Indian Institute of Technology, Kanpur
Computer Science and Engineering

CS685 Data Mining Project Report - Group 9

Analysis of Research trends in Computer Science

Team Members:
Anirudh Nanduri(21111011)
Mail Id: anirudhn21@iitk.ac.in

Archi Gupta(21111014)
Mail Id: archig21@iitk.ac.in

Nagarjun Reddy(21111031)
Mail Id: nagarjunr21@iitk.ac.in

Siddhartha(21111059)
Mail Id: bsiddharth21@iitk.ac.in

Akhil Ratna(180360)
Mail Id: akhilbbn@iitk.ac.in

Supervisor:
Prof. Arnab Bhattacharya

Year 2021-22

Acknowledgement

We would like to extend our sincere gratitude to our Project Guide, Prof. Arnab Bhattacharya for his advice, guidance, patience and timely help during the project. The freedom he gave us, to work with anything at any time, encouraged us to try out new things and helped to work more efficiently.

Anirudh Nanduri(21111011)
Archi Gupta(21111014)
Nagarjun Reddy(21111031)
Siddhartha(21111059)
Akhil Ratna(180360)

Contents

List of Figures

Abstract

The Digital Bibliography and Library Project (DBLP) is a popular computer science bibliography website hosted at the University of Trier in Germany. It currently contains 2,722,212 computer science publications with additional information about the authors and conferences, journals, or books in which these are published. Although the database covers the majority of papers published in this field of research, it is still hard to browse the vast amount of textual data manually to find insights and correlations in it, in particular time-varying ones. In this project we use the DBLP data to analyze trends (geographical, temporal, social).We have shown the results in easily interpretative diagrams.

Geographical research trends based on the author's origins. Additional computation details of author's origin is mentioned in section 3.1. Topic modelling of publication data (abstract, title, keywords, field of study) is done to understand broad topics and commonly occurring keywords.Analysis and clustering of phd thesis titles available in dblp dataset is done. Relationship between co-authors is analyzed using graphical models.Association rule mining is done to find out works that occur together in publication titles.Correlation between words in publication is also found out.Finally year-wise distribution of publication data in the dataset is obtained.

These results give a broad prespective of the research trends in computer science. Summary of the results is mentioned in Results(Section 4) of the report

Chapter 1

1 Introduction and Broad Aims of the Project

- Analysis of data provided by dblp website [1] is done in this project to understand temporal and social research trends.

- The Digital Bibliography and Library Project (DBLP) is a popular computer science bibliography website hosted at the University of Trier in Germany. It currently contains 2,722,212 computer science publications with additional information about the authors and conferences, journals, or books in which these are published.
  Data present in dblp has been consolidated by aminer website [2] and another dataset DBLP-Citation-network V13[3] is used in this project.

- Geographical trends of research is obtained by attributing authors to their origins[section 3.1].Based on that country-wise research activity and yearly research activity in each country is obtained.

- In this project efforts have been made to model topics from research publication data(abstract,field of study,key words,title).Analysis on PhD thesis titles and general analysis on publication titles is done.

- Patterns have been obtained between co-authors and prediction of co-authors is performed.Jacaard similarity score between co-authors is obtained.This helps us to understand the similarity between them and the possibility that they may work in the future.

- Associate rule mining is done to find frequently occurring words in computer science research publications.FP-tree algorithm is used for associative rule mining.FP-tree algorithm is used to find associative rules because of its speed of computation and it is easy to understand.Phi correlation is used to find the between words in publication titles.

- Year wise distribution of publications is found out.It gives an understanding of temporal trends of publications.

2 Data Extraction and Preperation

- Major challenge for the data processing faced in this project is the size of dataset.dblp.xml [1] is around 3.5GB and aminer citation dataset [2] is 16GB which render inmemory operations useless.File I/O operations have been used extensively which increased the overall latency of preprocessing.
- In memory json and xml parsers could not be used because of the large size of data files.
- Data is extracted from the dataset by reading the file line by line and selecting required fields. https://www.aminer.org/citation[3]. This website is used because it provides abstract, key-words and field of study information which is not provided by dblp.
- The type of data used in this proj is textual.Hence most common operations performed on the data are tokenization, vectorization, substring matching, word frequency calculation, lemmatization and string comparision.
- To vectorize data , first data is tokenized(split on space).Long text strings are converted to list of words.
- Lemmatization is performed to convert the string in its root form.[Section 3.2]
- For the vectorized representation of text bag-of-words concept and tf-idf algorithm is used.
- It has been observed that many publications are in languages other than English(Eg. German, English) which were decreasing coherence of results.Such publications have been filtered out to improve accuracy and interpretability.

3 Data analysis

The whole analysis is done in different phases as follows:

3.1 Geographical Trends in Research

The attribution of a publication to any geography has been done on the basis of its authors' origins. There is no explicit information about the author's resident country in the raw data but both json and xml files have attributes for author's organisation given as 'org' and <school> respectively. In majority of the cases the country is also mentioned. The xml has this attribute for only 1% of the entire publication data whereas it was available for 60% of the authors in the json file. Finally, we used the 16 GB json file in order to retrieve the country information. We also used another file containing all the countries names for string matching. We were able to extract year wise publication data and number of citations for all countries. We have compared the total publications with total citations of a country since citations is a strong measure of quality of research. Moreover, we have also compared the number of publications with the country's population to get a paper:population index. We have defined it as the scaled ratio of total research papers to the total population of the country. It has been found that there is absolutely no correlation between the total number of papers published and paper:population index. Some European countries and Singapore has exceptionally high paper:population index.

