, toFloat(SUM(r.rating))/COUNT(r) as avg\_rating

SET u.avg\_rating = avg\_rating

1. Calculate Cosine, Pearson, & MeanSquare Similarity of each user of type ‘Couple’ to ‘Married’ in “Similarity” relationship [Users(c) – Similarity – Users(r)].   
   And put the similarity values in “consine”, “pearson”, “meansquare” properties in the “Similarity” relationship.

MATCH (u1:Users{attr:'C'})-[x:RATED]->(b:Books)<-[y:RATED]-(u2:Users{attr:'M'})

WITH u1, u2,

COUNT(b)/SUM((x.rating-y.rating)^2) as MeanSquare,

SUM(x.rating \* y.rating) AS xy,

SQRT(SUM(x.rating^2)) as sqrtx2,

SQRT(SUM(y.rating^2)) as sqrty2,

SUM((x.rating-u1.avg\_rating) \* (y.rating-u2.avg\_rating)) AS xx\_yy\_,

SQRT(SUM((x.rating-u1.avg\_rating)^2)) as sqrtxx\_2,

SQRT(SUM((y.rating-u2.avg\_rating)^2)) as sqrtyy\_2

MERGE (u1)-[s:SIMILARITY]-(u2)

ON CREATE SET s.cosine = xy / (sqrtx2\*sqrty2),

s.pearson = CASE WHEN (sqrtxx\_2\*sqrtyy\_2)=0 THEN 0 ELSE xx\_yy\_ / (sqrtxx\_2\*sqrtyy\_2) END,

s.meansquare = MeanSquare

1. Calculate SImilarity of Users of type ‘Married’ with the both Users of type ‘Couple’ (as a sum) in “cosine”, “pearson”, “meansquare” property of Users of type ‘Married’

MATCH (c1:Users{user\_id:'12874'})-[s1:SIMILARITY]-(u1:Users{attr:'M'})

MATCH (c2:Users{user\_id:'30944'})-[s2:SIMILARITY]-(u2:Users{attr:'M'})

WHERE u1.user\_id=u2.user\_id

WITH u1, (s1.cosine+s2.cosine) AS cosine,

(s1.pearson+s2.pearson) AS pearson,

(s1.meansquare+s2.meansquare) AS meansquare

SET u1.cosine=cosine,

u1.pearson=pearson,

u1.meansquare=meansquare

1. Calculate 100 highest similar Books for the Users of type ‘Couple’ calculated by the User Similarity of Users of type ‘Married’ and
   1. put in “cosine”, “pearson”, “meansquare” property of those books and
   2. set property equal ‘true’ in “registry\_cosine”, “registry\_pearson”, “registry\_meansquare” property of those books.

MATCH (u2:Users{attr:'M'})-[r:RATED]->(b:Books)

MATCH (u1:Users {attr:'C'})

WHERE NOT((u1)-[:RATED]->(b))

WITH b, SUM(u2.cosine\*r.rating) AS Weighted\_Sim

ORDER BY Weighted\_Sim DESC

SET b.registry\_cosine = true, b.cosine = Weighted\_Sim

RETURN b.book\_id, Weighted\_Sim

LIMIT 100

MATCH (u2:Users{attr:'M'})-[r:RATED]->(b:Books)

MATCH (u1:Users {attr:'C'})

WHERE NOT((u1)-[:RATED]->(b))

WITH b, SUM(u2.pearson\*r.rating) AS Weighted\_Sim

ORDER BY Weighted\_Sim DESC

SET b.registry\_pearson = true, b.pearson = Weighted\_Sim

RETURN b.book\_id, Weighted\_Sim

LIMIT 100

MATCH (u2:Users{attr:'M'})-[r:RATED]->(b:Books)

MATCH (u1:Users {attr:'C'})

WHERE NOT((u1)-[:RATED]->(b))

WITH b, SUM(u2.meansquare\*r.rating) AS Weighted\_Sim

ORDER BY Weighted\_Sim DESC

SET b.registry\_meansquare = true, b.meansquare = Weighted\_Sim

RETURN b.book\_id, Weighted\_Sim

LIMIT 100

1. See the common books suggested by all the similarity method

MATCH (b1:Books{registry\_cosine:true})

