BUILDING A HYBRID ENGINE

Master in Big Data and Business Analytics

Recommendation Engines

PROFESSOR IVAN TARRADELLAS OLMO

Madrid, 28 September 2019

Group A

Benjamin Chumaceiro

Ignacio Mouawad

Maria Joyce

Rosalia Contreras Moreira

Eduardo Cort Pons

Recommendation Engines Quora

Benjamin Chumaceiro | Ignacio Mouawad | Maria Joyce | Rosalia Contreras Moreira | Eduardo Cort Pons | Srishti Singh

10/12/2019

Hybrid Challenge

The Assignment Question: Define your own Quora-like RecSystem. Choose a feature-weighted linear stacking, a trust-aware CF, content-based similarity or build your own. It is key in this exercise to explain in detail your solutions with good argumentations. The "best argumented" solution will have the best note. Remember that a RecSystem is not just an algorithm. i.e. A good way to show it is to use a mockup of the site, or app, that you envision, pointing out the motivations and/or algorithm behind of each UI component (To collect implicit feedback, to fill the profile explicitly, to recommend non-personalised questions to solve cold-start, etc.) Solution for the Hybrid Challenge

Introduction to Quora

Quora is an American question-and-answer website where questions are asked, answered, and edited by users of the site or mobile app. Currently, Quora has different ways to recommend questions to users:

Home feed question recommendations Quora provides "interesting" questions that are relevant to the User's preferences for topics that they have selected when they first sign up as a user to quora.

Daily Digest Quora sends a daily email containing a set of questions with one answer that is deemed the best answer, given certain ranking criteria.

Related Questions A set of questions that relates to the current question is displayed on the side. This display is not tailored to the specific user, it is based on the similarity to the current question.

Requested Answers This feature lets a user direct a question to other users whom they consider better suited to answer it.

Summary of Improvements to Quora's recommendations

To improve the recommendations given to users, we decided to introduce three new features.

Feature 1 is called Magic Buttons.

This feature will provide the user with recommendations based on the User's current daily activity. The criteria for the recommendations will be: . the time it will take to read the question and answer . the time of day

Feature 2 is called User Dynamic Topic Selection

This feature gives the user more opportunity to find new Topics that they might be interested in. Currently, quora mobile app users can only search by typing in specific topics. This new feature will give the user a List of Broad topics that they can browse through, select and then select subtopics.

Feature 3 is called Hot Topic Push

For User with an interest in some broad topics such as Sport, this feature allows these Users to receive notifications about what are considered 'Hot Topics'. These are Topics that are Hot in the sense that they are new, topical and often about an event that comes under the Parent topic of sport and can be about a SubTopic underneath the Sport Parent Topic. Examples of this would be a new Topic which is live during the two week duration of the Olympics, or long weekend of a Science and Technology Conference.

Feature 1 Magic Buttons

These buttons will appear in the app, on a left panel alongside the usual feed of questions. The options that we will present and discuss here are as follows:

Icon as seen in the App	Magic Button Nickname	Selection Criteria	Comment
	Breakfast	Q was created < 2 days ago	This provides the User with recent trending topics.
	Bathroom	Q has < 150 words	This Q will be a quick read for the user.
	Commute	Q has > 700 words	This Q is a longer read, suitable for reading while on a commute.