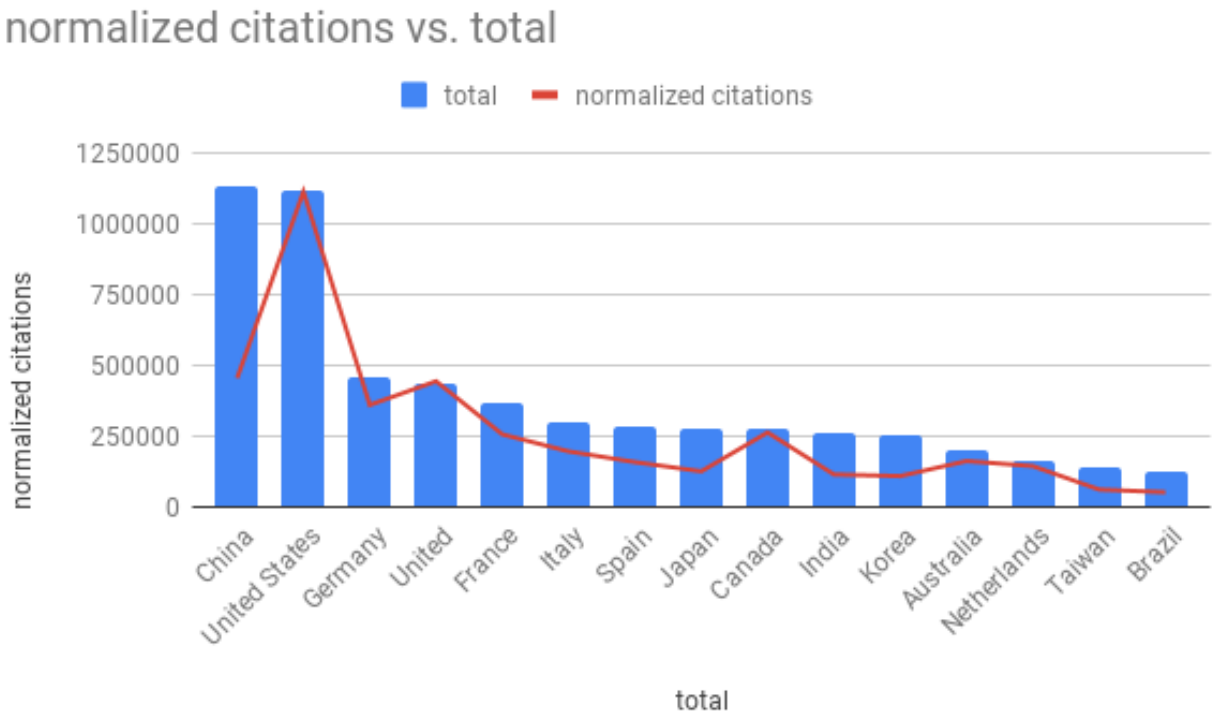

Figure 3.1: normalized citations vs. total publications

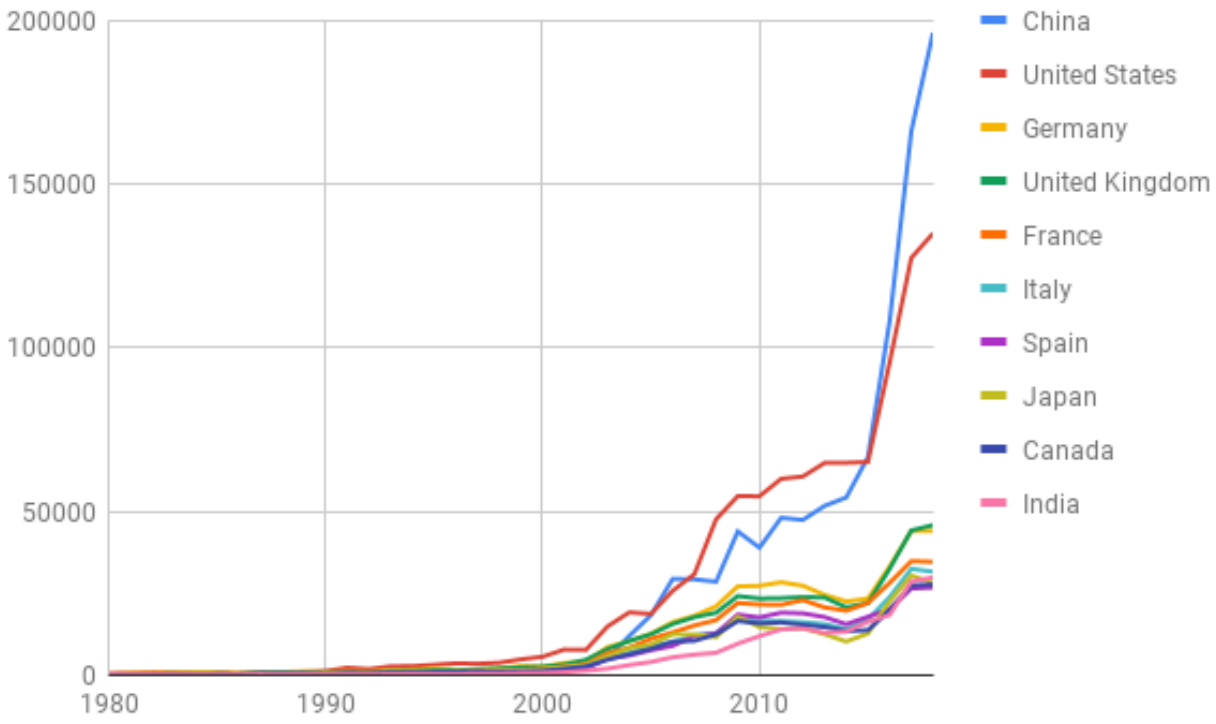

Figure 3.2: year wise trends across geographies

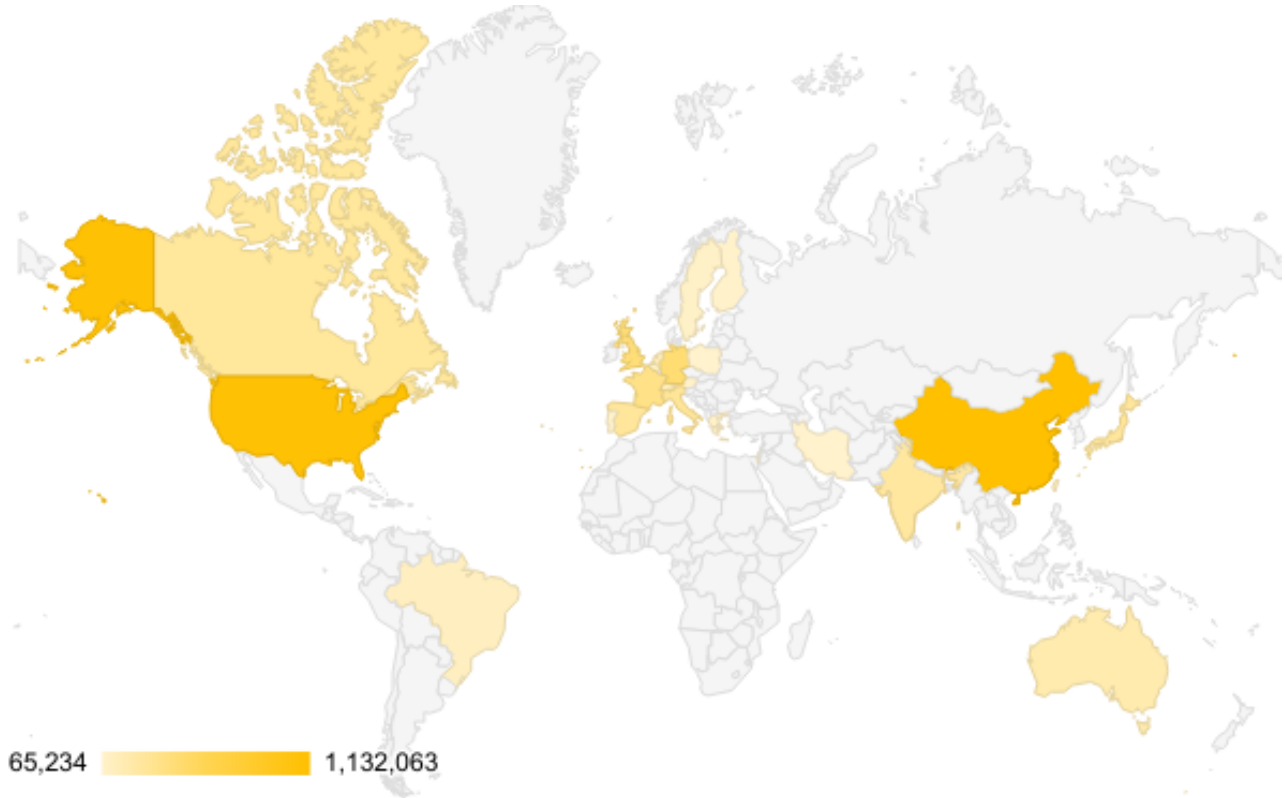

Figure 3.3: Number of publications across geographies

Name	total	Papers:Population Index
China	1132063	8
United States	1115296	34
Germany	460499	55
United Kingdom	435363	64
France	365324	56
Italy	300006	50
Spain	282229	60
Japan	277772	22
Canada	274790	73
India	261023	2
Korea	252959	33
Australia	195720	77
Netherlands	158192	92
Taiwan	135556	57
Brazil	119559	6
Hong Kong	112024	149
Singapore	109523	187
Switzerland	107621	124
Belgium	91080	79
Sweden	85817	85
Austria	83414	93
Greece	81299	78

Figure 3.4: Total publications vs Paper : population index

3.2 Topic modelling of research data and extracting trends

Data from title, abstract,key-words and field of study(fos) of a publication(journal,conference, book) is extracted from DBLP-Citation-network V13.File operations have been extensively used because of the large size of dataset(16GB). Trending topics for every 20 years (1961-1981, 1981-2001, 2001-2021) are taken.

- Data is pre-processed by tokenization and lemmatization is performed.

  1. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meanings to one word.
  2. Examples of lemmatization:

     -> rocks : rock
     -> corpora : corpus
     -> better : good

- TF-IDF is used to convert document to vector If abstract data and field of study fields are also present in that particular publication then they are appended to the title.

  1. TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a

text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

2. Term Frequency: In document d, the frequency represents the number of instances of a given word t. Therefore, we can see that it becomes more relevant when a word appears in the text, which is rational. Since the ordering of terms is not significant, we can use a vector to describe the text in the bag of term models. For each specific term in the paper, there is an entry with the value being the term frequency.
$tf(t,d) = count\ of\ t\ in\ d / number\ of\ words\ in\ d$

3. Document Frequency: This tests the meaning of the text, which is very similar to TF, in the whole corpus collection. The only difference is that in document d, TF is the frequency counter for a term t, while df is the number of occurrences in the document set N of the term t. In other words, the number of papers in which the word is present is DF.
$df(t) = occurrence\ of\ t\ in\ documents$

4. Inverse Document Frequency: Mainly, it tests how relevant the word is. The key aim of the search is to locate the appropriate records that fit the demand. Since tf considers all terms equally significant, it is therefore not only possible to use the term frequencies to measure the weight of the term in the paper. First, find the document frequency of a term t by counting the number of documents containing the term:
$df(t) = N(t)$
where
$df(t) = Document\ frequency\ of\ a\ term\ t$
$N(t) = Number\ of\ documents\ containing\ the\ term\ t$

$$tf - idf(t,d) = tf(t,d) * idf(t)$$

Eg.

```
d0 = 'term1 term2 term1'
d1 = 'term1'
d2 = 'term3'

# merge documents into a single corpus
string = [d0, d1, d2]
Word indexes:
{'term1':0, 'term2':1, 'term2':2}
tf-idf value
(0,0) 0.549351
(0,1) 0.835591
(1,1) 1.0
(1,2) 1.0
```

- Topic modelling is doe using LDA algorithm [4] (Latent Dirichlet Allocation) after the data is vectorized using tf-idf.

1. Topic modeling is a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. Latent Dirichlet Allocation (LDA) is an example of topic model and is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.
2. The following steps are carried out in LDA [5] to assign topics to each of the documents:

   1) For each document, randomly initialize each word to a topic amongst the K topics where K is the number of pre-defined topics.

   2) For each document d:

   For each word w in the document, compute:

   $P(topict|documentd)$:Proportion of words in document d that are assigned to topic t

   $P(wordw|topict)$: Proportion of assignments to topic t across all documents from words that come from w

   3) Reassign topic T' to word w with probability p(t'|d)*p(w|t') considering all other words and their topic assignments

   The last step is repeated multiple times till we reach a steady state where the topic assignments do not change further. The proportion of topics for each document is then determined from these topic assignments.

1. Import the data from DBLP-Citation-network V13 file
2. The below algorithm is applied for the data starting from 1961 every 20 years interval.
3. publication data(title+abstract+keywords+field of study) is converted to tokens and pre-processed using lemmatization.
4. tf-idf algorithm is applied on the data to convert it to a list of vectors.
5. Least frequent and most frequent terms are removed and lda algorithm is applied to get the topics
6. Select the top topics as the result.