MATCH (b2:Books{registry\_pearson:true})

MATCH (b3:Books{registry\_meansquare:true})

WHERE b1.book\_id = b2.book\_id AND b2.book\_id = b3.book\_id

WITH b1.book\_id as book\_id,

b1.cosine as cosine,

b2.pearson as pearson,

b3.meansquare as meansquare

RETURN book\_id, cosine, pearson, meansquare

### Recommend Books for Gift to the invited Friends ###

1. Create similarity of Users to Books of “registry\_cosine” on all four features in “UB\_title, UB\_authors, UB\_language, UB\_year” relationship [Users-UB…->Books (registry\_cosine)]. Set the similarity value in “cosine” property of this “UB…” relationship.

MATCH (u:Users {attr:'F'})-[uat:UA\_title]->(idft1:IDF\_title)

MATCH (b:Books {registry\_cosine:true})-[tft:TF\_title]->(idft2:IDF\_title)

WHERE idft1.word=idft2.word

WITH u, b,

SUM(tft.val\*toFloat(idft1.IDF)\*uat.val) as sim\_title

MERGE (u)-[t:UB\_title]->(b)

ON CREATE SET t.cosine = sim\_title

MATCH (u:Users {attr:'F'})-[uaa:UA\_authors]->(idfa1:IDF\_authors)

MATCH (b:Books {registry\_cosine:true})-[tfa:TF\_authors]->(idfa2:IDF\_authors)

WHERE idfa1.author=idfa2.author

WITH u, b,

SUM(tfa.val\*toFloat(idfa1.IDF)\*uaa.val) as sim\_authors

MERGE (u)-[a:UB\_authors]->(b)

ON CREATE SET a.cosine = sim\_authors

MATCH (u:Users {attr:'F'})-[ual:UA\_language]->(idfl1:IDF\_language)

MATCH (b:Books {registry\_cosine:true})-[tfl:TF\_language]->(idfl2:IDF\_language)

WHERE idfl1.language=idfl2.language

WITH u, b,

SUM(tfl.val\*toFloat(idfl1.IDF)\*ual.val) as sim\_language

MERGE (u)-[l:UB\_language]->(b)

ON CREATE SET l.cosine = sim\_language

MATCH (u:Users {attr:'F'})-[uay:UA\_year]->(idfy1:IDF\_year)

MATCH (b:Books {registry\_cosine:true})-[tfy:TF\_year]->(idfy2:IDF\_year)

WHERE  idfy1.year=idfy2.year

WITH u, b,

SUM(tfy.val\*toFloat(idfy1.IDF)\*uay.val) as sim\_year

MERGE (u)-[y:UB\_year]->(b)

ON CREATE SET y.cosine = sim\_year

1. Create similarity of Users to Books of “registry\_pearson” on all four features in “UB\_title, UB\_authors, UB\_language, UB\_year” relationship [Users-UB…->Books (registry\_pearson)]. Set the similarity value in “pearson” property of this “UB…” relationship.

MATCH (u:Users {attr:'F'})-[uat:UA\_title]->(idft1:IDF\_title)

MATCH (b:Books {registry\_pearson :true})-[tft:TF\_title]->(idft2:IDF\_title)

WHERE idft1.word=idft2.word

WITH u, b,

SUM(tft.val\*toFloat(idft1.IDF)\*uat.val) as sim\_title

MERGE (u)-[t:UB\_title]->(b)

SET t.pearson = sim\_title

MATCH (u:Users {attr:'F'})-[uaa:UA\_authors]->(idfa1:IDF\_authors)

MATCH (b:Books {registry\_pearson:true})-[tfa:TF\_authors]->(idfa2:IDF\_authors)

WHERE idfa1.author=idfa2.author

WITH u, b,

SUM(tfa.val\*toFloat(idfa1.IDF)\*uaa.val) as sim\_authors

MERGE (u)-[a:UB\_authors]->(b)