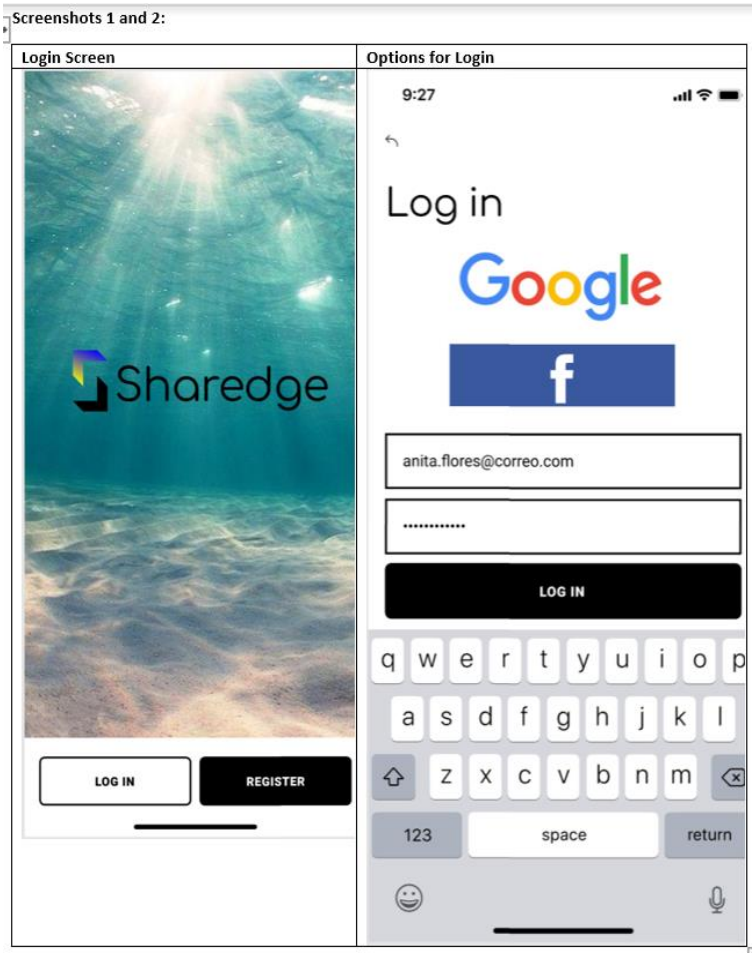
Existing Users: For existing users such as Users 1 - 3 in the dataset we have been provided with, the Bathroom and Comute Questions are chosen from the ranked recommendations for the user which have already been identified using the Content Based IDF. For the Breakfast Question, the question is chosen from the currently Trending Topics across the overall application. A Trending Topic is a topic that has had a high level of recent activity in terms of upvotes and answers.

Cold Start Users: For Cold Start users, such as User 4 in the dataset, as discussed above in the Switched Hybrid Solution, the most popular questions will be determined based on a combination of the average of the IDF predictions and the standardized answers score.

Let's walk through some mock-ups of how these Buttons will appear in the mobile app.

For this assignment, we have renamed the app as 'Sharedge'; a portmanteau of Share and Knowledge.

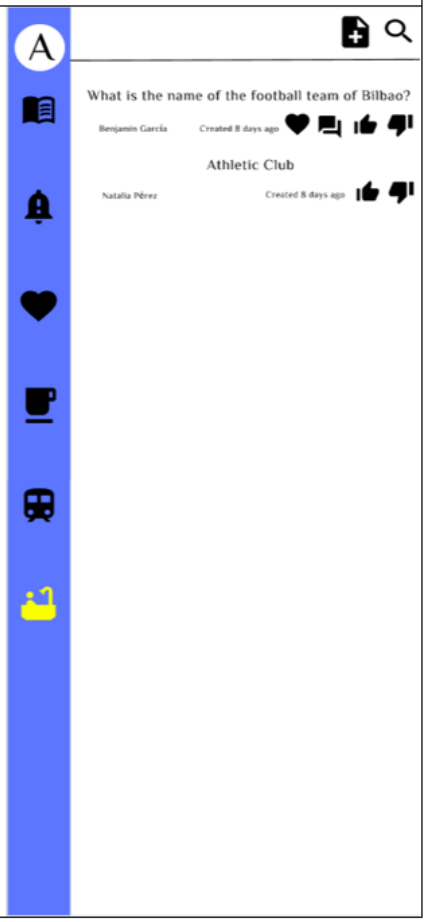
QandA_wordcount: this represents the number of words in the Question plus Answer combined.



Screenshots 3 and 4:

Asking the User to choose their favourite topics	**The landing page, displayed upon Login. Note, the Magic Buttons are visible on the left side panel.**
	

Screenshots 5 and 6:

The User has just selected the Breakfast Magic Button. A trending Question less than 2 days old is displayed. In this case the question is 22 hours old.	The User has just selected the Bathroom Magic Button. A Question with less than 150 QandA_wordcount is displayed.
	

Screenshot 7:

Below is the logic that would be behind those mock up screens.