Results

- Year 1961 to 1981
  ****

  Topic: 0 Word: 0.036*"Mathematics" + 0.030*"algorithm" + 0.022*"Algorithm" + 0.019*"Applied mathematics" + 0.018*"Mathematical optimization" + 0.018*"linear" + 0.016*"problem" + 0.014*"Mathematical analysis" + 0.014*"cod" + 0.014*"method"

  Topic: 1 Word: 0.044*"Artificial intelligence" + 0.039*"graph" + 0.026*"Computer science" + 0.024*"pattern" + 0.023*"model" + 0.022*"theory" + 0.022*"recognition" +

0.021*"Combinatorics" + 0.021*"network" + 0.020*"Mathematics"

Topic: 2 Word: 0.018*"Information retrieval" + 0.016*"image" + 0.015*"Computer science" + 0.014*"analysis" + 0.013*"test" + 0.012*"automatic" + 0.012*"time" + 0.011*"process" + 0.010*"information" + 0.010*"retrieval"

Topic: 3 Word: 0.040*"Discrete mathematics" + 0.038*"Mathematics" + 0.029*"Combinatorics" + 0.027*"Algebra" + 0.016*"logic" + 0.014*"theorem" + 0.014*"function" + 0.014*"set" + 0.013*"finite" + 0.012*"class"

Topic: 4 Word: 0.020*"systems" + 0.017*"data" + 0.016*"design" + 0.016*"information" + 0.013*"software" + 0.012*"Computer science" + 0.012*"science" + 0.011*"network" + 0.011*"program" + 0.010*"control"

Topic: 5 Word: 0.064*"Computer science" + 0.047*"Programming language" + 0.038*"program" + 0.035*"language" + 0.022*"data" + 0.021*"structure" + 0.020*"languages" + 0.013*"base" + 0.013*"design" + 0.011*"implementation"

Topic: 6 Word: 0.020*"simulation" + 0.018*"model" + 0.016*"Engineering" + 0.016*"review" + 0.014*"signal" + 0.013*"filter" + 0.012*"Electronic engineering" + 0.012*"frequency" + 0.011*"digital" + 0.010*"circuit"
****

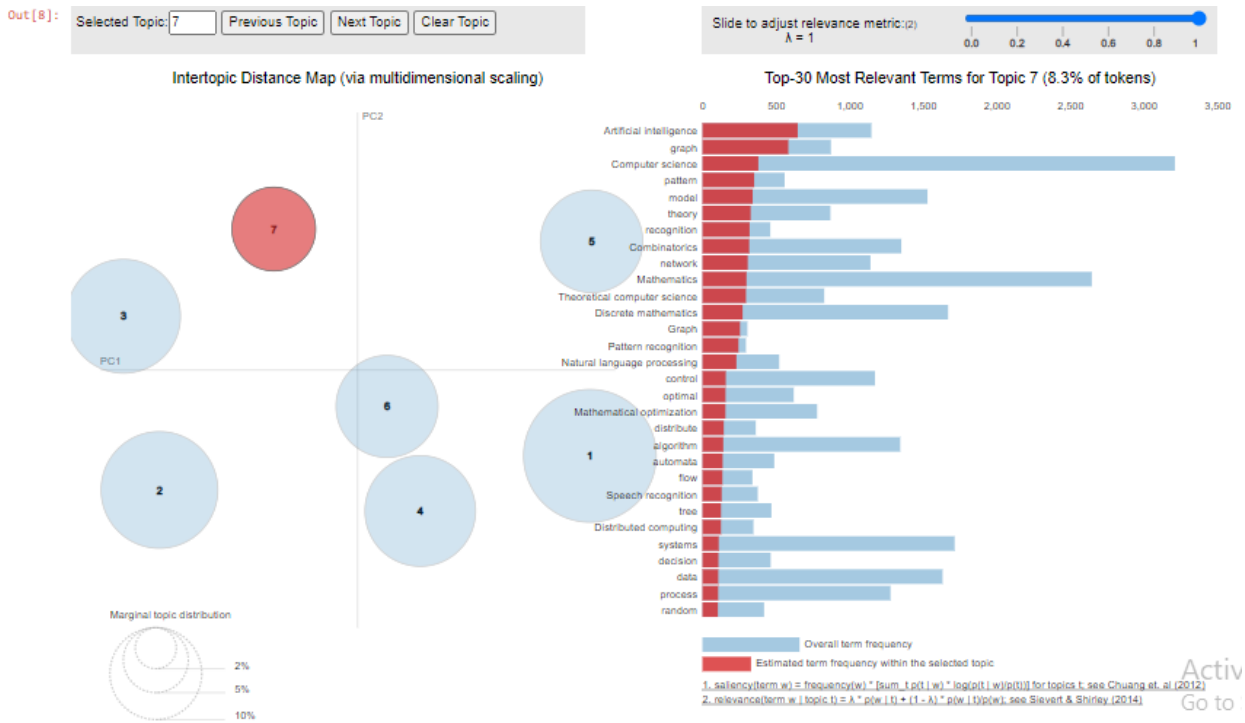

Figure 3.5: 1961-1981 trending research topics

- Year 1981 to 2021
  ****
  Topic: 0 Word: 0.032*"Programming language" + 0.028*"program" + 0.023*"software" + 0.021*"language" + 0.019*"object" + 0.016*"Computer science" + 0.016*"database" + 0.016*"Software engineering" + 0.015*"logic" + 0.015*"model"
  Topic: 1 Word: 0.045*"image" + 0.025*"Computer vision" + 0.020*"Artificial intelligence" + 0.013*"motion" + 0.013*"surface" + 0.013*"method" + 0.012*"shape" + 0.012*"model" + 0.012*"object" + 0.011*"feature"
  Topic: 2 Word: 0.019*"information" + 0.019*"systems" + 0.015*"design" + 0.015*"service" + 0.014*"management" + 0.014*"network" + 0.013*"Multimedia" + 0.012*"Computer science" + 0.012*"Human–computer interaction" + 0.012*"Engineering"
  Topic: 3 Word: 0.075*"Computer science" + 0.066*"neural" + 0.064*"Artificial intelligence" + 0.058*"speech" + 0.052*"network" + 0.049*"Speech recognition" + 0.043*"Artificial neural network" + 0.038*"learn" + 0.032*"Natural language processing" + 0.030*"recognition"
  Topic: 4 Word: 0.061*"Mathematics" + 0.056*"Combinatorics" + 0.055*"Discrete mathematics" + 0.043*"graph" + 0.023*"tree" + 0.015*"number" + 0.014*"bound" + 0.013*"proof" + 0.013*"set" + 0.012*"complexity"
  Topic: 5 Word: 0.017*"parallel" + 0.017*"performance" + 0.015*"Parallel computing" + 0.015*"simulation" + 0.014*"channel" + 0.013*"circuit" + 0.013*"time" + 0.012*"memory" + 0.011*"Electronic engineering" + 0.011*"fault"
  Topic: 6 Word: 0.016*"control" + 0.016*"Control theory" + 0.016*"Mathematical optimization" + 0.014*"problem" + 0.013*"algorithm" + 0.013*"Mathematics" + 0.012*"function" + 0.012*"method" + 0.012*"model" + 0.011*"linear"
  ****

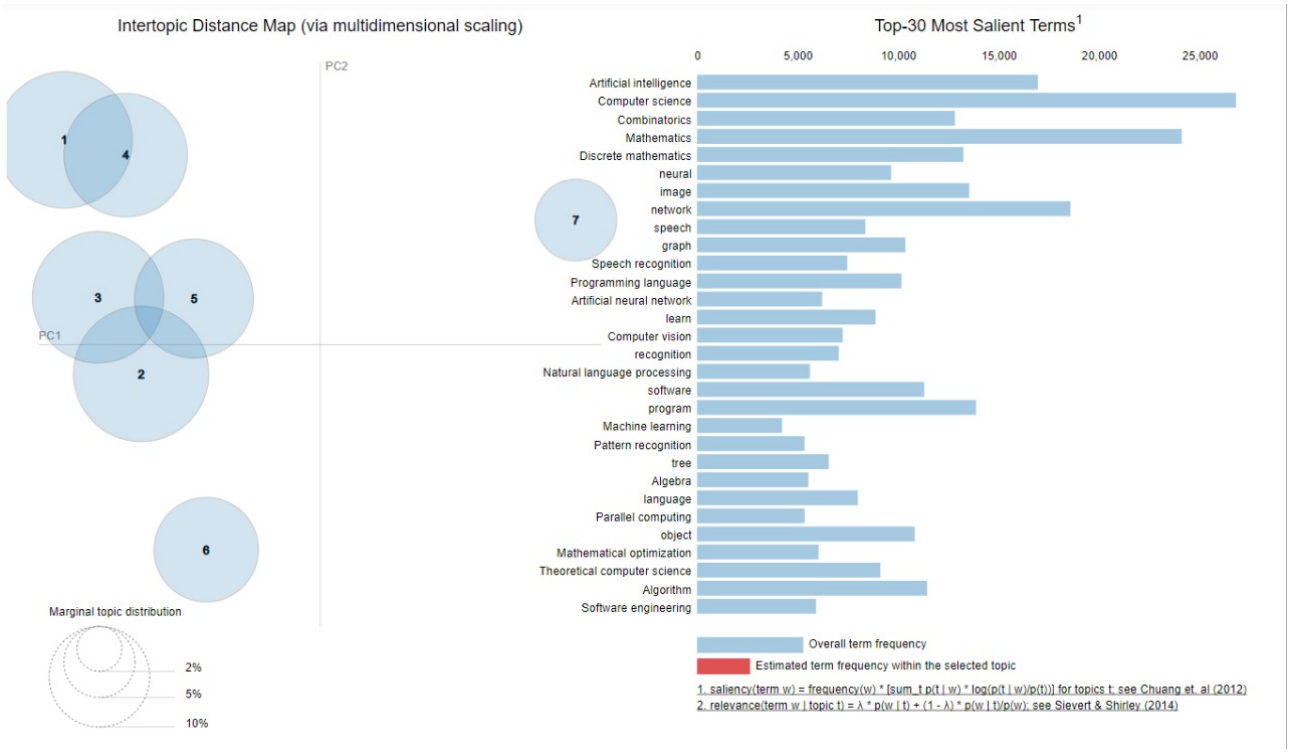

Figure 3.6: 1982-2021 trending research topics

3.3 Analysis of Phd Thesis data

For analysing phd thesis data we extracted the data from dblp.xml file provided by the website. The size of data file dblp.xml is over 3.5GB.Hence instead of parsing the data file in main memory file operations are extensively used.Topics on which analysis is done are listed below:

Example of raw form of data in dblp.xml:

```
<phdthesis mdate="2021-07-17" key="series/faia/2005-148">
<author>Kristian Kersting</author>
<title>An Inductive Logic Programming Approach to
Statistical Relational Learning</title>
<year>2005</year>
<pages>1-228</pages>
<publisher>IOS Press</publisher>
<series href="db/series/faia/index.html">Frontiers in
Artificial Intelligence and Applications</series>
<volume>148</volume>
<school>University of Freiburg, Germany</school>
<isbn>978-1-58603-674-4</isbn>
<ee>http://www.booksonline.iospress.nl/Content/View.aspx?piid=96</ee>
<ee>https://d-nb.info/983957975</ee>
</phdthesis>
```

- Clustering of phd thesis titles using k-means algorithm.
- Top 15 universities which have maximum phd thesis data in dblp.xml.
- Count of phd thesis decade-wise.

