TOPICAL CORRELATED PERSISTENCE GRAPH

S.K.Risheek Rakshit

Objective:

With the evolutions being rapid in today's world, it is essential that we keep a track of the evolutions in order to study patterns and assimilate these outcomes. The very purpose of this project is to understand and develop a model that visualizes how the topics have evolved based on the citation network provided in the dataset and how they've met their end over time.

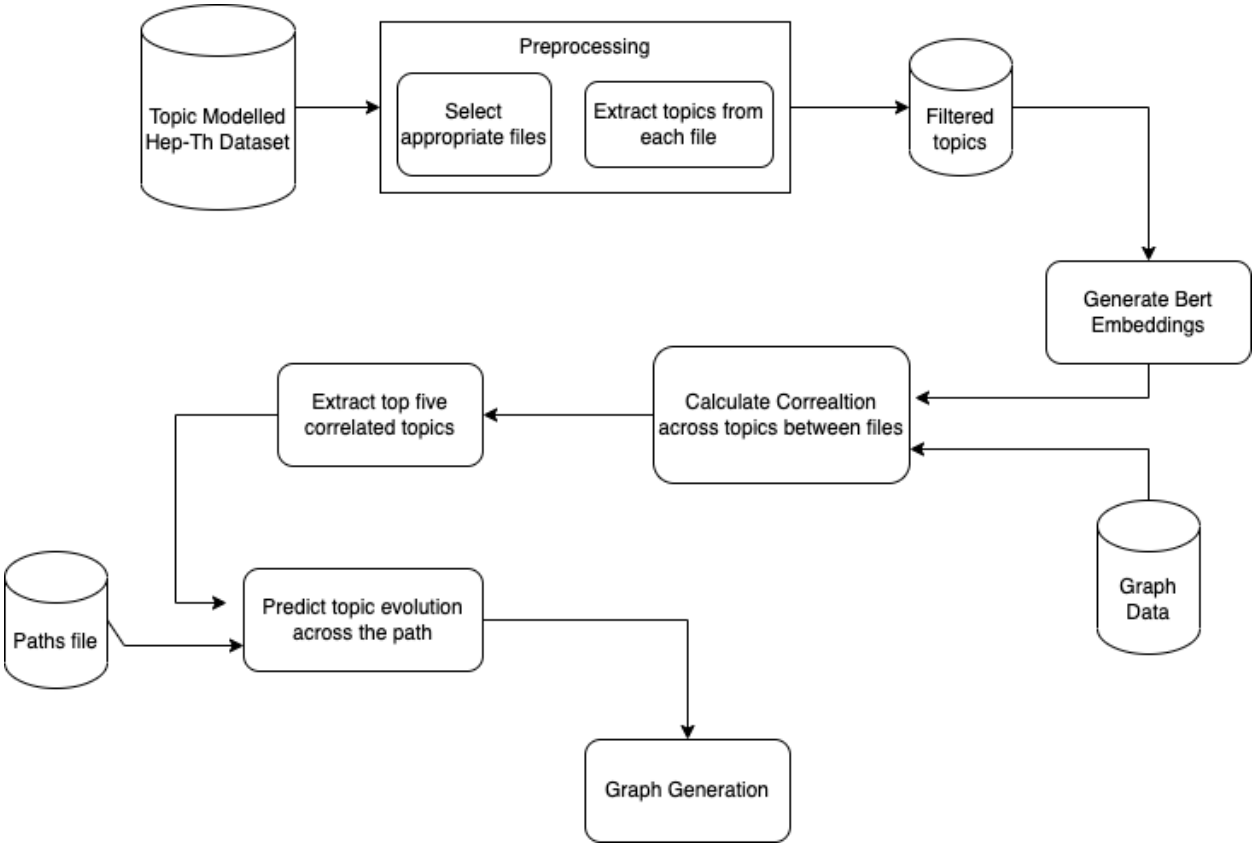
Dataset:

The dataset we worked upon was the HEP-Th(High Energy Physics - Theory) dataset modeled on Double Latent Dirichlet Allocation. If a topic *i* is cited in a paper with topic *j*, the graph contains an undirected edge from *i* to *j*. If the paper contains *k* citations this generates a completely connected (sub)graph on *k* nodes.