There is information in the dataset about whether the user has upvoted, downvoted, or answered the question. If the user didn't do any of this actions, we considered the user as not having read the question. This is only assumed to create a recommendation system for current users. Note also, we created a separate table in a file called (userfeedback.csv) that contains this information.

```
read_table <- data.frame()
for(i in 1:ncol(feedback)){
  for(k in 1:nrow(feedback)){
  if(!is.na(feedback[k,i]) | !is.na(answers[k,i]) ){
    read_table[k,i] <- 1
  }
    else{
      read_table[k,i]<- 0
    }
```

```
  }
}
rownames(read_table) <- rownames(question_topics)
colnames(read_table) <- colnames(idf_predict_table)
read_table
```

```
##            User.1 User.2 User.3 User.4
## question1       1      1      0      0
## question2       1      1      1      0
## question3       0      0      0      0
## question4       0      1      0      0
## question5       0      1      1      0
## question6       1      0      0      0
## question7       0      0      1      0
## question8       0      1      1      0
## question9       0      0      0      0
## question10      0      0      0      0
## question11      0      0      0      0
## question12      0      1      1      0
## question13      0      1      1      0
## question14      0      0      0      0
## question15      0      0      1      0
## question16      1      0      1      0
## question17      0      1      1      0
## question18      0      0      0      0
## question19      1      1      1      0
## question20      0      1      1      0
```

For the sake of creating an app and adding more features, we created a random word count for each question. Moreover, we also created random dates for those questions, as an extra feature for our app. The date is used to calculate when the question was created.

```
wordcount <-
c(138,744,32,24,850,34,234,235,101,803,578,754,843,104,83,356,768,126,868
,38)
x<-Sys.Date()
dates <- c(x-455,x-1,x-1,x-322,x-1,x-157,x-230,x-100,x-455, x-1, x-1, x-
56, x-1,x,x-1, x-7,x-1,x-1,x-23, x-1)

wordcount_date <- data.frame(wordcount,dates)
colnames(wordcount_date) <- c("word count", "date posted")
rownames(wordcount_date) <- rownames(question_topics)
wordcount_date
```

```
##            word count date posted
## question1         138  2018-07-15
## question2         744  2019-10-12
## question3          32  2019-10-12
## question4          24  2018-11-25
## question5         850  2019-10-12
```

```
## question6          34  2019-05-09
## question7         234  2019-02-25
## question8         235  2019-07-05
## question9         101  2018-07-15
## question10        803  2019-10-12
## question11        578  2019-10-12
## question12        754  2019-08-18
## question13        843  2019-10-12
## question14        104  2019-10-13
## question15         83  2019-10-12
## question16        356  2019-10-06
## question17        768  2019-10-12
## question18        126  2019-10-12
## question19        868  2019-09-20
## question20         38  2019-10-12
```

In our app, we created a new Magic Button for bathroom reads. This Magic Button recommends questions for users where their predicted score is bigger than 0, the wordcount of each question is less than 150 (quick read) and to make sure that the user hasn't seen it before (based on previous assumption).

```
bathroom_pred <- vector()
bathroom_pred_list <- data.frame("dummy" = c(1:20))
predict_table_all_users <- idf_predict_table
predict_table_all_users$User.4<- idf_average

for(i in 1:ncol(predict_table_all_users)){
  for(k in 1:nrow(read_table)){
    if(read_table[k,i]==0 & wordcount[k]<150 &
predict_table_all_users[k,i]>0){
      bathroom_pred[k]=predict_table_all_users[k,i]
    }
    else{bathroom_pred[k]=0}
  }
  bathroom_pred_list <- cbind(bathroom_pred_list,bathroom_pred)

}
bathroom_pred_list$dummy <- NULL
colnames(bathroom_pred_list)<- colnames(predict_table_all_users)
rownames(bathroom_pred_list)<-rownames(question_topics)
bathroom_pred_list[!(apply(bathroom_pred_list,1,sum)==0),]

##                User.1    User.2    User.3     User.4
## question3  0.23502693 0.0000000 0.0000000 0.00000000
## question4  0.00000000 0.0000000 0.0000000 0.02153709
## question6  0.00000000 0.0000000 0.0000000 0.02715675
## question9  0.36075074 0.0000000 0.0000000 0.00000000
## question14 0.00000000 0.4265722 0.1533189 0.17481149
## question15 0.07215623 0.1852638 0.0000000 0.00000000
```

```
## question18 0.18160090 0.1889433 0.0000000 0.12351472
## question20 0.02986545 0.0000000 0.0000000 0.06795358
```