3.3.1 Clustering of phd thesis titles using K-means clustering algorithm

As a part of data extraction, title of phd thesis is taken. Preprocessing steps:

1. Excluding thesis titles of other languages such as German, French etc.polyglot python library is used. Eg. Invarianz der Parsingeigenschaft unter Grammatikmorphismen. (German) is filtered and not considered for analysis
2. TF-IDF is used to convert thesis title to vector representation
3. K-means clustering algorithm [6] is applied on the vectorized representation of thesis titles
    - We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the kMeans algorithm[7]; an unsupervised learning algorithm.
      (It will help if you think of items as points in an n-dimensional space). The algorithm will categorize the items into k groups of similarity. To calculate that similarity, we will use the euclidean distance as measurement. The algorithm works as follows:
      1.First, we initialize K points, called means, randomly.

      2.We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that mean so far.

      3.We repeat the process for a given number of iterations and at the end, we have our clusters.
4. Results:
    - Optimal value of K is found by elbow method:

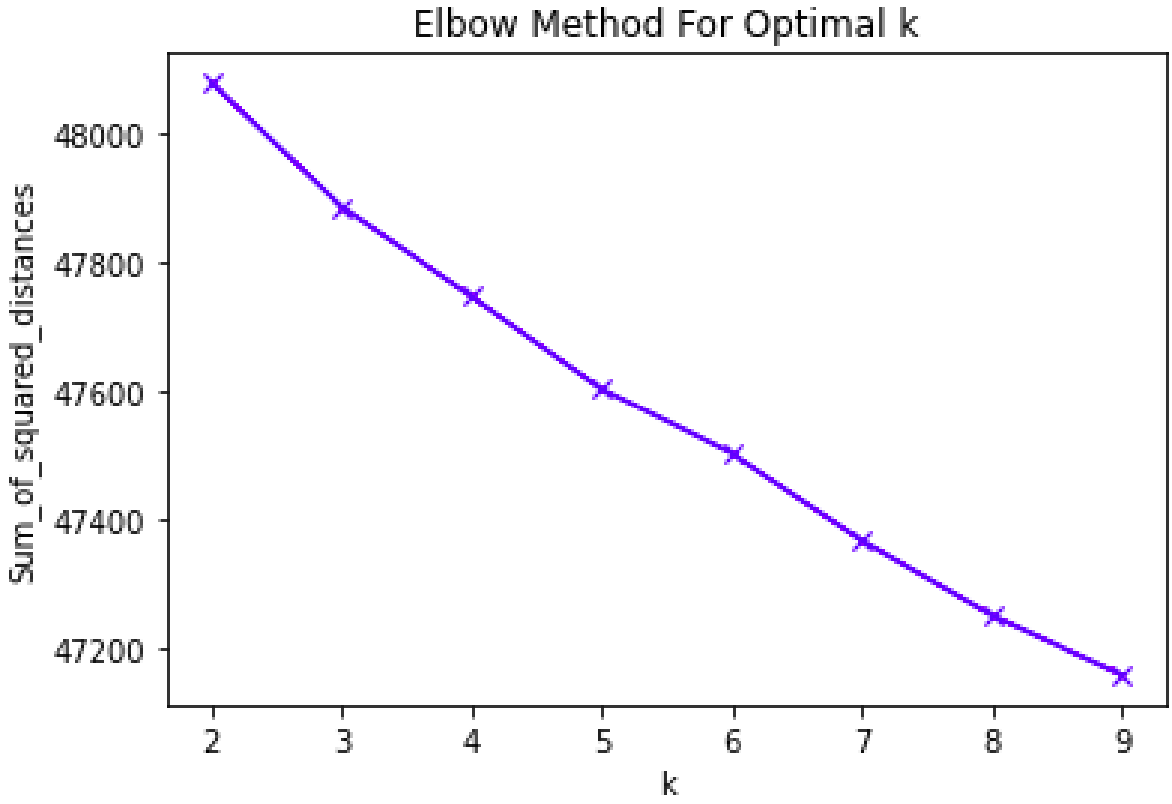

Figure 3.7: Elbow method for optimal k

- Word cloud of clusters

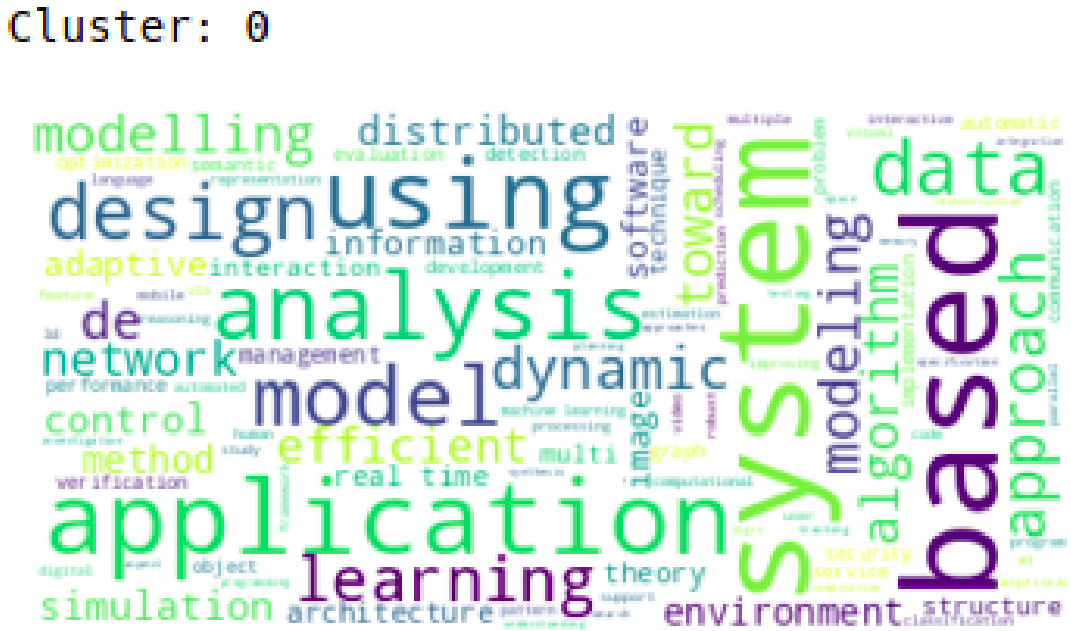

Figure 3.8

Cluster: 1

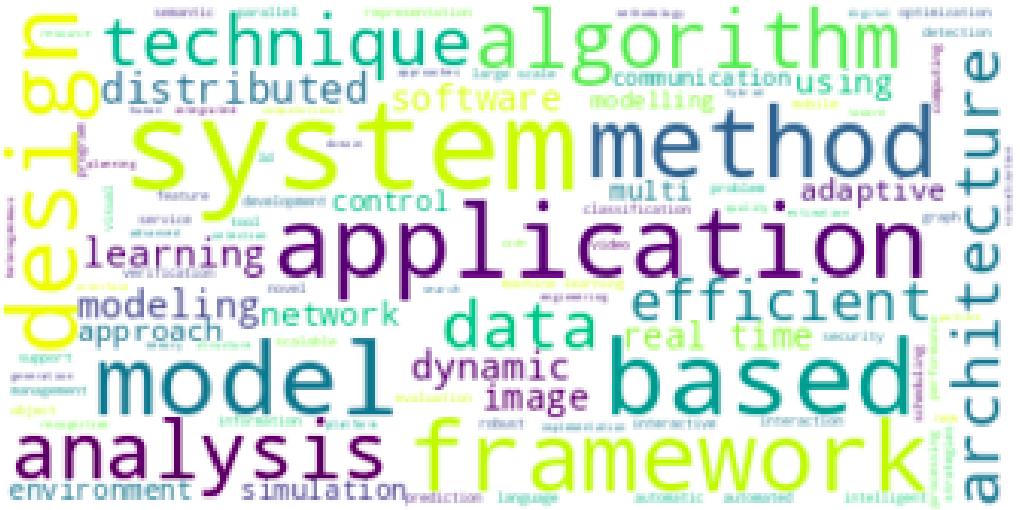

Figure 3.9

Cluster: 2

Figure 3.10

Cluster: 3

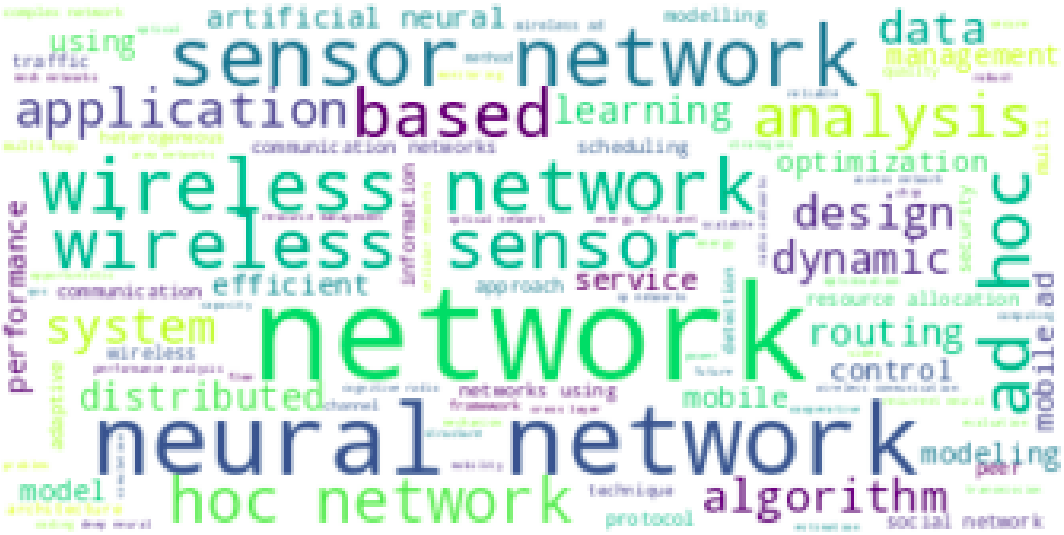

Figure 3.11

- Some of the thesis titles assigned to clusters

Cluster 0

1. An Inductive Logic Programming Approach to Statistical Relational Learning
2. Neural Generation of Textual Summaries from Knowledge Base Triples
3. Relationships between assumptions.
4. Multimedia information system with automatic content retrieval.
5. Projection-based strictness analysis - theoretical and practical aspects

Figure 3.12

Cluster 1

1. Semantic Methods for Execution-level Business Process Modeling - Modeling Support Through Process Verification and Service Composition
2. Markov localization - a probabilistic framework for mobile robot localization and navigation.
3. Dynamic Code Evolution for Java.
4. Hardware and software mechanisms for reducing load latency.
5. Instruction-processing optimization techniques for VLSI microprocessors.