SET a.pearson = sim\_authors

MATCH (u:Users {attr:'F'})-[ual:UA\_language]->(idfl1:IDF\_language)

MATCH (b:Books {registry\_pearson:true})-[tfl:TF\_language]->(idfl2:IDF\_language)

WHERE idfl1.language=idfl2.language

WITH u, b,

SUM(tfl.val\*toFloat(idfl1.IDF)\*ual.val) as sim\_language

MERGE (u)-[l:UB\_language]->(b)

SET l.pearson = sim\_language

MATCH (u:Users {attr:'F'})-[uay:UA\_year]->(idfy1:IDF\_year)

MATCH (b:Books {registry\_pearson:true})-[tfy:TF\_year]->(idfy2:IDF\_year)

WHERE  idfy1.year=idfy2.year

WITH u, b,

SUM(tfy.val\*toFloat(idfy1.IDF)\*uay.val) as sim\_year

MERGE (u)-[y:UB\_year]->(b)

SET y.pearson = sim\_year

1. Create similarity of Users to Books in “registry\_meansquare” on all four features in “UB\_title, UB\_authors, UB\_language, UB\_year” relationship [Users-UB…->Books (registry\_meansquare)]. Set the similarity value in “meansquare” property of this “UB…” relationship.

MATCH (u:Users {attr:'F'})-[uat:UA\_title]->(idft1:IDF\_title)

MATCH (b:Books {registry\_meansquare:true})-[tft:TF\_title]->(idft2:IDF\_title)

WHERE idft1.word=idft2.word

WITH u, b,

SUM(tft.val\*toFloat(idft1.IDF)\*uat.val) as sim\_title

MERGE (u)-[t:UB\_title]->(b)

SET t.meansquare = sim\_title

MATCH (u:Users {attr:'F'})-[uaa:UA\_authors]->(idfa1:IDF\_authors)

MATCH (b:Books {registry\_meansquare:true})-[tfa:TF\_authors]->(idfa2:IDF\_authors)

WHERE idfa1.author=idfa2.author

WITH u, b,

SUM(tfa.val\*toFloat(idfa1.IDF)\*uaa.val) as sim\_authors

MERGE (u)-[a:UB\_authors]->(b)

SET a.meansquare = sim\_authors

MATCH (u:Users {attr:'F'})-[ual:UA\_language]->(idfl1:IDF\_language)

MATCH (b:Books {registry\_meansquare:true})-[tfl:TF\_language]->(idfl2:IDF\_language)

WHERE idfl1.language=idfl2.language

WITH u, b,

SUM(tfl.val\*toFloat(idfl1.IDF)\*ual.val) as sim\_language

MERGE (u)-[l:UB\_language]->(b)

SET l.meansquare = sim\_language

MATCH (u:Users {attr:'F'})-[uay:UA\_year]->(idfy1:IDF\_year)

MATCH (b:Books {registry\_meansquare:true})-[tfy:TF\_year]->(idfy2:IDF\_year)

WHERE  idfy1.year=idfy2.year

WITH u, b,

SUM(tfy.val\*toFloat(idfy1.IDF)\*uay.val) as sim\_year

MERGE (u)-[y:UB\_year]->(b)

SET y.meansquare = sim\_year

1. Show the top 10 Gift suggestion of Users of type ‘Friend’ to the Users of type ‘Couple’ for 100 Registry books found using Cosine, Pearson, & MeanSquare user similarity.  
   In the following 3 queries, you can
   1. Change the user\_id and put any user\_id of Users of type ‘Friend’.
   2. Change the weight of title/authors/language/year to calculate recommendation.