As well as the bathroom Magic Button, we created a commute Magic Button which recommends questions that usually take more time to read (> 700 words) and therefore makes the commute more entertaining.

```
commute_pred <- vector()
commute_pred_list <- data.frame("dummy" = c(1:20))
predict_table_all_users <- idf_predict_table
predict_table_all_users$User.4<- idf_average


for(i in 1:ncol(predict_table_all_users)){
  for(k in 1:nrow(read_table)){
    if(read_table[k,i]==0 & wordcount[k]>700 &
predict_table_all_users[k,i]>0){
      commute_pred[k]=predict_table_all_users[k,i]
    }
    else{commute_pred[k]=0}
  }
  commute_pred_list <- cbind(commute_pred_list,commute_pred)

}
commute_pred_list$dummy <- NULL
colnames(commute_pred_list)<- colnames(predict_table_all_users)
rownames(commute_pred_list)<-rownames(question_topics)
commute_pred_list[!(apply(commute_pred_list,1,sum)==0),]

##               User.1     User.2     User.3     User.4
## question2  0.0000000 0.00000000 0.00000000 0.12839500
## question5  0.0000000 0.00000000 0.00000000 0.07133580
## question10 0.0000000 0.09417315 0.01583125 0.00000000
## question12 0.6220964 0.00000000 0.00000000 0.00000000
## question13 0.0000000 0.00000000 0.00000000 0.05542998
## question17 0.0000000 0.00000000 0.00000000 0.12839500
```

and as for our last feature, we created a breakfast feature that includes trending topics from the last 48 hours, taking into considerstion the date column previously created.

```
breakfast_pred <- vector()
breakfast_pred_list <- data.frame("dummy" = c(1:20))
predict_table_all_users <- idf_predict_table
predict_table_all_users$User.4<- idf_average


for(i in 1:ncol(predict_table_all_users)){
  for(k in 1:nrow(read_table)){
    if(read_table[k,i]==0 & dates[k]>(Sys.Date()-2) &
predict_table_all_users[k,i]>0){
```

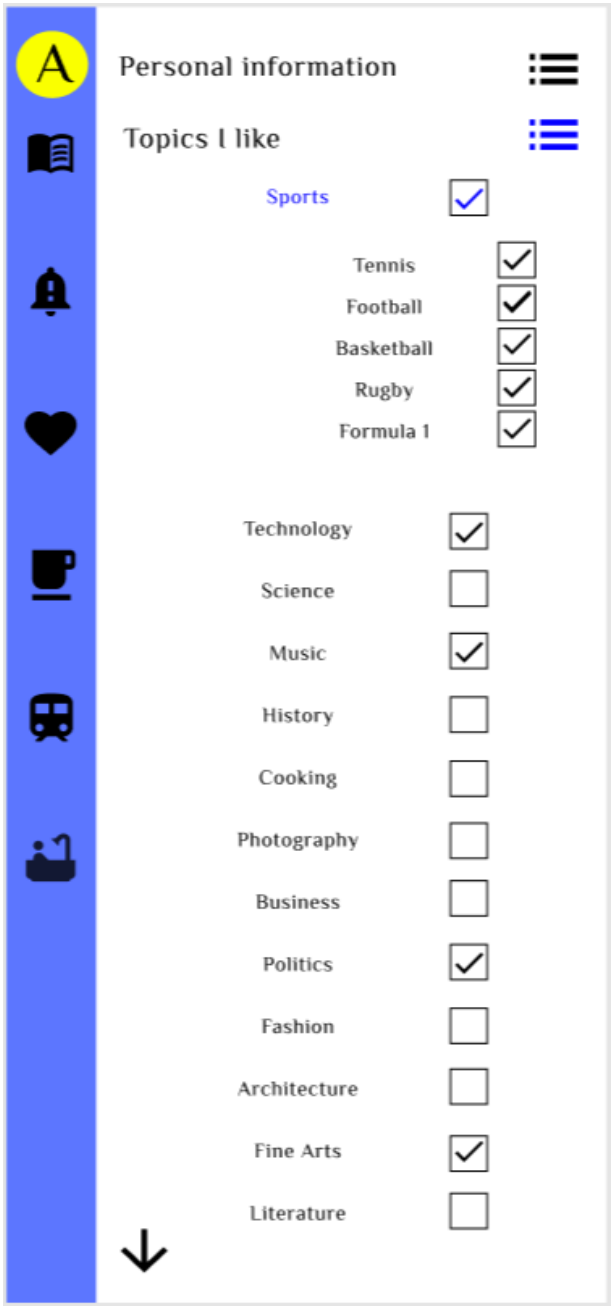
```
      breakfast_pred[k]=predict_table_all_users[k,i]
    }
    else{breakfast_pred[k]=0}
  }
  breakfast_pred_list <- cbind(breakfast_pred_list,breakfast_pred)

}
breakfast_pred_list$dummy <- NULL
colnames(breakfast_pred_list)<- colnames(predict_table_all_users)
rownames(breakfast_pred_list)<-rownames(question_topics)
breakfast_pred_list[!(apply(breakfast_pred_list,1,sum)==0),]
```

```
##                User.1     User.2     User.3     User.4
## question2  0.00000000 0.00000000 0.00000000 0.12839500
## question3  0.23502693 0.00000000 0.00000000 0.00000000
## question5  0.00000000 0.00000000 0.00000000 0.07133580
## question10 0.00000000 0.09417315 0.01583125 0.00000000
## question11 0.05245020 0.04831938 0.05338959 0.05138639
## question13 0.00000000 0.00000000 0.00000000 0.05542998
## question14 0.00000000 0.42657220 0.15331891 0.17481149
## question15 0.07215623 0.18526376 0.00000000 0.00000000
## question17 0.00000000 0.00000000 0.00000000 0.12839500
## question18 0.18160090 0.18894326 0.00000000 0.12351472
## question20 0.02986545 0.00000000 0.00000000 0.06795358
```