topic_id	weight	Text
0	0.2858055177	(0, '0.043*casimir + 0.029*force + 0.020*two + 0.019*attractive + 0.016*objects + 0.016*distance + 0.014*body + 0.014*sign + 0.012*energy + 0.011*always')
1	0.3099532175	(1, '0.028*condition + 0.020*leave + 0.019*wick + 0.019*derivative + 0.018*originate + 0.018*existence + 0.017*based + 0.017*come + 0.017*shape + 0.014*rigorous')
2	0.2962112523	(2, '0.026*sally + 0.023*problem + 0.020*tends + 0.020*conjectured + 0.016*value + 0.015*eq + 0.015*cos + 0.015*following + 0.014*conclusion + 0.013*kde')
3	0.2953314893	(3, '0.020*not + 0.019*energy + 0.018*eq + 0.017*derivative + 0.017*unlike + 0.017*hebieuc + 0.016*correction + 0.015*finiteness + 0.015*non + 0.015*addressed')
4	0.3070439607	(4, '0.026*due + 0.024*dierent + 0.021*hyperbolic + 0.020*univer + 0.018*innity + 0.018*least + 0.017*cos + 0.016*increasing + 0.015*change + 0.015*not')
5	0.2843111737	(5, '0.021*formal + 0.020*argument + 0.019*general + 0.017*repulsion + 0.016*files + 0.016*always + 0.016*univer + 0.015*shell + 0.014*show + 0.014*issue')
6	0.3258815318	(6, '0.026*quite + 0.025*assertion + 0.022*changed + 0.020*time + 0.020*bulk + 0.019*sign + 0.018*general + 0.017*important + 0.016*increasing + 0.016*issue')
7	0.3118372852	(7, '0.023*transforming + 0.021*distances + 0.018*magneticly + 0.018*monotonically + 0.018*kde + 0.017*corrections + 0.017*therefore + 0.017*sally + 0.017*actually + 0.017*small')
8	0.3055769594	(8, '0.021*conclusion + 0.021*leave + 0.020*boundary + 0.019*show + 0.017*attractive + 0.017*tends + 0.016*known + 0.015*innity + 0.015*yield + 0.014*belief')
9	0.3153663882	(9, '0.031*positivity + 0.025*innity + 0.022*puz + 0.021*parts + 0.020*small + 0.017*namely + 0.016*cosine + 0.015*show + 0.014*dierent + 0.014*forces')
10	0.3216414598	(10, '0.026*compared + 0.025*spite + 0.023*typically + 0.019*properties + 0.019*two + 0.019*type + 0.018*not + 0.016*similarly + 0.016*hieuc + 0.015*positivity')
11	0.2594405757	(11, '0.019*nature + 0.017*considerable + 0.016*polder + 0.016*relative + 0.015*nd + 0.015*repulsion + 0.014*least + 0.013*increasing + 0.012*even + 0.012*ing')
12	0.2820837732	(12, '0.020*complete + 0.020*real + 0.018*natural + 0.017*rather + 0.017*belief + 0.016*hbbieuc + 0.015*existence + 0.015*eq + 0.015*properties + 0.015*radia')
13	0.2928677035	(13, '0.027*geometries + 0.023*cosine + 0.022*originate + 0.017*wick + 0.016*subtracted + 0.016*bulk + 0.015*objects + 0.015*sphere + 0.014*vacuum + 0.014*indeed')
14	0.2926334524	(14, '0.022*given + 0.018*interest + 0.017*change + 0.016*counter + 0.016*originate + 0.016*project + 0.015*finiteness + 0.015*comment + 0.015*condition + 0.015*polarizable')
15	0.2877376655	(15, '0.029*tion + 0.018*conducting + 0.017*correction + 0.017*result + 0.016*hbbieuc + 0.015*boundary + 0.015*wick + 0.015*files + 0.014*kde + 0.014*hi')
16	0.2984414588	(16, '0.021*paradoxes + 0.021*discussion + 0.020*transforming + 0.018*compared + 0.017*correction + 0.016*conjectured + 0.016*properties + 0.016*since + 0.015*issue + 0.014*hieuc')
17	0.3103424504	(17, '0.022*files + 0.020*rst + 0.019*specically + 0.019*geometries + 0.019*forces + 0.019*remarked + 0.016*quite + 0.016*trigonometric + 0.016*magnet + 0.015*say')
18	0.3175767382	(18, '0.021*type + 0.021*even + 0.021*identify + 0.020*ka + 0.020*heeieuc + 0.019*namely + 0.018*order + 0.016*magnet + 0.016*non + 0.015*clearly')
19	0.3025179684	(19, '0.029*keywords + 0.024*conclusion + 0.020*energy + 0.020*width + 0.017*vanishing + 0.017*mutual + 0.017*two + 0.017*mainly + 0.014*given + 0.013*would')

	A	B	C
1	9204040	9304045	0.08862522545
2	9204040	9401139	0.11088564
3	9204040	9405002	0.1246353905
4	9204040	9703156	0.09022430015
5	9204040	9707203	0.03662195308
6	9204040	204145	0.04371121059
7	9204040	9505123	0.06212134891
8	9204040	9502144	0.08369767729
9	9204040	9306074	0.1370872402
10	9204040	9510186	0.04713671987
11	9204040	9906028	0.03399017476
12	9203084	9308122	0.202453136
13	9203084	9309097	0.2636887544
14	9203084	9702155	0.2038689153
15	9203084	1025	0.2447650978
16	9203084	9308005	0.04427673099
17	9203084	9308083	0.09964916899
18	9203084	9402119	0.2357900687
19	9203084	9403187	0.1949397992
20	9203084	9406055	0.1458600049
21	9203084	9807087	0.03356811661
22	9203084	9912188	0.01378223817
23	9203084	9906242	0.04354687797
24	9203084	9403096	0.2278458165
25	9203084	9611184	0.1905237115
26	9203084	9503017	0.01318714494
27	9203084	9410184	0.1502134532
28	9203084	9504114	0.01283072979
29	9203084	9504115	0

Here the first two columns represent the research papers which form an edge in the network. Whereas the third column represents the cosine similarity between these two research papers.