Figure 3.13

Cluster 2

1. The complexity of simple subtyping systems.
2. The support feature machine - an odyssey in high-dimensional spaces.
3. On the representation of mathematical concepts and their translation into first-order logic.
4. The Asynchronous Graz Brain Switch.
5. The temporal properties of english conditionals and modals.

Figure 3.14

Cluster 3

1. An anytime approach to connectionist theory refinement - refining the topologies of knowledge-based neural networks.
2. Relatedness in evidence networks.
3. Performance management in ATM networks.
4. Topology preserving neural networks - connectionist learning of structured knowledge.
5. Handling realtime traffic in mobile networks.

Figure 3.15

3.3.2 Top 15 universities which have maximum phd thesis data

Based on the data available in dblp.xml top 15 univerisities which have maximum phd data is enumerated. The results are as follows:

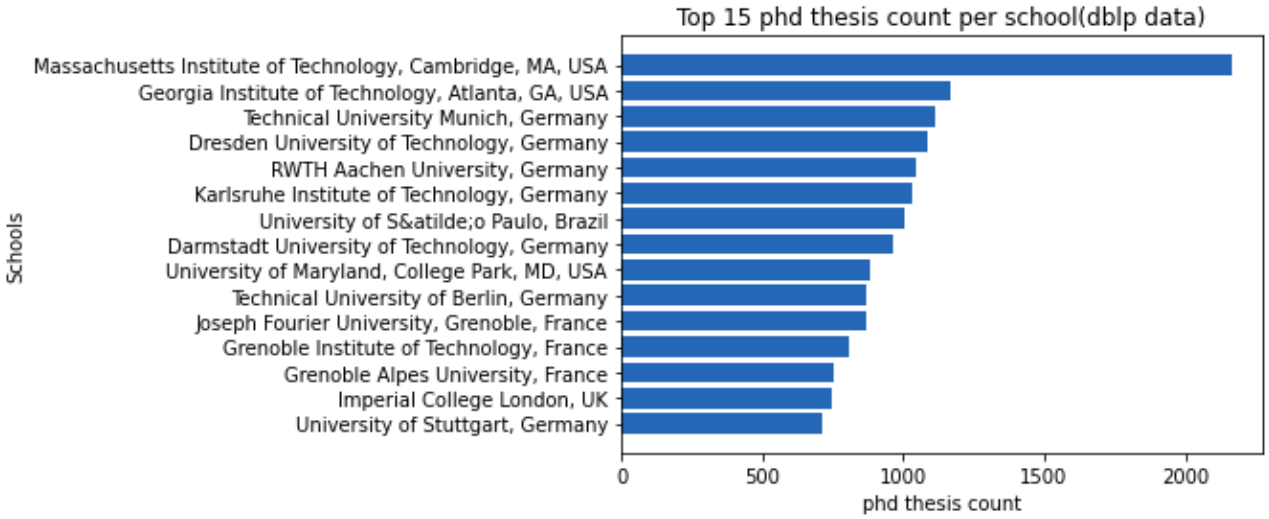

Figure 3.16: Top 15 phd thesis count per school

3.3.3 Count of phd thesis decade-wise.

Based on the data available in dblp.xml top 15 univerisities which have maximum phd data is enumerated. The results are as follows:

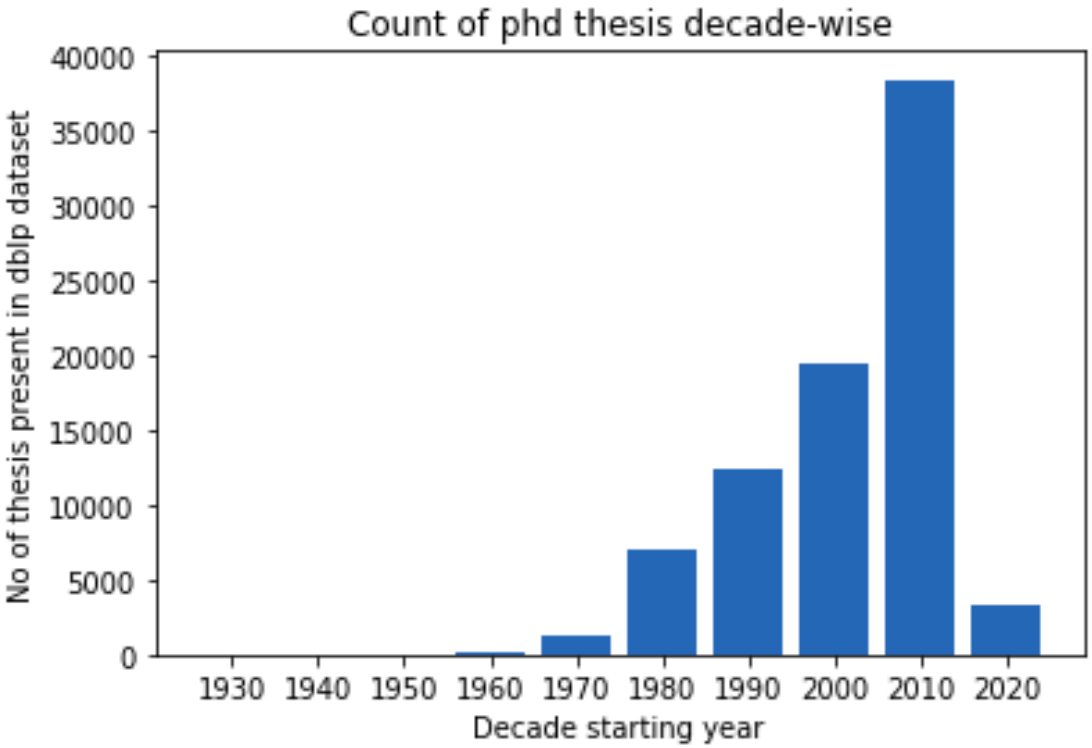

Figure 3.17: Count of phd thesis decade-wise

3.4 Analysis of DBLP publication titles and authors

We make use of the dblp data which consists of information about more than 5 million publications.

3.4.1 Methodology

We first calculate the frequency of words in the titles of the publications. Then, we create a wordcloud for the top 100 most frequent words. WordClouds are the visual representations of words that give more prominence to the words that appear frequently.

We also calculate the count of number of publications by authors and create wordcloud for the top 100 authors with the greatest number of publications.

We also use bar plots to visualise the year wise frequency distribution of words that occur in the titles of the publications. We create a file named "words_years.csv" and save the year wise distribution of words in it.

We also use bar plots to visualise the year wise publication count distribution of authors. We create a file named "authors_years.csv" and save the year wise publication count of authors in it. We consider the years 1970 to 2021.

3.4.2 Results

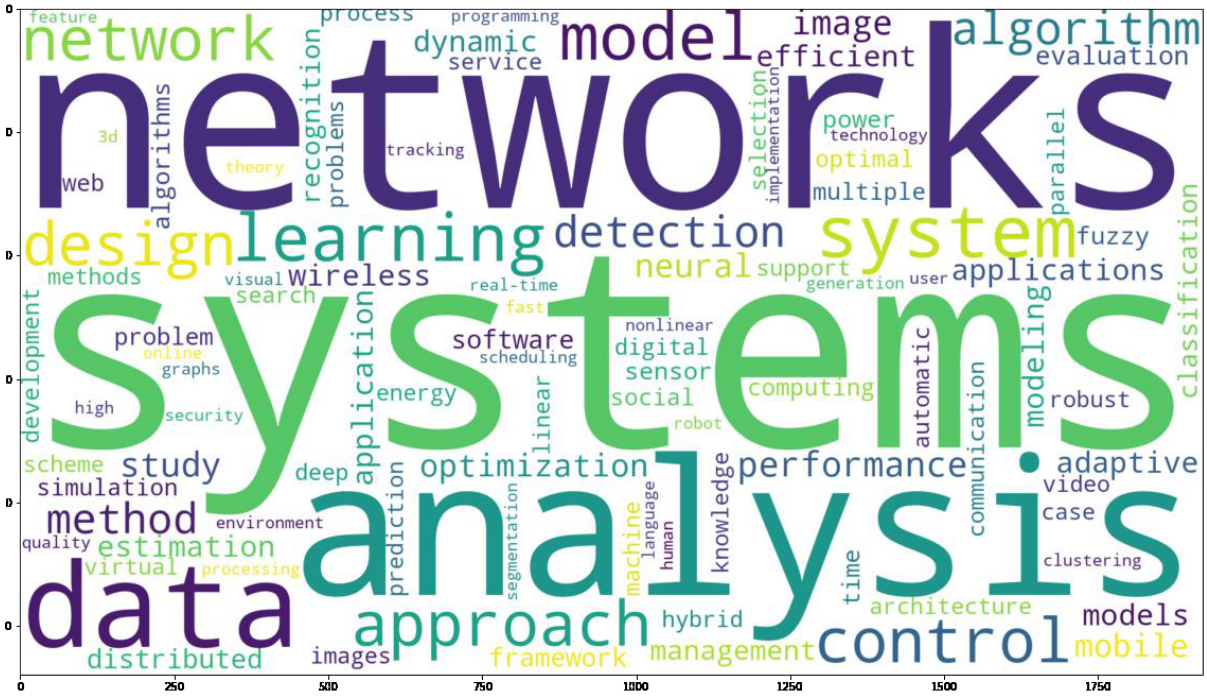

Figure 3.18: word cloud for title words

The above figure is the word cloud for title words. From the above figure, we can see that the words networks and systems are among the most frequently occurring words in the titles of the publications.