Feature 2: User Dynamic Topic Selection

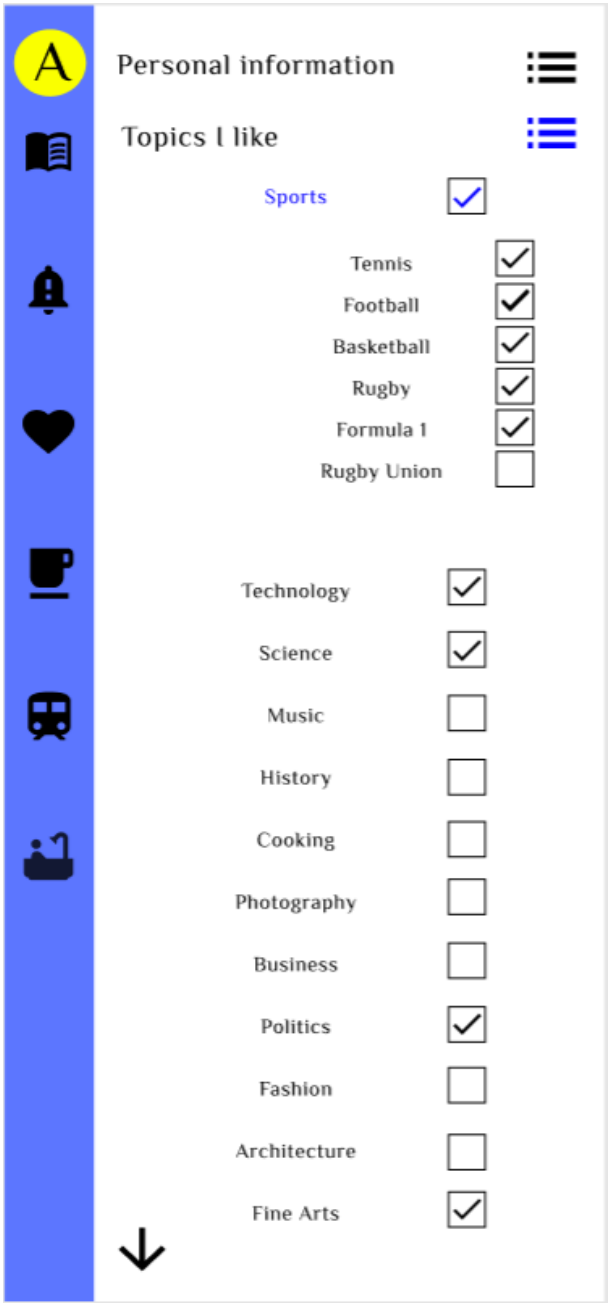
The screenshot below shows the Dynamic Topic Selection Option available to the User. In this screenshot you can see the the Main Topics such as Sports, Technology and Science. Under sports you can see some SubTopics such as Tennis, Football, etc.



Feature 3: Hot Topic Push

The screenshot below shows the notification that the User will see. The notification will be sent by the app because the User already follows a Sports Topic, and it is likely that they may also be interested in following a Trending Topic such as an international Sports tournament.

If the User responds Yes to Following the new Hot Topic, then the following Screen will be displayed:


Personal information
Topics I like
Sports
Tennis
Football
Basketball
Rugby
Formula 1
Rugby Union
Technology
Science
Music
History
Cooking
Photography
Business
Politics
Fashion
Architecture
Fine Arts