Data preprocessing:

As each given file consisted of multiple versions, there was a need to filter out the appropriate versions and use it for further works. Once the appropriate files are selected, the topics in each folder have to be extracted by removing the probability scores associated with them. This is done with the following code:

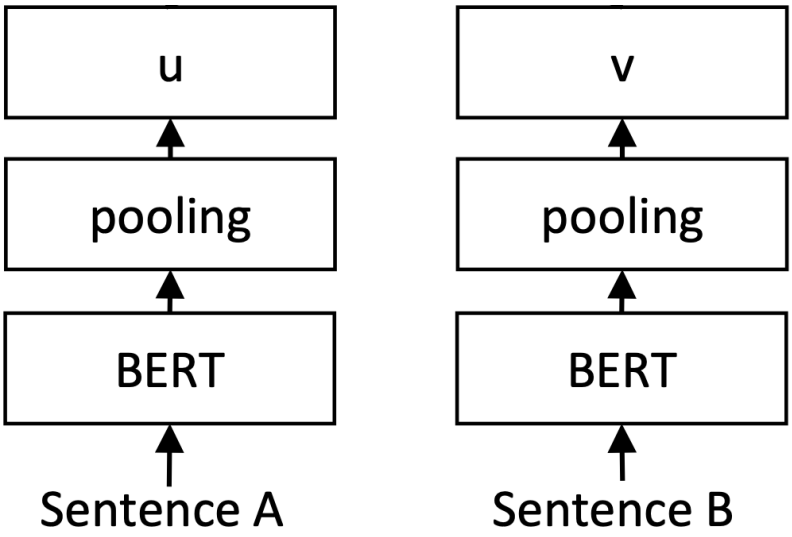
```
path = str('/content/drive/MyDrive/Data Mining (1)/Dataset/Topic_modelled_dataset/Versionwise Input/v1/')
for file in dir2:
  npath = path + file
  dataset=pd.read_csv(npath).values
  topics=[]
  for iter in range(len(dataset)):
    curr=[]
    mystring=dataset[iter][2][3:]
    update=re.sub('[^A-Za-z]+', ' ', mystring)
    curr.append(dataset[iter][1])
    curr.append(update)
    topics.append(curr)

  cleaned=pd.DataFrame(topics)
  fd = r'/content/drive/MyDrive/Data Mining (1)/Output/clean/'
  fd = fd + file
  with open(fd,'w') as f:
    cleaned.to_csv(f)

```

On preprocessing each file is converted to a dataframe that represents the topic score and the topic itself as shown below:

A	B	C
	0	1
0	0.2976672348	gauge group invariant noncommutative transformations full brane theory string therefore
1	0.3312477088	rely two intimately intuitive purpose tion generaliza results describe witten
2	0.3133405112	regardless fact rely context considered understanding replaced jhep clearly carfully
3	0.3460038277	issue fact spacetime volume purpose depends case string th introductory
4	0.3234445055	context references actions results case accordingly project spacetime eld manifold
5	0.3598704649	parameter dieomorphisms ns exist poisson considered geometrically noting able string
6	0.343892349	world spacetime hand various textfiles rst introductory hep tion commutator
7	0.3690934692	using describing elds terms able hep theory connected twoform nonabelian
8	0.3785401566	simpler replacement dieomorphisms project jhep tool arbitrary accordingly rule context
9	0.3455979897	examples exist bigdata mean hand tion model important leave theory
10	0.318930004	ac algebraic simpler reconsider sector works run geometrical extremely large
11	0.3590403243	rule mean following nonlocal must later describing yang substitute leave
12	0.3462566783	algebraic described carfully gauge hep form number cornalba ns keywords
13	0.3209466376	denition reconsider shall carfully works situation sector context general replacement

BERT’s architecture lends itself to be adopted for different kinds of tasks, either through adding task-specific tokens in the input or task-specific networks to the end of the model, utilizing its token embeddings. These modifications allow us to use BERT for, just to name a few, classification, regression, and sentence similarity and in our case topic similarity which is a subset of sentence similarity

SBERT is a so-called twin network which allows it to process two sentences in the same way, simultaneously. These two twins are identical down to every parameter (their weight is tied), which allows us to think about this architecture as a single model used multiple times.



It becomes apparent from the image that BERT makes up the base of this model, to which a pooling layer has been appended. This pooling layer enables us to create a fixed-size representation for input sentences of varying lengths. The authors experimented with different pooling strategies; MEAN- and MAX pooling or utilizing the CLS token BERT per default already generates. How these perform and compare will be discussed later

```
for file in dirList:
    path = datapath + '\\' + file
    dataset = pd.read_csv(path).values
    textContent = []
    for i in range(len(dataset)):
        textContent.append(dataset[i][2])
    sentence_embeddings = sbert_model.encode(textContent,convert_to_numpy=True)

    cleaned=pd.DataFrame(sentence_embeddings)
    fd = r'C:\Users\rishe\Desktop\DMproject\Bert\\'
    fd = fd + file
    with open(fd,'w') as f:
        cleaned.to_csv(f)
```

.

