Prototype 1

Team 5
Peihao, Nithin, Sangeeta

1. K-means (Peihao)

*Hyperlinks are set up in entities here, feel free to check them by clicking

Introduction

For these days I'm working on using k-means clustering to make classification for paragraph corpus. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as prototype as the cluster.

Difficulties

The whole paragraph corpus is so large so to do clustering for it takes indecent long time. What's more, the centroids of each cluster are large dimension vectors. Computing similarity between query/paragraph vector and centroid vector is expensive. So, there is no way to process them on searching time. My approach to do it is, read paragraph corpus offline (createVector.py)to get paragraph vectors and a dimension term vector (as a notation for each value in paragraph vectors) and store them in a xml format file for future use.

Usage

K-means clustering gives us a classification for paragraphs. Each classification has a centroid vector. By computing similarity between centroid vector and query vector, we will have a rank for clusters. For un-prediction usage, we give a weight to the rank of paragraph's cluster and combine this rank with paragraph rank to come up with a new rank. For prediction usage, we set paragraphs to clusters by using the similarity between paragraph vector and centroid vector. Then do the same thing as un-prediction usage.

Implementation & Results

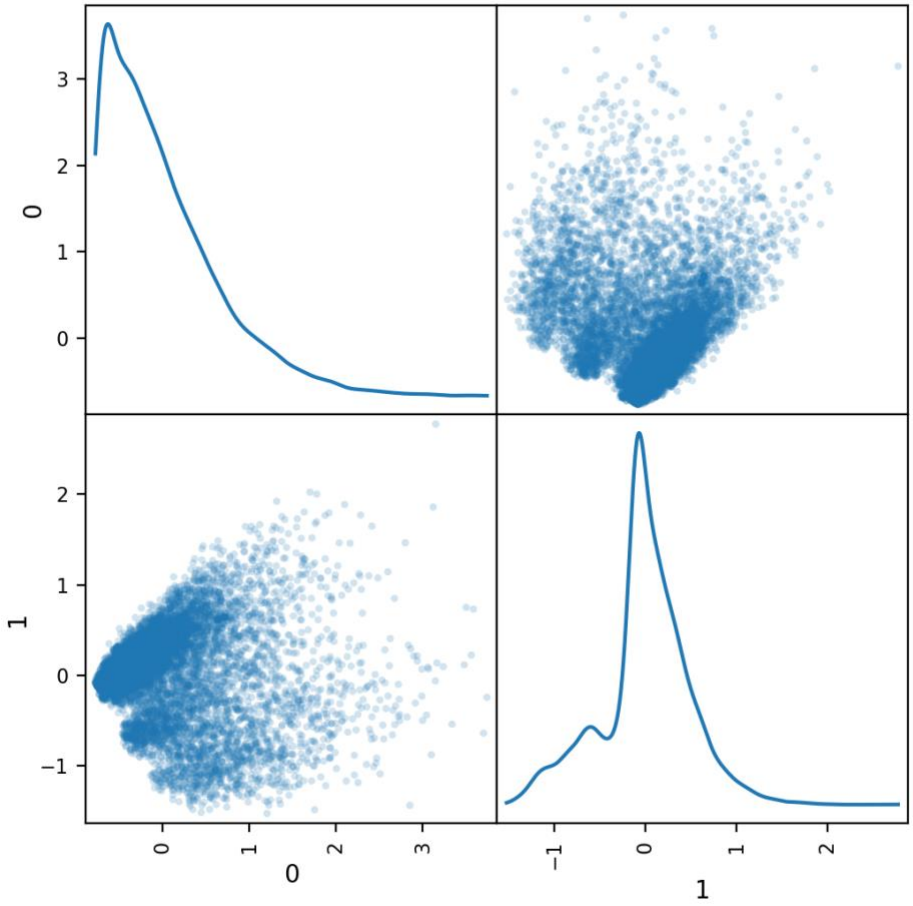
The dataset here is 1000 paragraphs from corpus. After calculation, the dimension for this dataset is 56341.

I came up two approaches to use k-means clustering on our task. First is to do re-rank once we get the ranking by using clustering during query time and then come up with the runfile for evaluation. Second is to get the runfile first. And then run reRanker.py on runfile to re-rank by clustering and generate a new runfile for evaluation. The first approach will be done by pageQuery.java. The second will be done by runRanker.py.

I'm working on them in the same time since I want to see which one has a less cost and it is the most important thing we care about.

The reason that I want to try to use python script to re-rank is, there is a lot of vector computations during re-rank time and the numpy tool in python is developed by C language so it has a good performance on vector computations.

I'm using 3 as the value for k in k-means. Here's the plot for paragraph vectors. Since each vector is a 56431 dimensions vector so here I used a method called Principle Component Analysis (PCA) to reduce the dimension to 2.



We can image that there are three clusters from the plot, which make k = 3 reasonable.

To process k-means clustering, I'm using sklearn tool in python.

