CS 4125 Seminar in Research Methodology for Data Science Coursework B

Dimitropoulos Georgios: 4727657
Manousogiannis Emmanouil: 4727517
Mastoropoulou Emmeleia: 4743539
Group 13

March 20, 2018

1 Evaluation

A)

In order to be able to find the best three systems for the search engine, we analyzed the results of the study. Each search engine consists of three main components. Namely, it has a model, a tokenizer and a Lexical Unit Generator (LUG). Now, for each of these components we have different options. More specifically, there are 6 different choices for the tokenizer, 6 different options for the LUG and 17 different choices for the model. Hence we have 612(6*6*17) different systems to choose from. Each one of these 612 different topics is evaluated with 200 topics. Finally, given a results list from a system in response to a topic, we have 5 different metrics to assess how good our system is. Hence, we ran all 612 systems with the 50 topics found in each of the 4 datasets, and scored them according to the 5 evaluation metrics. So, we have finally 612000 scores (our dataset) which is a value between 0 and 1. Based on the description, the mean score over topics is the final criterion which we use to compare systems and make a choice. The method which we propose in order to find the best three systems for the search engine is as follows:

Step1) For each of the 5 metrics, we calculated the average score of each system in the different topics.
Step2) For each metric, we ranked the systems based on the previously calculated average score .
Step3) Finally, we added the rankings of the systems in the 5 different metrics, and select the top 3 systems based on their overall ranking.
The results of our procedure can be found in the table 1.

Token	Lug	Model	NDCG	P10	RBP	ERR	AP
Terrier	SnowballPorter	DPH	0.5678	0.4430	0.4710	0.4627	0.2408
Terrier	Krovetz	JSKLS	0.5654	0.4320	0.4661	0.4690	0.2423
Terrier	Porter	DPH	0.5687	0.4425	0.4704	0.4607	0.2410

Table 1: Best three systems

However, due to the nature of this procedure and based on the fact that there is a no a generic system that outperforms every other, there is no a general rule to tell us that we indeed we have three "best" systems.

B)

In order to improve the search engine we need to recommend a component to change. First of all we create the following linear models, in order to predict the impact of the predictor variables, to the score of our system:
1)Model 0, the null predictor
2)Model 1, the Lexical Unit Generator
3)Model 2 has as predictor the Tokenizer

4)Model 3 has as predictor the Retrieval model

After creating those models, we compared them using anova function with test F, in order to determine which one has a significant impact on the dependent variable 'score'. We can observe that p-value is the same for every model and it is lower than 0.05. Hence, we check the F value for every model,compared to null, searching for the greatest distance from 1. Based on that, the Tokenizer has a significant impact on the prediction. Thus, in order to improve the search engine the tokenizer component that we should try to change.

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
Model 0	611999	46402				
Model 1	611994	46349	5	52.836	139.53	< 2.2e-16

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
Model 0	611999	46402				
Model 2	611994	45863	5	539.47	1439.7	2.2e-16

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
Model 0	611999	46402				
Model 3	611983	45906	16	496.33	413.54	2.2e-16

2 Critical Analysis

After looking at the overview of the design and the implementation of the whole procedure, we are now able to present the main design drawbacks and limitations as well as some measurements and sampling issues.For each one of them, we propose some possible improvements and solutions
The first issue we can notice is that the TREC series of conferences that provided the evaluation procedure was done back in 1992. This means it is almost 25 years ago, so probably a more up to date approach is required.

Furthermore, the datasets in which the search engines have access to, TIPSTER and WTlog, are also really old and the documents websites or files that they will return will have significant differences from the ones that would be available nowadays. A solution would be a more up-to date database which will containt representative sample documents, websites,multimedia and files regarding the company's domain.This means that the new datasets will have to be relevant to the available databases of the company and its' needs.

In addition, the fact that NSA analysts were selected to choose the search topics in which the search engines will be evaluated. However, an average NSA alanyst does dot represent an average employee of the company who will use the company's search engine, so this choice may intoduce bias to our results.An easier and more accurate approach, would be to assign this task to some experienced employees of the company, from different departments in order to ensure diversity.

Another part that we should mention, is that from the whole number of candidate topics that the analysts propose, only a subset of 50 are selected as a sample, ensuring a diversity in topics and difficulty level. This procedure is not clearly explained , so someone may wonder how the difficulty of each topic is determined.

Another basic limitaion of the evaluation process, is that only the first 100 documents that a search engine returns after a query are evaluated, and the rest are considered irrelevant. This can be considered as a clear bias.It may seem logical that most users would never even turn to the second page of the search query results, however this may not be the case for an employee of this specific company. It clearly depends on the task.If an employee tries to retrieve as much data as possible regarding a query, then it is obvious that we should not ignore the rest of the documents.

Finally, the metrics used to assign a score to our systems, do not always consider the same scale for relevance and irrelevance judgement, as some consider a binary relevance value and some consider a graded relevance value.This makes it extremely difficult to evaluate different systems that have different performance for each metric.So a more scale insensitive scale system could be a better solution for our case.
To conclude, we could say that updating the datasets used , selecting a variety of examined topics that employees would propose and considering a larger percentage of the retrieved documents of each search engine for each evaluation, as well as updating the metrics accordingly would be some clear, and easy to implement, improvements to our design and selection of the appropriate search engine.

[[Appendix A-Source Code]

```
    setwd("C:\\Users\\pc1\\Desktop\\Seminar\\Assignment2")
library("dplyr")
d <- read.csv("data.csv")


ndcg<-subset(d,metric=="ndcg")
ap<-subset(d,metric=="ap")
err<-subset(d,metric=="err")
p10<-subset(d,metric=="p10")
rbp<-subset(d,metric=="rbp")


ndcgScores<-ndcg%>%group_by(token,lug  ,model)%>% summarize(mean=(mean(score)))
apScores<-ap%>%group_by(token,lug  ,model)%>% summarize(mean=(mean(score)))
errScores<-err%>%group_by(token,lug  ,model)%>% summarize(mean=(mean(score)))
p10Scores<-p10%>%group_by(token,lug  ,model)%>% summarize(mean=(mean(score)))
rbpScores<-rbp%>%group_by(token,lug  ,model)%>% summarize(mean=(mean(score)))

ndcgScores$rank <- NA
ndcgScores$rank[order(ndcgScores$mean)] <- 1:nrow(ndcgScores)

apScores$rank <- NA
apScores$rank[order(apScores$mean)] <- 1:nrow(apScores)

errScores$rank <- NA
errScores$rank[order(errScores$mean)] <- 1:nrow(errScores)

p10Scores$rank <- NA
p10Scores$rank[order(p10Scores$mean)] <- 1:nrow(p10Scores)


rbpScores$rank <- NA
rbpScores$rank[order(rbpScores$mean)] <- 1:nrow(rbpScores)

final<-data.frame(ndcgScores$token,ndcgScores$lug,ndcgScores$model,ndcgScores$rank,ap
final$totalScore <- rowSums(final[4:8])





install.packages("dplyr")
#########################################1b##################################
require(Matrix)
library(lme4)
library(MASS)
library(foreign)
library(car)
library(ggplot2)
library(nlme)
library(reshape)
library(graphics)
require(lattice)
library(lattice)
require(Matrix)
library(lme4)
```

```
model0<-lm(d$score~1)
model1 <- lm(d$score ~ lug, data=d)
model2<-lm(d$score ~ token,data=d)
model3<-lm(d$score ~ model,data=d)

anova(model0,model1,test = "F")

anova(model0,model2,test = "F")

anova(model0,model3,test = "F")
```