	0	1	2	3	4	5	6	7	8	9	10	11
0	-0.071803	-0.2538758	0.37227425	0.45020485	-0.17623624	-1.0811703	0.25733018	0.11975553	-0.08699177	0.64896935	-1.1609282	0.9289315
1	-0.34851053	0.10583577	1.3778272	0.6960948	0.42308512	-0.6273561	-0.29191524	-0.20511611	0.8510196	-0.7474825	-0.13576633	1.0245354
2	-0.24716708	0.6081026	1.0731839	0.6171064	0.08710299	-0.4791069	-0.25529072	0.34433356	0.576206	-0.27652392	-0.33203548	1.1825448
3	0.14030333	-0.08260977	1.3770226	0.5752324	0.5327238	0.06739094	-0.9242748	0.46717474	0.22756933	-0.32287532	-1.2432708	0.9538332
4	-0.23840846	-0.014737496	1.4683752	0.51070064	0.38039798	0.123280145	-0.9668185	0.34019908	0.05340532	-0.71774495	-0.83033025	1.1277987
5	-0.3633015	-0.16870038	1.3833718	0.6973424	0.22884472	-0.38264596	-0.7164763	-0.115843266	0.2426316	-0.5551799	-0.0918696	1.2476358
6	-0.22724152	0.52771586	0.8600279	0.37036097	0.45120764	0.01312544	-0.74997586	0.8601443	-0.49210912	-0.058314852	-0.65294945	1.1446286
7	-0.27513507	0.16876945	0.38753438	0.32499382	0.39391074	-0.046877116	0.4409628	-0.08584059	0.38670507	-0.54748124	-0.52580637	0.9180293
8	-0.10133973	0.22503191	1.1855443	0.5863521	0.110272616	-0.37653658	-0.079707734	-0.5915722	0.37893468	-0.49964058	-0.41922486	1.0989928
9	0.047025442	-0.07976022	0.83853835	0.1309964	0.60414755	-0.5618276	-0.6519276	0.2312554	-0.40621173	0.35306078	-0.46392342	0.84588325
10	-0.96433294	-0.20640874	0.5812807	0.64103925	-0.5875849	-0.8398292	-0.88823426	0.3203715	0.1142578	-1.0042276	-0.40065405	0.73428816
11	-0.19528233	-0.13794602	1.5741036	0.23566766	0.81353337	-0.12710635	0.19261487	-0.045248237	0.9637841	-0.17429905	-0.8276781	0.62420136
12	-0.29294682	0.6003105	0.6094593	0.49225307	0.322006	-0.02107536	-0.48816934	-0.0489055	0.43140578	0.20801486	-0.21513893	0.8772179
13	-0.10943057	0.43120256	1.8672429	0.38019323	0.3877519	0.44077018	-0.27949977	-0.45622155	0.60389507	-0.7096294	-0.60289073	1.1139688
14	0.2915977	-0.14048298	1.3060731	0.8990157	0.07829271	-0.7375935	-0.6104368	-0.06423255	0.44388643	-0.111740254	-0.038098812	1.0095102

The output we get is of fixed length irrespective how large the topic is.

Computing Correlations:

Once we have generated the embeddings for each topic in all the files, we have to compute how well each topic in one file is correlated with the topics of the other. The files are selected from the nodes in the graph. Since the embeddings generated for each topic is a 1-D array, we use the spatial correlation function. Spatial correlation computes the correlation distance between two 1-D arrays. The correlation distance between *u* and *v*, is defined as:

$$1 - \frac{(u - \bar{u}) \cdot (v - \bar{v})}{||(u - \bar{u})||_2 ||(v - \bar{v})||_2}$$