To calculate the similarity, I'm using numpy in python. The result is showed in the following pic

```
---
query:  Credit rationing
0.00031787737964
0.00075507376991
-5.21473990769e-18
---
---
query:  Natural growth promoter
0.00561933126823
0.00507221893849
0.000723002948156
---
---
query:  Noise-induced hearing loss
0.0053945580175
0.007474848962
0.00747103046427
---
```

The first line is the query string and the following three lines give you the similarity between the query and each centroid.

Works in process

Once I figure out which method gives a better performance on vector computation by using the sample corpus (10000 paragraphs). I'll use it for the complete corpus. As we can predict, it will take a long time to get the offline work for the complete corpus. But once I get the offline works done, we'll see is k-means helpful or not.

What's more, I'll try to use only entities to reduce the size of corpus and see what can I get. I'll also use the rank of entities which is generated by Sangeeta by using DBpedia and see is it helpful or not.

2. Classifier (Nithin)

Files

Paths in the server:
Index path - /home/ns1077/ParagraphIndex
Runfiles - /home/ns1077/Runfile
Paragraphcorpus - /home/ns1077/paragraphCorpus

Implementation

1.Lucene Index in the server.

2.Heading weight variation for pages, interior headings, lowest heading using BM25.
3.Used learning to rank "RandomForest" approach to get the best ranking for the heading weights.
4.Has done classifier which classifies passages to headings, that will be pushed to prototype 2.
5. I have included my jar file which computes Heading weights
6. It is in the repository..
6.Download the file CS980.01-0.0.1-SNAPSHOT-jar-with-dependencies.jar.zip from the repository.
7. Run the jar using the command
8. Java -jar CS980.01-0.0.1-SNAPSHOT-jar-with-dependencies [pages.cbor] [index-path] [output folder]
9. Make sure you create the output folder in the path which you specify before you run the script.
10. You will see the 3 run files in the folder /home/ns1077/Runfile.

Results

The MAP results on train data are

Pages - 0.0821
Sections - 0.1039
Low Section - 0.0585

3. DBpedia (Sangeeta)

Implemention

DBpedia Entity linking, grouping and page ranking.
It contains three major modules:

1) **Entity Identification and linking:** This module identifies the entity (Key word) from the paragraph corpus. In the second part of this module, it links the entity (category to which the keyword belong) to the keyword.
2) **Grouping and structuring of the entities:** Each keyword having similar category were then grouped under that category.
   For example: Country <USA, India, China, Mexico>
   Here country is the category entity and USA, India, China, Mexico are the keyword entity.
   Note: Both keyword and category in this reference means DBpedia page of it.
3) **Page Ranking:** Each keyword and category refers to its DBpedia page. The keyword page gives
   reference of the category page. The concept is similar to a page giving citation of the other page.
   The score is proportional to the number of reference.
   For the above mentioned case : Country : 4
   The rank is decided in accordance to the descending order of scores.

Results

The rank was then compared with the top 100 retrieved from online search.

Entities	Score	Rank
DBpedia:Agent	393	1
DBpedia:Place	378	2
Schema:Place	378	2
DBpedia:PopulatedPlace	344	3
DBpedia:Person	221	4
Schema:Person	221	4
DBpedia:Person	221	4
DBpedia:Settlement	192	5
Schema:Organization	172	6
DBpedia:Organisation	172	6
DBpedia:Country	93	7
Schema:Country	93	7
DBpedia:City	81	8
Schema:City	81	8
Schema:CreativeWork	77	9
DBpedia:Work	77	9
Schema:Event	60	10
DBpedia:Event	60	10
DBpedia:OfficeHolder	59	11
Schema:AdministrativeArea	53	12
DBpedia:AdministrativeRegion	53	12
DBpedia:Region	53	12
DBpedia:Royalty	49	13
DBpedia:BritishRoyalty	49	13
DBpedia:MilitaryConflict	48	14
DBpedia:MilitaryUnit	44	15
DBpedia:Company	40	16
DBpedia:FictionalCharacter	34	17
DBpedia:Software	31	18
DBpedia:Artist	30	19
Schema:MusicGroup	30	19
DBpedia:Language	27	20
Schema:Language	27	20
DBpedia:ArchitecturalStructure	24	21
DBpedia:VideoGame	23	22
DBpedia:EducationalInstitution	20	23
Schema:EducationalOrganization	20	23
DBpedia:University	20	23

Schema:CollegeOrUniversity	20	23
DBpedia:ComicsCharacter	19	24
DBpedia:MusicalArtist	17	25
DBpedia:Film	16	26
Schema:Movie	16	26
DBpedia:Infrastructure	16	26
DBpedia:Disease	16	26
Schema:SportsTeam	16	26
DBpedia:SportsTeam	16	26
DBpedia:Species	15	27
DBpedia:SoccerClub	15	27
DBpedia:Eukaryote	15	27
DBpedia:Activity	14	28
DBpedia:Town	14	28
DBpedia:Sport	14	28
DBpedia:Band	13	29
DBpedia:EthnicGroup	12	30
DBpedia:AnatomicalStructure	11	31
DBpedia:Cleric	11	31
DBpedia:Athlete	10	32
DBpedia:GovernmentAgency	10	32
Schema:GovernmentOrganization	10	32
DBpedia:WrittenWork	10	32
DBpedia:RouteOfTransportation	9	33
DBpedia:Plant	9	33
DBpedia:Road	9	33
DBpedia:ChemicalCompound	8	34
DBpedia:ChemicalSubstance	8	34
DBpedia:TelevisionStation	7	35
Schema:WebPage	7	35
DBpedia:Building	7	35
Schema:TelevisionStation	7	35
DBpedia:Website	7	35
DBpedia:Broadcaster	7	35
DBpedia:Animal	6	36
DBpedia:TelevisionShow	6	36
DBpedia:PeriodicalLiterature	6	36
DBpedia:ChristianBishop	6	36
DBpedia:Writer	6	36
DBpedia:SportsLeague	6	36
DBpedia:RecordLabel	6	36
Schema:Product	6	36
DBpedia:MeanOfTransportation	6	36