Figure 3.19: word cloud for authors

The above figure is the word cloud for authors. From the above figure, we can see that the authors Wei Wang and Yang Liu are among the authors with most number of publications.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	word	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	199
2	relationship	4	4	5	6	6	8	11	5	11	8	13	8	12	14	19	12	23	25	15	29	2
3	canopy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
4	parameters	5	9	6	13	12	8	12	10	10	22	19	15	27	18	31	28	43	41	37	67	7
5	spectrum	1	1	3	5	6	5	5	22	17	14	16	17	17	19	33	29	44	38	30	46	3
6	winter	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	2	1	2	
7	wheat	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	
8	irrigations	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	hebei	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	province	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
11	solution	35	35	30	37	49	44	40	40	34	46	46	48	48	47	55	66	81	79	101	128	13
12	problem	49	60	65	102	122	118	103	150	140	128	173	181	189	206	202	235	231	301	317	371	44

Figure 3.20

We create a file named “words_years.csv” and save the year wise distribution of words in it with columns word,1970,1971……..2021 as shown above.

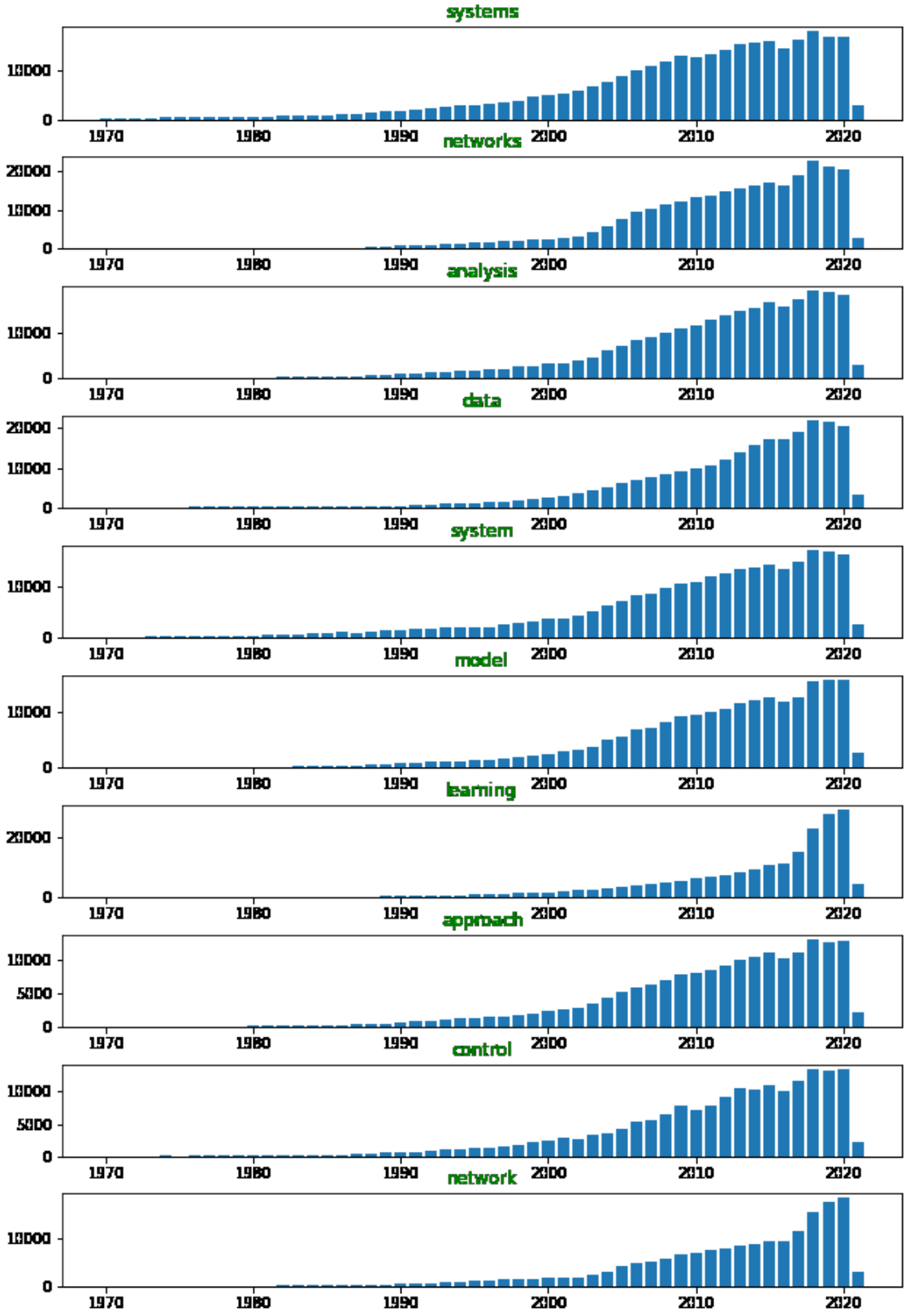

Figure 3.21

The above image shows the year wise frequency distribution of few words that occur in the

titles of the publications.

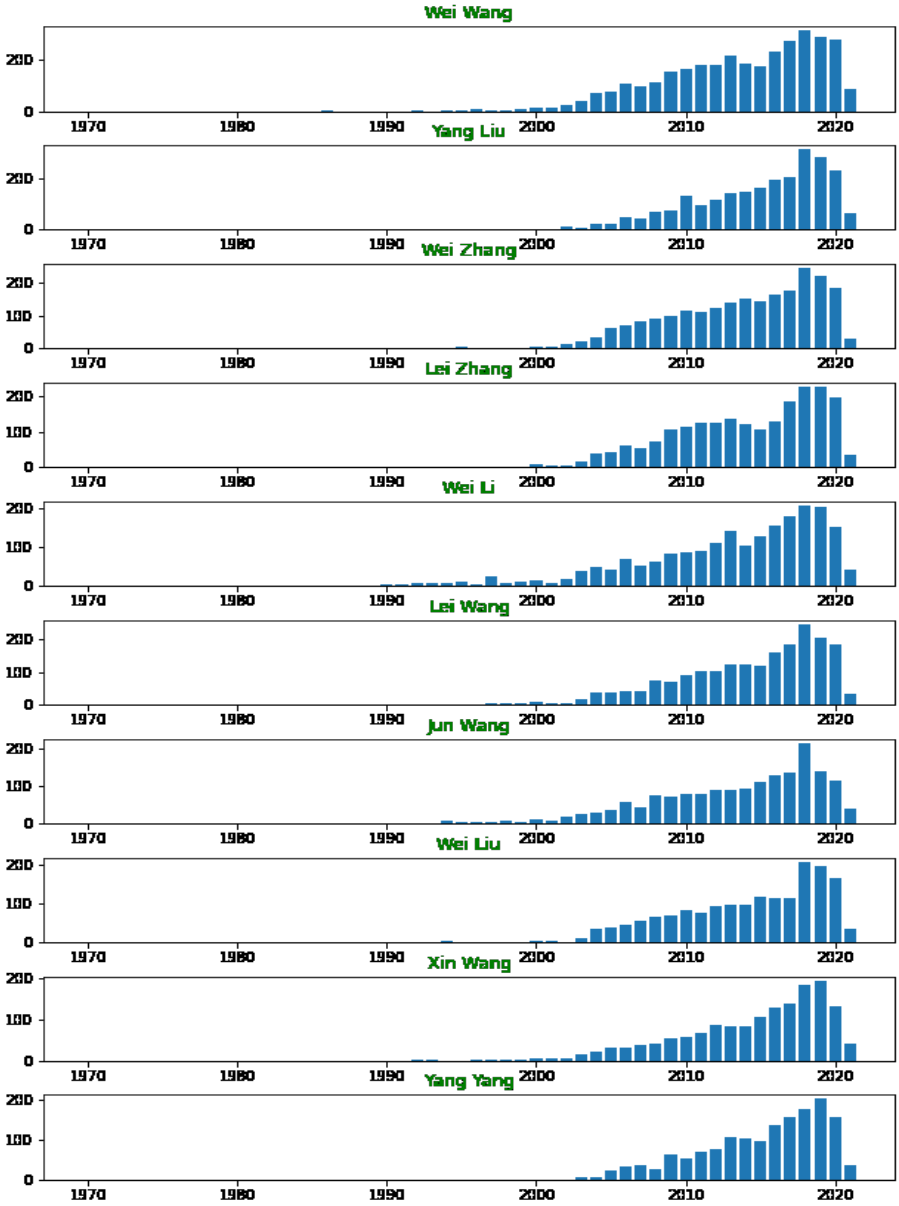

Figure 3.22

The above figure shows the year wise publication count distribution of few authors.

3.5 Analysis and Prediction of Co-Authors

We make use of the dblp data which consists of information about more than 5 million publications.

3.5.1 Methodology

We first find out the authors who have worked together in their publications from the dblp data. Then, we make a graph or network of the authors that have worked together using an adjacency list.