Where $\bar{u}$ is the mean of the elements of *u* and $x \cdot y$ is the dot product of x and y

```
pathBert = r'/content/drive/MyDrive/Data Mining/Dataset/BertEmbedding/'
pathTopics = r'/content/drive/MyDrive/Data Mining/Dataset/Topics/'

def formNetwork(file1,file2,score):
  Pbert1 = pathBert + str(file1) + '.csv'
  Pbert2 = pathBert + str(file2) + '.csv'
  Ptopics1 = pathTopics + str(file1) + '.csv'
  Ptopics2 = pathTopics + str(file2) + '.csv'
```

```
  try:
    bertEmbed1 = pd.read_csv(Pbert1).values
    bertEmbed2 = pd.read_csv(Pbert2).values
    topics1 = pd.read_csv(Ptopics1).values
    topics2 = pd.read_csv(Ptopics2).values

  except:
    lout = []
    lout.append(file1)
    lout.append(file2)

    fd = r'/content/drive/MyDrive/Data Mining/Output/Error.csv'
    with open(fd,'a') as f:
      write = csv.writer(f)
      write.writerow(lout)
    print("error")
    return

  bertEmbed1 = bertEmbed1[:,1:]
  bertEmbed2 = bertEmbed2[:,1:]

  n1=(len(bertEmbed1))
  n2=(len(bertEmbed2))
  corr=[[-999 for i in range(n1)]for j in range(n2)]
  for i in range(n1):
      for j in range(n2):

corr[i][j]=round(spatial.distance.correlation(bertEmbed1[i],bertEmbed2[j]),
3)

  writeToCSV(corr,file1,file2)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.444	0.551	0.607	0.523	0.61	0.451	0.628	0.678	0.472	0.529	0.537	0.435	0.479	0.547	0.605	0.639	0.598	0.557	0.494	0.457
1	0.469	0.465	0.401	0.442	0.374	0.332	0.333	0.541	0.452	0.316	0.512	0.333	0.55	0.439	0.548	0.314	0.381	0.407	0.31	0.393
2	0.429	0.372	0.384	0.342	0.315	0.349	0.306	0.409	0.39	0.279	0.344	0.393	0.353	0.379	0.455	0.341	0.269	0.38	0.269	0.364
3	0.353	0.435	0.422	0.395	0.398	0.383	0.564	0.507	0.424	0.394	0.538	0.298	0.526	0.451	0.435	0.474	0.472	0.408	0.451	0.44
4	0.421	0.451	0.405	0.46	0.417	0.413	0.548	0.643	0.492	0.467	0.622	0.219	0.635	0.46	0.509	0.449	0.463	0.399	0.488	0.517
5	0.404	0.479	0.342	0.515	0.425	0.365	0.428	0.526	0.427	0.36	0.564	0.327	0.523	0.38	0.451	0.446	0.412	0.422	0.316	0.319
6	0.512	0.411	0.462	0.423	0.519	0.447	0.654	0.644	0.498	0.544	0.621	0.415	0.646	0.46	0.635	0.532	0.531	0.497	0.575	0.591
7	0.477	0.408	0.403	0.389	0.415	0.269	0.33	0.46	0.335	0.297	0.34	0.326	0.363	0.424	0.434	0.349	0.319	0.389	0.249	0.286
8	0.453	0.439	0.385	0.404	0.362	0.393	0.394	0.416	0.358	0.297	0.401	0.415	0.372	0.377	0.321	0.483	0.297	0.369	0.266	0.245
9	0.304	0.354	0.378	0.328	0.372	0.342	0.437	0.345	0.439	0.381	0.501	0.408	0.586	0.356	0.503	0.293	0.375	0.332	0.496	0.556
10	0.326	0.6	0.429	0.578	0.663	0.531	0.518	0.462	0.472	0.484	0.623	0.512	0.607	0.402	0.566	0.576	0.577	0.618	0.525	0.503
11	0.452	0.42	0.418	0.3	0.268	0.381	0.497	0.487	0.36	0.376	0.4	0.395	0.408	0.411	0.329	0.358	0.272	0.272	0.379	0.35
12	0.344	0.44	0.297	0.375	0.452	0.315	0.508	0.452	0.355	0.408	0.523	0.354	0.454	0.293	0.493	0.457	0.456	0.44	0.397	0.422
13	0.423	0.401	0.282	0.336	0.205	0.372	0.476	0.431	0.367	0.319	0.485	0.337	0.483	0.316	0.279	0.356	0.24	0.269	0.373	0.403
14	0.394	0.431	0.341	0.357	0.34	0.371	0.482	0.548	0.352	0.358	0.565	0.234	0.569	0.375	0.496	0.378	0.392	0.354	0.459	0.48
15	0.557	0.59	0.478	0.579	0.505	0.52	0.463	0.561	0.469	0.393	0.523	0.494	0.494	0.533	0.455	0.608	0.475	0.54	0.357	0.325
16	0.497	0.463	0.449	0.42	0.465	0.404	0.411	0.42	0.376	0.364	0.376	0.501	0.362	0.404	0.44	0.475	0.375	0.441	0.312	0.292
17	0.352	0.442	0.316	0.38	0.431	0.419	0.523	0.46	0.359	0.334	0.491	0.418	0.446	0.237	0.482	0.513	0.433	0.466	0.372	0.318
18	0.345	0.421	0.529	0.411	0.568	0.402	0.382	0.332	0.397	0.378	0.288	0.56	0.351	0.468	0.566	0.473	0.467	0.548	0.419	0.425
19	0.5	0.431	0.306	0.347	0.352	0.412	0.547	0.545	0.372	0.345	0.515	0.419	0.47	0.299	0.398	0.405	0.307	0.319	0.348	0.351

The image above represents the correlation of two articles, rows talk about the topics of say, article A and columns talk about that of B and each cell gives the score of correlation between the topics.

Once the correlations are calculated we are supposed to find the top five highly correlated topics between two nodes in the graph.Which are paramount for the detection of persistence in topic evolution.