DBpedia:SportsEvent	6	36
Schema:SportsEvent	6	36
DBpedia:Olympics	6	36
DBpedia:Saint	5	36
DBpedia:Criminal	5	36
DBpedia:Genre	5	36
DBpedia:Colour	5	36
DBpedia:MusicGenre	5	36
DBpedia:TopicalConcept	5	36
DBpedia:Election	5	36
DBpedia:NaturalPlace	5	36
DBpedia:MilitaryPerson	5	36
DBpedia:Airport	5	36
Schema:Airport	5	36
DBpedia:Island	5	36
DBpedia:InformationAppliance	4	37
DBpedia:Single	4	37
Schema:Book	4	37
DBpedia:SoapCharacter	4	37
DBpedia:Currency	4	37
DBpedia:IceHockeyLeague	4	37
DBpedia:Book	4	37
DBpedia:MusicalWork	4	37
DBpedia:Aircraft	4	37
DBpedia:Device	4	37
DBpedia:Legislature	4	37
Http://purl.org/ontology/bibo/Book	4	37
DBpedia:Holiday	4	37
DBpedia:PoliticalParty	4	37
Schema:BodyOfWater	3	38
DBpedia:HistoricPlace	3	38
DBpedia:RacingDriver	3	38
Schema:LandmarksOrHistoricalBuildings	3	38
DBpedia:BodyOfWater	3	38
Schema:RiverBodyOfWater	3	38
DBpedia:River	3	38
DBpedia:Stream	3	38
DBpedia:CelestialBody	3	38
DBpedia:SoccerPlayer	3	38
DBpedia:Year	2	39
DBpedia:Ship	2	39
DBpedia:Grape	2	39
DBpedia:MilitaryStructure	2	39

DBpedia:TradeUnion	2	39
Schema:Mountain	2	39
DBpedia:Newspaper	2	39
DBpedia:AcademicJournal	2	39
DBpedia:Philosopher	2	39
DBpedia:RaceTrack	2	39
DBpedia:Insect	2	39
DBpedia:Village	2	39
DBpedia:Mountain	2	39
DBpedia:Comedian	2	39
DBpedia:Murderer	2	39
Schema:TVEpisode	2	39
DBpedia:Award	2	39
DBpedia:ComicsCreator	2	39
DBpedia:Mammal	2	39
DBpedia:TelevisionEpisode	2	39
Schema:Museum	2	39
DBpedia:Racecourse	2	39
DBpedia:Galaxy	2	39
DBpedia:TennisPlayer	2	39
DBpedia:Magazine	2	39
DBpedia:Museum	2	39
DBpedia:SportFacility	2	39
DBpedia:TimePeriod	2	39
DBpedia:FloweringPlant	2	39
DBpedia:Publisher	1	40
Schema:Festival	1	40
DBpedia:Station	1	40
DBpedia:Fish	1	40
DBpedia:Food	1	40
DBpedia:Airline	1	40
DBpedia:Planet	1	40
DBpedia:Politician	1	40
Schema:Park	1	40
DBpedia:President	1	40
Schema:Continent	1	40
DBpedia:GolfLeague	1	40
DBpedia:Reptile	1	40
DBpedia:PowerStation	1	40
DBpedia:SoccerManager	1	40
DBpedia:RugbyClub	1	40
DBpedia:SoccerLeague	1	40
DBpedia:Swimmer	1	40

DBpedia:Cricketer	1	40
DBpedia:MusicFestival	1	40
DBpedia:Musical	1	40
DBpedia:Park	1	40
DBpedia:SportsManager	1	40
DBpedia:FormulaOneRacer	1	40
DBpedia:FashionDesigner	1	40
DBpedia:Continent	1	40

This is the result of the category scoring of the search file obtained online.

Entities	**Score**	**Rank**
DBpedia:Disease	3	1
Schema:Place	2	2
DBpedia:ChemicalCompound	2	2
DBpedia:Food	2	2
DBpedia:ChemicalSubstance	2	2
DBpedia:Place	2	2
DBpedia:Settlement	1	3
DBpedia:PopulatedPlace	1	3

The methods were compared and was found that both the categories have similarity in ranking.

The offline and online ranking matches.

Note: This method was implemented for a portion of paragraph corpus.