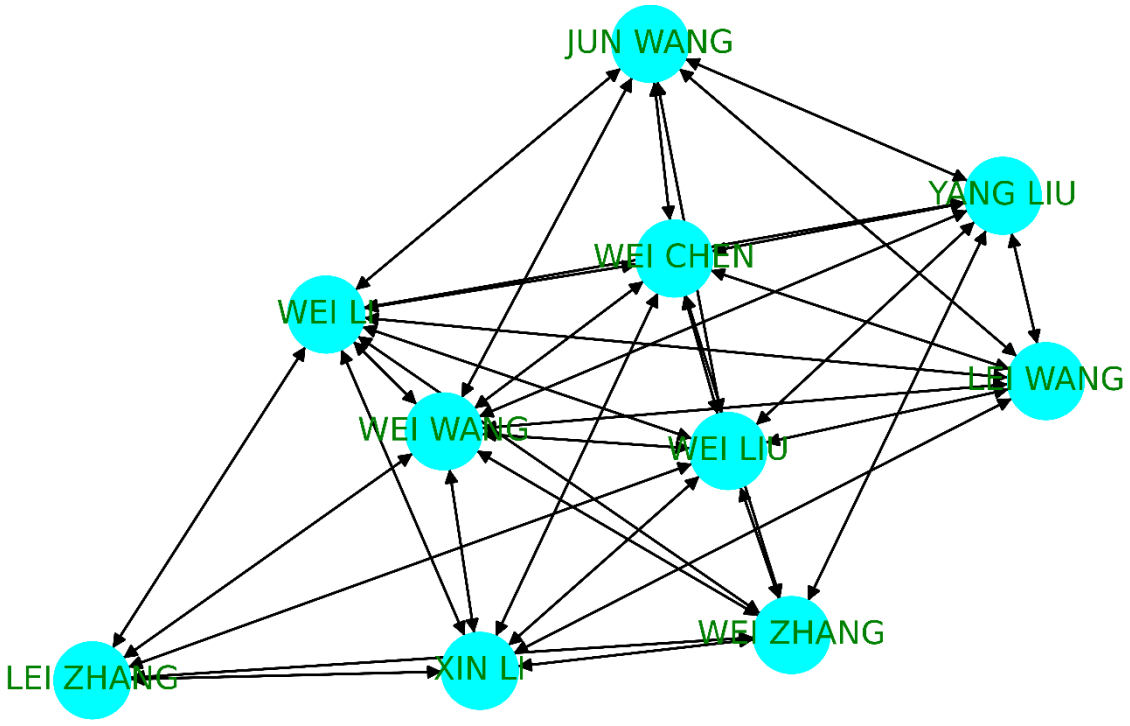

Figure 3.23

For example, in the above figure, we have represented the network of 10 authors who have worked together in their publications. We have used networkx library in python for the creation of the above figure. Each node represents the author. Each edge between 2 nodes represent that these authors have worked together previously.

After the construction of the adjacency list, we will then use certain link prediction techniques in order to predict the possibility of co-authors working together in the future for any particular author who have not worked together in the past.

We have used 2 link prediction techniques. They are:

1) Jaccard Coefficient

2) Adamic Adar Coefficient

Jaccard Coefficient

The Jaccard Coefficient, also known as Jaccard index or Jaccard similarity coefficient, is a statistic measure used for comparing similarity of sample sets. It is usually denoted as J(x,y)

where x and y represent two different nodes in a network. In link prediction, all the neighbours of a node are treated as a set and the prediction is done by computing and ranking the similarity of the neighbour set of each node pair. The Jaccard Coefficient is in the range of 0 and 1. The higher the Jaccard Coefficient, the better chance for the authors to collaborate in the future.

The formula for Jaccard Coefficient is

$$\mathrm{J}(A,B) = \frac{\left|A \cap B\right|}{\left|A \cap B\right|}$$

Figure 3.24

Adamic/Adar Coefficient

The Adamic/Adar index is a measure introduced in 2003 by Lada Adamic and Eytan Adar to predict links in a social network, according to the amount of shared links between two nodes. It is defined as the sum of the inverse logarithmic degree centrality of the neighbours shared by the two nodes.

$$A(x,y) \sum_{u \in N(x) \cap N(y)} \frac{1}{\log \left|N(u)\right.}$$

Figure 3.25

Here, N is the function which returns all the neighbour nodes of any input node. The definition is based on the concept that common elements with very large neighbourhoods are less significant when predicting a connection between two nodes compared with elements shared between a small number of nodes. The higher the Adamic/Adar Coefficient, the better chance for the authors to collaborate in the future.

For predicting, we consider any particular author as input and for that author, the neighbours of the neighbouring authors to the input author will be considered as the possible future coauthors. Then, the Jaccard index and Adamic/Adar Index will be calculated. Then, we output at most 5 possible authors with the highest Jaccard Index and Adamic/Adar Index separately.

We have done this for 10000 randomly selected authors. We will then save the results in two csv files named “jaccard_predictions.csv” and “aac_predictions.csv”.

```
from math import log
def jacard(x,y):
    return len(nei[x].intersection(nei[y]))/len(nei[x].union(nei[y]))
def aac(x,y):
    return sum(1/log(len(nei[z]),10) for z in nei[x].intersection(nei[y]))
def predict(x):
    res1=set()
    res2=set()
    for y in nei[x]:
        for z in nei[y]:
            if z not in nei[x] and z!=x:

                res1.add((jacard(x,z),z))
                res2.add((aac(x,z),z))
    res1=list(res1)
    res2=list(res2)
    res1.sort(reverse=True)
    res2.sort(reverse=True)
    return res1[:min(5,len(res1))],res2[:min(5,len(res2))]
```

Figure 3.26

The jacard function is used to calculate the jaccard index for nodes x and y. The aac function is used to calculate the Adamic/Adar Index for the nodes x and y. The predict function takes x as input and it gives out the outputs according to jaccard index as well as aac.

3.5.2 Results

We predict the co-authors for 10000 randomly selected authors. We will then save the results in two csv files named "jaccard_predictions.csv" and "aac_predictions.csv".

	author	co_author	jac
2	PEKKA SUTELA	TEEMU MYLLYLA	0.090909091
3	PEKKA SUTELA	TUIJA HILTUNEN	0.071428571
4	PEKKA SUTELA	JUHA NIKKINEN	0.045454545
5	PEKKA SUTELA	TEEMU MYLLYLÃ„	0.041666667
6	PEKKA SUTELA	PIERRE LEVAN	0.041666667
7	HYONG UK MOON	YONGKYUN CHO	0.083333333
8	HYONG UK MOON	JAE WOOK JEAN	0.076923077
9	HYONG UK MOON	JINYEONG MOON	0.071428571
10	HYONG UK MOON	CHONG HO YOON	0.071428571
11	HYONG UK MOON	CHAE HONG YI	0.071428571
12	CATHY H. XIA	CATHY XIA	0.131578947
13	CATHY H. XIA	JAMES R. CHALLENGER	0.081081081
14	CATHY H. XIA	MEISAM FATHI	0.078947368
15	CATHY H. XIA	LIANG GENG	0.076923077
16	CATHY H. XIA	MICHEL HACK	0.071428571
17	TAKESHI KIMURA	KYOICHI SAITO	0.090909091
18	TAKESHI KIMURA	KAZUTOMO SHIBAHARA	0.076923077
19	TAKESHI KIMURA	KAZUYOSHI SHOGEN	0.075
20	TAKESHI KIMURA	HISAYUKI OHMATA	0.072727273
21	TAKESHI KIMURA	TOMOYUKI INOUE	0.069767442
22	ASHUTOSH KUMAR	TUSHANT JHA	0.046153846
23	ASHUTOSH KUMAR	B. D. CHAUDHARY	0.043478261
24	ASHUTOSH KUMAR	BANSHI DHAR CHAUDHARY	0.042253521
25	ASHUTOSH KUMAR	DEAN DORON	0.04
26	ASHUTOSH KUMAR	SHACHAR LOVETT	0.033333333
27	MITALI DESAI	BHAVISHA S. PARMAR	1
28	MITALI DESAI	DEVESH C. JINWALA	0.04
29	PEYMAN TAVALLALI	WALID MOSTAFA	0.111111111
30	PEYMAN TAVALLALI	RENDONG BAI	0.111111111

jacard_predictions

Figure 3.27: Jaccard predictions

"jaccard_predictions.csv" will look like the above figure. First column is the input author for whom we have to predict the co-author. The second column is the predicted co-author. The third column is the jaccard Coefficient.

	1	2	3	4
1	author	co_author	aac	
2	PEKKA SUTELA	YUFENG ZANG	0.812711509	
3	PEKKA SUTELA	YU-FENG ZANG	0.812711509	
4	PEKKA SUTELA	XINDI WANG	0.812711509	
5	PEKKA SUTELA	X WANG	0.812711509	
6	PEKKA SUTELA	VILLE ISOMÃ–TTÃ–NEN	0.812711509	
7	HYONG UK MOON	KYUNG WHAN KIM	1.252289107	
8	HYONG UK MOON	JAEIL KIM	1.252289107	
9	HYONG UK MOON	YONGKYUN CHO	0.79663977	
10	HYONG UK MOON	YEO KEUN KIM	0.79663977	
11	HYONG UK MOON	JIN HO KIM	0.79663977	
12	CATHY H. XIA	HAO WANG	3.701192597	
13	CATHY H. XIA	CATHY XIA	2.86112188	
14	CATHY H. XIA	YANG LIU	2.675388527	
15	CATHY H. XIA	PHILIP S. YU	2.663923831	
16	CATHY H. XIA	JIAN TAN	2.588619758	
17	TAKESHI KIMURA	HISAYUKI OHMATA	3.307131626	
18	TAKESHI KIMURA	KYOICHI SAITO	2.957644089	
19	TAKESHI KIMURA	YOSUKE ENDO	2.738961173	
20	TAKESHI KIMURA	HIROYUKI IMAIZUMI	2.516502856	
21	TAKESHI KIMURA	TOMOYUKI INOUE	2.309800675	
22	ASHUTOSH KUMAR	NARESH KUMAR NAGWANI	3.321928095	
23	ASHUTOSH KUMAR	TUSHANT JHA	3.061961742	
24	ASHUTOSH KUMAR	GEETIKA SHARMA	2.514770771	
25	ASHUTOSH KUMAR	SHACHAR LOVETT	2.16181239	
26	ASHUTOSH KUMAR	RAJANI KUMARI	2.084382862	
27	MITALI DESAI	DEVESH C. JINWALA	2.095903274	
28	MITALI DESAI	BHAVISHA S. PARMAR	2.095903274	
29	PEYMAN TAVALLALI	MARIKO S. BURGIN	2.227316328	
30	PEYMAN TAVALLALI	JAKOB J. VAN ZYL	2.227316328	

aac_predictions

Figure 3.28: aac predictions

"aac_predictions.csv" will look like the above figure. First column is the input author for whom we have to predict the co-author. The second column is the predicted co-author. The third column is the adamic/adar Coefficient.

3.6 Association Rule Mining

We make use of the dblp data which consists of information about more than 5 million publications. We perform the association rule mining of words that occur in the publication title.

3.6.1 Methodology

Association rules are "if-then" statements. They help us to show the probability of relationships between data items. They are useful in fining correlations and co-occurrences between data sets and also can be used to find the patterns in data. Association rule has two parts. They are antecedent(if) and consequent(then).

There are few important terms we need to understand while performing Association Mining. They are support, confidence.