```
def topFive(matrix):
 qSize = 5
 queue = []
 for i in range(qSize):
   arr = []
   arr.append(0)
   arr.append(i)
   arr.append(matrix[0][i])
   queue.append(arr)
 queue.sort(key = cond)
  for i in range(qSize,len(matrix[0])):
   if matrix[0][i] < cond(queue[0]):
     continue
   else:
     queue.pop(0)
     arr = []
     arr.append(0)
     arr.append(i)
     arr.append(matrix[0][i])
     queue.append(arr)
     queue.sort(key = cond)

 for j in range(1,len(matrix)):
   for k in range(j,len(matrix[j])):
     if matrix[j][k] < cond(queue[0]):
       continue
     else:
       queue.pop(0)
       arr = []
       arr.append(j)
       arr.append(k)
       arr.append(matrix[j][k])
       queue.append(arr)
       queue.sort(key = cond)
 return queue
```

The top five highly correlated topics along with their nodes are stored in a csv file.In order to give us a clear understanding of how the topics have evolved.

9204040	9401139	0.11088564	[[6, 19, 0.668], [1, 19, 0.669], [0, 4, 0.672], [2, 3, 0.672], [7, 10, 0.684]]
9204040	9405002	0.1246353905	[[1, 19, 0.684], [1, 6, 0.686], [16, 19, 0.686], [2, 6, 0.689], [1, 4, 0.69]]
9204040	9703156	0.09022430015	[[16, 16, 0.613], [7, 14, 0.617], [1, 16, 0.623], [7, 15, 0.623], [15, 16, 0.646]]
9204040	9707203	0.03662195308	[[2, 2, 0.666], [1, 3, 0.693], [12, 12, 0.693], [7, 12, 0.713], [0, 3, 0.773]]
9204040	204145	0.04371121059	[[1, 12, 0.668], [2, 12, 0.677], [7, 13, 0.707], [7, 7, 0.708], [16, 18, 0.724]]
9204040	9505123	0.06212134891	[[16, 17, 0.687], [1, 7, 0.688], [16, 16, 0.702], [7, 12, 0.714], [1, 14, 0.741]]
9204040	9306074	0.1370872402	[[2, 3, 0.656], [1, 3, 0.659], [2, 15, 0.663], [15, 15, 0.681], [1, 15, 0.746]]
9204040	9906028	0.03399017476	[[2, 7, 0.674], [7, 11, 0.681], [1, 15, 0.682], [1, 19, 0.716], [1, 7, 0.753]]
9203084	9702155	0.2038689153	[[2, 18, 0.706], [1, 2, 0.716], [0, 5, 0.739], [1, 6, 0.757], [2, 6, 0.758]]
9203084	1025	0.2447650978	[[1, 3, 0.66], [1, 6, 0.66], [2, 3, 0.662], [2, 6, 0.665], [1, 8, 0.698]]
9203084	9402119	0.2357900687	[[1, 2, 0.709], [8, 11, 0.717], [1, 11, 0.723], [1, 1, 0.728], [2, 11, 0.75]]
9203084	9403187	0.1949397992	[[2, 18, 0.67], [2, 19, 0.672], [1, 19, 0.676], [2, 14, 0.693], [1, 14, 0.7]]
9203084	9912188	0.01378223817	[[1, 3, 0.742], [1, 2, 0.744], [2, 2, 0.747], [2, 3, 0.756], [1, 18, 0.762]]
9203084	9906242	0.04354687797	[[2, 6, 0.668], [1, 6, 0.679], [1, 18, 0.696], [1, 15, 0.746], [2, 15, 0.759]]
9203084	9403096	0.2278458165	[[1, 11, 0.77], [2, 11, 0.773], [2, 12, 0.775], [1, 16, 0.794], [1, 12, 0.838]]
9203084	9611184	0.1905237115	[[6, 17, 0.669], [13, 16, 0.685], [2, 8, 0.729], [1, 3, 0.735], [2, 3, 0.736]]
9203084	9503017	0.01318714494	[[1, 5, 0.658], [1, 11, 0.671], [1, 2, 0.694], [2, 14, 0.697], [1, 14, 0.779]]
9203084	9410184	0.1502134532	[[1, 13, 0.697], [1, 14, 0.697], [1, 7, 0.773], [1, 4, 0.799], [2, 4, 0.808]]
9203084	9504114	0.01283072979	[[1, 5, 0.719], [1, 17, 0.728], [1, 15, 0.73], [2, 7, 0.744], [2, 17, 0.76]]

Topic Evolution:

For the given path file we check the above generated file to check which topic has evolved into which one and in case if there is no match in the top five correlated topics, we check the correlation matrix between the nodes and find the highly correlated topic given a particular topic from a file.We have set the threshold for correlation to be 0.35. If we are unable to find a correlation score greater than that we assume the topic to be dead in that path and the following evolutions are filled with -1.

```
paths=pd.read_csv('/Users/vishalbharadwaj/Downloads/PATH TOPICS
FINAL.csv').values
val=0
AllFiles=[]
AllTopics=[]
for val in range(len(paths)):
    if val%2==1:
        continue
    first = paths[val][0][1:-1]
    first = first.split("], ")
    files = []
    topics = []
    for i in first:
        i = i.replace("[","")
        i = i.replace(']',"")
        i = i.split(", ")
    for j in i:
        j=j[1:-1]
        files.append(int(j))
        AllFiles.append(files)
    second=paths[val][1][1:-1]
    second=second.split("], ")
    for i in second:
        i = i.replace("[","")
        i = i.replace(']',"")
        i = i.split(", ")
        topics.append(i)
        AllTopics.append(topics)
    val=val+1
```

The following code is used to change the length of the file by adding leading zeroes:

```
def expand(file):
    file = str(file)
    if len(file) == 7:
        return str(file)
    else:
        num = 7-len(file)
```

```
        for i in range(0,num):
            file = str(0) + file
    return str(file)
```