* For registry\_cosine

WITH '21228' as id,

6 as weight\_title,

3 as weight\_authors,

1 as weight\_language,

3 as weight\_year

MATCH (u:Users {user\_id:id})-[ubt:UB\_title]->(b:Books {registry\_cosine:true})

MATCH (u:Users {user\_id:id})-[uba:UB\_authors]->(b:Books {registry\_cosine:true})

MATCH (u:Users {user\_id:id})-[ubl:UB\_language]->(b:Books {registry\_cosine:true})

MATCH (u:Users {user\_id:id})-[uby:UB\_year]->(b:Books {registry\_cosine:true})

WITH b, ubt.cosine as sim\_title, uba.cosine as sim\_authors, ubl.cosine as sim\_language, uby.cosine as sim\_year, (weight\_title\*ubt.cosine + weight\_authors\*uba.cosine + weight\_language\*ubl.cosine + weight\_year\*uby.cosine) as sim\_total

ORDER BY (weight\_title\*sim\_title + weight\_authors\*sim\_authors + weight\_language\*sim\_language + weight\_year\*sim\_year) DESC

RETURN b.title, b.authors, b.language, b.year,

round(sim\_title\*10000)/10000 as sim\_title,

round(sim\_authors\*10000)/10000 as sim\_authors,

round(sim\_language\*10000)/10000 as sim\_language,

round(sim\_year\*10000)/10000 as sim\_year,

round(sim\_total\*10000)/10000 as sim\_total

LIMIT 10

* For registry\_pearson

WITH '21228' as id,

6 as weight\_title,

3 as weight\_authors,

1 as weight\_language,

3 as weight\_year

MATCH (u:Users {user\_id:id})-[ubt:UB\_title]->(b:Books {registry\_pearson:true})

MATCH (u:Users {user\_id:id})-[uba:UB\_authors]->(b:Books {registry\_pearson:true})

MATCH (u:Users {user\_id:id})-[ubl:UB\_language]->(b:Books {registry\_pearson:true})

MATCH (u:Users {user\_id:id})-[uby:UB\_year]->(b:Books {registry\_pearson:true})

WITH b, ubt.pearson as sim\_title, uba.pearson as sim\_authors, ubl.pearson as sim\_language, uby.pearson as sim\_year, (weight\_title\*ubt.pearson + weight\_authors\*uba.pearson + weight\_language\*ubl.pearson + weight\_year\*uby.pearson) as sim\_total

ORDER BY (weight\_title\*sim\_title + weight\_authors\*sim\_authors + weight\_language\*sim\_language + weight\_year\*sim\_year) DESC

RETURN b.title, b.authors, b.language, b.year,

round(sim\_title\*10000)/10000 as sim\_title,

round(sim\_authors\*10000)/10000 as sim\_authors,

round(sim\_language\*10000)/10000 as sim\_language,

round(sim\_year\*10000)/10000 as sim\_year,

round(sim\_total\*10000)/10000 as sim\_total

LIMIT 10

* For registry\_meansquare

WITH '21228' as id,

6 as weight\_title,

3 as weight\_authors,

1 as weight\_language,

3 as weight\_year

MATCH (u:Users {user\_id:id})-[ubt:UB\_title]->(b:Books {registry\_meansquare:true})

MATCH (u:Users {user\_id:id})-[uba:UB\_authors]->(b:Books {registry\_meansquare:true})

MATCH (u:Users {user\_id:id})-[ubl:UB\_language]->(b:Books {registry\_meansquare:true})

MATCH (u:Users {user\_id:id})-[uby:UB\_year]->(b:Books {registry\_meansquare:true})

WITH b, ubt.meansquare as sim\_title, uba.meansquare as sim\_authors, ubl.meansquare as sim\_language, uby.meansquare as sim\_year, (weight\_title\*ubt.meansquare + weight\_authors\*uba.meansquare + weight\_language\*ubl.meansquare + weight\_year\*uby.meansquare) as sim\_total

ORDER BY (weight\_title\*sim\_title + weight\_authors\*sim\_authors + weight\_language\*sim\_language + weight\_year\*sim\_year) DESC

RETURN b.title, b.authors, b.language, b.year,

round(sim\_title\*10000)/10000 as sim\_title,

round(sim\_authors\*10000)/10000 as sim\_authors,

round(sim\_language\*10000)/10000 as sim\_language,

round(sim\_year\*10000)/10000 as sim\_year,

round(sim\_total\*10000)/10000 as sim\_total

LIMIT 10