```
def fetchCorrelation(start,topic1,destination,topic2):
    path = r'/Users/vishalbharadwaj/Downloads/Correlation-deepscore/' +
str(expand(start)) + '-' + str(expand(destination)) + '.csv'
    if os.path.isfile(path):
        value = pd.read_csv(path).values
        row = int(topic1)
        column = int(topic2)+1
        return value[row][column]
    else:
        path = r'/Users/vishalbharadwaj/Downloads/Correlation-deepscore/' +
str(expand(destination)) + '-' + str(expand(start)) + '.csv'
        value = pd.read_csv(path).values
        row = int(topic2)
        column = int(topic1)+1
        return value[row][column]



def writeNetwork(file1,topic1,file2,topic2):
    lout = []
    score = fetchCorrelation(file1,topic1,file2,topic2)
    entry1 = str(file1) + '-' + str(topic1)
    entry2 = str(file2) + '-' + str(topic2)

    lout.append(entry1)
    lout.append(entry2)
    lout.append(score)
    list1.append(lout)
```

```
linkups=[]
count=0
for row in range(len(paths)):
    for col in range(len(AllTopics[row])):
        for val in range(len(AllTopics[row][col])-1):
            if(AllTopics[row][col][val]=='-1' or
AllTopics[row][col][val+1]=='-1'):
```

```
                continue

            try:

src=str(AllFiles[row][val])+'-'+str(AllTopics[row][col][val])

dst=str(AllFiles[row][val+1])+'-'+str(AllTopics[row][col][val+1])
                link=[]
                link.append(src)
                link.append(dst)

writeNetwork(AllFiles[row][val],AllTopics[row][col][val],AllFiles[row][val+
1],AllTopics[row][col][val+1])
                linkups.append(link)
            except:
                count=count+1
```

```
def relationCheckFirst(start,destination):
 topTopics = pd.read_csv('/content/drive/MyDrive/Data
Mining/topFive-deepscore.csv').values
 for i in range(len(topTopics)):
   if topTopics[i][0] == int(start) and topTopics[i][1] == int(destination):
     s = topTopics[i][3]
     array = topicsArray(s,0)
     return array

   elif topTopics[i][0] == int(destination) and topTopics[i][1] == int(start):
     s = topTopics[i][3]
     array = topicsArray(s,1)
     return array

 formNetwork(int(start),int(destination),1.11)
 return relationCheckFirst(start,destination)

def relationCheckLast(start,destination,superArray):
 topTopics = pd.read_csv('/content/drive/MyDrive/Data
Mining/topFive-deepscore.csv').values
 for i in range(len(topTopics)):
```

```
   if topTopics[i][0] == int(start) and topTopics[i][1] == int(destination):
     s = topTopics[i][3]
     array = topicsArray(s,0)
     return topicAdder(superArray, array,start=start, destination=destination)

   elif topTopics[i][0] == int(destination) and topTopics[i][1] == int(start):
     s = topTopics[i][3]
     array = topicsArray(s,1)
     return topicAdder(superArray, array,start=start, destination=destination)

 formNetwork(int(start),int(destination),1.11)
 return relationCheckLast(start,destination,superArray)


def topicAdder(superArray, array, start, destination):
 arr = superArray
 newArr = []
 for i in arr:
   number = i[-1]
   count = 0
   for j in array:
     if j[0] == number:
       a=[]
       count = count + 1
       a = copy.deepcopy(i)
       a.append(j[1])
       newArr.append(a)
   if count == 0:
     try:
       path = r'/content/drive/MyDrive/Data
Mining/Output/Correlation-deepscore/' + expand(start) + '-' +
expand(destination) + '.csv'
     except:
       path = r'/content/drive/MyDrive/Data
Mining/Output/Correlation-deepscore/' + expand(destination) + '-' +
expand(start) + '.csv'
     value = pd.read_csv(path).values
     row = number
     print(value[row] , number)
     max = -99999
     index = -1
     for column in range(1,len(value[row])):
       if value[row][column] > max and value[row][column] >=
correlationThreshold:
         index = column-1
```

```
            max = value[row][column]
        a = copy.deepcopy(i)
        a.append(index)
        newArr.append(a)

    return newArr

count = 0
for i in range(len(lisd)):
    try:
      path = BertChecker(lisd[i]):
      superArray = relationCheckFirst(path[i][0] , path[i][1])
      begin=1
      end=begin+1
      while end<len(path[i]):
        start=expand(path[i][begin])
        dest=expand(path[i][end])
        superArray = relationCheckLast(start,dest,superArray)
        begin = begin+1
        end = end+1
      writeNetwork(path,superArray)

    except:
      fd = r'/content/drive/MyDrive/Data Mining/Output/Error-pathFinder.csv'
      with open(fd,'a') as f:
        write = csv.writer(f)
        write.writerow(lisd[i])
```

The output is a list of arrays each of the same dimension of the path and the topic evolution is arranged in the same order of the paths.

['9207071', '9601070', '9904105', '5187', '105305', '208087']	[[2, 15, 18, 6, 14, 9], [2, 15, 15, 19, 14, 9], [2, 15, 19, 16, 16, 6], [5, 15, 18, 6, 14, 9], [5, 15, 15, 19, 14, 9], [5, 15, 19
['9410188', '9503157', '9709138', '9711001', '9805191', '9912185', '12145', '106048', '20	[[2, 16, 9, 16, 10, 19, 0, 1, 13, 16], [2, 16, 9, 11, 17, 17, 8, 15, 12, 17], [2, 16, 9, 11, 18, 18, 19, 8, 11, 9], [2, 16, 9, 1
['9201036', '9307176', '9508030', '9610224', '9805160', '212084']	[[11, 15, 16, 12, 15, 4], [11, 15, 16, 12, 19, 4], [11, 15, 16, 12, 14, 1], [11, 15, 16, 12, 13, 1], [11, 15, 16, 12, 16, 4], [
['9512066', '9612114', '203020', '204054', '208074', '209179', '301009']	[[19, 19, 3, 16, 15, 13, 9], [19, 19, 3, 14, 17, 13, 9], [19, 19, 3, 9, 13, 13, 9], [18, 19, 3, 16, 15, 13, 9], [18, 19, 3, 14,
['9311081', '9607026', '9609017', '9612157', '9703030', '9707093', '9803196', '9907152',	[[2, 3, 5, 16, 16, 5, 0, 10, 0], [2, 3, 5, 16, 16, 5, 0, 15, 13], [2, 3, 5, 16, 16, 5, 0, 17, 14], [2, 3, 5, 16, 16, 5, 0, 12, 14]
['9304045', '9601108', '9604034', '9706109', '9707093', '9711200', '9712251', '9902098',	[[8, 8, 18, 9, 5, 12, 16, 3, 8, 14], [8, 8, 12, 9, 5, 12, 16, 3, 8, 14], [0, 6, 1, 13, 8, 13, 14, 5, 7, 8], [0, 8, 18, 9, 5, 12, 16
['9202059', '9703206', '9707126', '9710151', '9908160', '11086', '106015', '205066', '206	[[5, 10, 15, 19, 10, 15, 18, 13, 4, 5], [5, 10, 12, 12, 14, 1, 5, 11, 19, 6], [5, 10, 12, 12, 14, 1, 5, 12, 19, 6], [5, 10, 12,
['9211121', '9505123', '9607058', '9701015', '9706047', '9801134', '9905111', '9906056',	[[11, 15, 18, 4, 13, 4, 16, 12, 18, 4, 4, 1, 15], [11, 15, 18, 4, 13, 4, 16, 12, 18, 4, 4, 1, 5], [11, 15, 18, 4, 13, 4, 16, 12
['9702194', '9712257', '2211', '9002', '102076', '103209', '106013', '106200', '204167']	[[0, 10, 15, 7, 16, 13, 14, 19, 9], [0, 10, 15, 7, 16, 13, 16, 2, 9], [0, 10, 15, 7, 16, 13, 18, 10, 7], [10, 15, 13, 3, 16, 13
['9407102', '9608105', '9704197', '9707126', '9709062', '9802067', '9805141', '9909028',	[[8, 10, 12, 3, 0, 5, 5, 17, 19, 1, 5, 13], [8, 10, 12, 3, 0, 5, 5, 17, 19, 1, 6, 17], [8, 10, 12, 3, 0, 5, 5, 17, 19, 1, 8, 10],
['9206106', '2245', '103110', '105005', '105235', '107261', '206045', '205314', '211257']	[[0, 12, 17, 12, 6, 18, 3, 18, 6], [6, 9, 17, 12, 6, 18, 3, 18, 6], [17, 18, 17, 12, 6, 18, 3, 18, 6], [14, 15, 12, 5, 14, 12, 1
['9401062', '9709101', '5075', '110058']	[[4, 14, 9, 9], [6, 12, 9, 9], [4, 8, 2, 3], [4, 4, 9, 9], [4, 4, 15, 6], [4, 4, 10, 3], [4, 4, 6, 19], [4, 12, 9, 9]]
['9212049', '9601151', '7146', '202094', '202127', '108119', '112218']	[[5, 13, 14, 19, 3, 18, 18], [5, 13, 14, 19, 3, 11, 4], [5, 13, 14, 18, 0, 4, 6], [5, 13, 13, 18, 0, 4, 6], [0, 12, 14, 19, 3, 18
['9206026', '9602064', '9604119', '9707205', '9708166', '9712186', '9712257', '6181', '91	[[15, 19, 11, 2, 19, 0, 10, 17, 17, 17], [15, 19, 11, 2, 19, 0, 10, 17, 17, 19], [12, 19, 11, 2, 19, 0, 10, 17, 17, 17], [12,

Graph generation:

```
edge = []
for row in range(len(df)):
    if row%2 == 0:
        l = []
        l.append(df[row][0])
        l.append(df[row][1])
        edge.append(l)
edge

G = nx.Graph()
G.add_edges_from(edge)
pos = nx.spring_layout(G)
f = plt.figure()
f.set_figwidth(4000)
f.set_figheight(6000)
```

```
nx.draw(
    G, pos, edge_color='yellow', width=1, linewidths=1,
    node_size=1000, node_color='pink', alpha=0.9,
    labels={node: node for node in G.nodes()}
)
```