Support

Support indicates the frequency of occurrence of an item in the dataset divided by the total transactions. We use the parameter min support in Association Mining. We consider the items who have a frequency greater than or equal to the min support. For example, if the rule is X=>Y, then the Support = Frequency(X,Y)/N where N is total number of transactions.

Confidence

Confidence indicates the conditional probability of the consequent given the antecedent.

For example, if the rule is X=>Y, then the Confidence = Frequency(X,Y)/Frequency(X)

We have used FP-Growth algorithm in order to generate the Association Rules.

FP-Growth

FP-Growth is a popular algorithm used for the mining of association rules. It is very fast when compared to Apriori Algorithm. FP-Growth algorithm organizes the data into a tree like structure in order for faster scanning.

We consider each publication title as a transaction. The words in the title are considered as the items.

The steps under this FP-Growth algorithm are:

1) We count the frequencies of individual items.

2) We assign a minimum support. We calculated it in the project as the min support = 100/(total transactions). Every item with fewer occurrences than the min support will be excluded.

3) We order the non-excluded words in each transaction in descending order of their frequencies. These are called ordered item sets.

4) We have a root node in the tree which is NULL. We create the tree by inserting all the ordered item sets.

5) After construction of the tree, the Conditional Pattern Base which is the path of the nodes which leads to any particular node.

6) We find the Conditional Frequent Pattern tree for each item. We do it by considering the set of common elements in the paths in Conditional Pattern Base and we calculate support count and take the ones with at least min support.

7) We now generate the frequent pattern rules by pairing items of conditional frequent pattern tree with its corresponding item.

8) We will assign a min confidence parameter. We set min confidence as 0.7 in the project. We now generate strong association rules from the frequent patterns. For example, if Y is a

frequent pattern set, then we generate all possible non empty subsets from Y and for each subset X, we calculate the confidence as confidence=support(Y-X)/support(X) for the rule X=>(Y-X). We consider only the rules with confidence >= min confidence.

In [3]:
```
from collections import defaultdict as dt
class node:
    def __init__(self,v=None):
        self.val=v
        self.child=dt(node)
        self.count=0
        self.par=None
```

In [4]:
```
def tree(root,x,i):
    if i>=len(x):
        return
    zz=root.child[x[i]]
    zz.count+=1
    zz.val=x[i]
    zz.par=root
    node_dict[x[i]].append(zz)
    tree(zz,x,i+1)
```

In [5]:
```
node_dict=dt(list)
root=node()
for i in data:
    tree(root,i,0)
```

In [6]:
```
def traverse(root,z):
    res=dt(int)
    wd=set()
    for a in root.child:
        z.append(a)
        qq=root.child[a]
        di=traverse(qq,z)
        del z[-1]
        if not z:continue
        for b in di:
            res[b]+=di[b]
            wd.add(b)
    if not z:return
    for i in wd:
        if res[i]/l>=min_support:
            patterns[i].append((tuple(z),res[i]))
            res[i]=0
    if root.val!=None:
        res[root.val]+=root.count
    return res
```

In [7]:
```
patterns=dt(list)
traverse(root,[])
```

In [10]:
```
def support(x):
    word = min((len(ind[i]),i) for i in x)[1]
    c=0
    for j in node_dict[word]:
        zz=j.count
        qq=0
        while j and qq!=len(x):
            if j.val in x:qq+=1
            j=j.par
        if qq==len(x):
            c+=zz
    return c
```

In [11]:
```
res=set()
def func(a,b,i,x,r):
    if i>=len(x):
        if a and b:
            supp_a=support(a)
            if supp_a==0:return
            conf = support(a.union(b))/supp_a
            if conf>=0.7:
                s="("
                for j in sorted(a):
                    s+=j+" ; "
                s+="), ("
                for j in sorted(b):
                    s+=j+" ; "
                s+="), "+str(conf)+"\n"
                res.add(s)
        return
    a.add(x[i])
    func(a,b,i+1,x,r)
    a.remove(x[i])
    b.add(x[i])
    func(a,b,i+1,x,r)
    b.remove(x[i])
    func(a,b,i+1,x,r)
```

In [12]:
```
d=0
for b in patterns:
    for a,c in patterns[b]:

        func(set(),{b},0,list(a),c)
        func({b},set(),0,list(a),c)
    d+=1
    print(d,end='\r')
```

Figure 3.29

We use the above code on the left to create the tree by insert the ordered item sets. Then, we generate the frequent patterns from that tree.

We use the above code on the right to then obtain the strong association rules from the frequent patterns.

3.6.2 Results

	antecedent	consequent	confidence
2	(worst ;)	(case ;)	0.745233969
3	(wavelength-routed ;)	(networks ;)	0.873239437
4	(warehousing ;)	(data ;)	0.87260274
5	(warehouses ;)	(data ;)	0.853416149
6	(warehouse ;)	(data ;)	0.70849528
7	(unicyclic ;)	(graphs ;)	0.840764331
8	(underwater ; wireless ;)	(networks ;)	0.736842105
9	(underlaying ;)	(networks ;)	0.800970874
10	(ultra-dense ;)	(networks ;)	0.715231788
11	(type-2 ;)	(fuzzy ;)	0.875609756
12	(trust ; wireless ;)	(networks ;)	0.815668203
13	(triangle-free ;)	(graphs ;)	0.8
14	(tree ; wireless ;)	(networks ;)	0.791666667
15	(tracking ; wireless ;)	(sensor ;)	0.725968436
16	(topology ; wireless ;)	(networks ;)	0.854195323
17	(time ; warping ;)	(dynamic ;)	0.783783784
18	(things ;)	(internet ;)	0.715523834
19	(tdma ; wireless ;)	(networks ;)	0.752380952
20	(target ; wireless ;)	(sensor ;)	0.944812362
21	(target ; wireless ;)	(networks ; sensor ;)	0.836644592
22	(target ; wireless ;)	(networks ;)	0.858719647
23	(systems ; tutoring ;)	(intelligent ;)	0.842205323
24	(synchronization ; wireless ;)	(networks ;)	0.703488372
25	(swarm ;)	(particle ;)	0.708597604
26	(survey ; wireless ;)	(networks ;)	0.748829953
27	(supervisory ; systems ;)	(control ;)	0.864353312
28	(supervisory ;)	(control ;)	0.810839532
29	(strict-feedback ;)	(systems ;)	0.957627119
30	(strategies ; wireless ;)	(networks ;)	0.712082262

association_rules

Figure 3.30: Association Rules

The association rules will be generated as an output in "association_rules.csv" file. There will be 3 columns antecedent, consequent and confidence. More than 2000 association rules were generated with min support of 100/(total transactions) and min confidence of 0.7.

3.7 Correlations

We make use of the dblp data which consists of information about more than 5 million publications. We find the Phi correlation coefficient between the words in the titles of the publications. Also, we find the Phi correlation coefficient between the authors of the publications too. We use seaborn library to plot the heatmap of correlation matrix.

3.7.1 Methodology

We first calculate the frequencies of title words. We will then consider the top 100 frequent words for finding the correlation between them.

	Has word Y	No word Y	Total
Has word X	n_{11}	n_{10}	$n_{1\cdot}$
No word X	n_{01}	n_{00}	$n_{0\cdot}$
Total	$n_{\cdot 1}$	$n_{\cdot 0}$	n

$$\phi = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1\cdot}n_{0\cdot}n_{\cdot 1}n_{\cdot 0}}}$$

Figure 3.31

Phi correlation coefficient is used for the measure of association between binary variables. Let us consider two words X and Y. Here, "Has word X" and "No word X" is a dichotomous variable. "Has word Y" and "No word Y" is also dichotomous variable.

Here, n11 means the number of titles that have both X and Y in them. n10 means the number of titles that have word X and no word Y. n01 means the number of titles that have word Y and no word X. n00 means the number of titles that don't have both X and Y.

By using the above formula, we then calculate the Phi correlation coefficient. We calculate these coefficients between all possible pairs of the top 100 frequent words. After calculating this coefficient, we then save these results in "title_words_correlation.csv" file. For the purpose of plotting the correlation matrix, we set the diagonal elements of the matrix to zero for better visualization of the correlation matrix. We use seaborn library to plot the heatmap of the correlation matrix.

We repeat the above process for the authors too.

In [1]:
```
from collections import defaultdict as dt
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
def plot_correlation_matrix(data,count,title,color):
    freq={}
    for i in data:
        for j in i:
            if j in freq:
                freq[j]+=1
            else:
                freq[j]=1
    words=[j for j,k in sorted(freq.items(),key=lambda i:i[1],reverse=True)[:count]]
    words_set=set(words)
    corr_mat=dt(int)
    for i in data:
        for j in range(len(i)):
            for k in range(len(i)):
                if i[j] in words_set and i[k] in words_set:
                    corr_mat[i[j],i[k]]+=1
    mat=[]
    for i in range(len(words)):
        mat.append([])
        for j in range(len(words)):
            if i==j:
                mat[-1].append(1)
                continue
            a,b=words[i],words[j]
            n11=corr_mat[a,b]
            n10=freq[a]-n11
            n01=freq[b]-n11
            n00=len(data)-(n01+n10+n11)
            z=((n11+n10)*(n11+n01)*(n00+n01)*(n00+n10))**.5

            mat[-1].append((n11*n00 - n10*n01)/z)
    if title[0]=='T':
        file=open("title_words_correlation.csv",'w',encoding="utf-8")
        file.write("word1,word2,correlation\n")
    else:
        file=open("authors_correlation.csv",'w',encoding="utf-8")
        file.write("author1,author2,correlation\n")
    for i in range(len(mat)):
        for j in range(len(mat)):
            file.write(words[i]+","+words[j]+","+str(mat[i][j])+"\n")
    for i in range(len(words)):
        mat[i][i]=0
    df=pd.DataFrame(mat)
    df.index=words
    df.columns=words
    plt.figure(figsize = (20,15))
    plt.title(title)
    fig=sn.heatmap(df,cmap=color)
    fig.get_figure().savefig(title+".png")
```

Figure 3.32

We use the above code for calculation of coefficients, plotting and saving the csv files.

3.7.2 Results

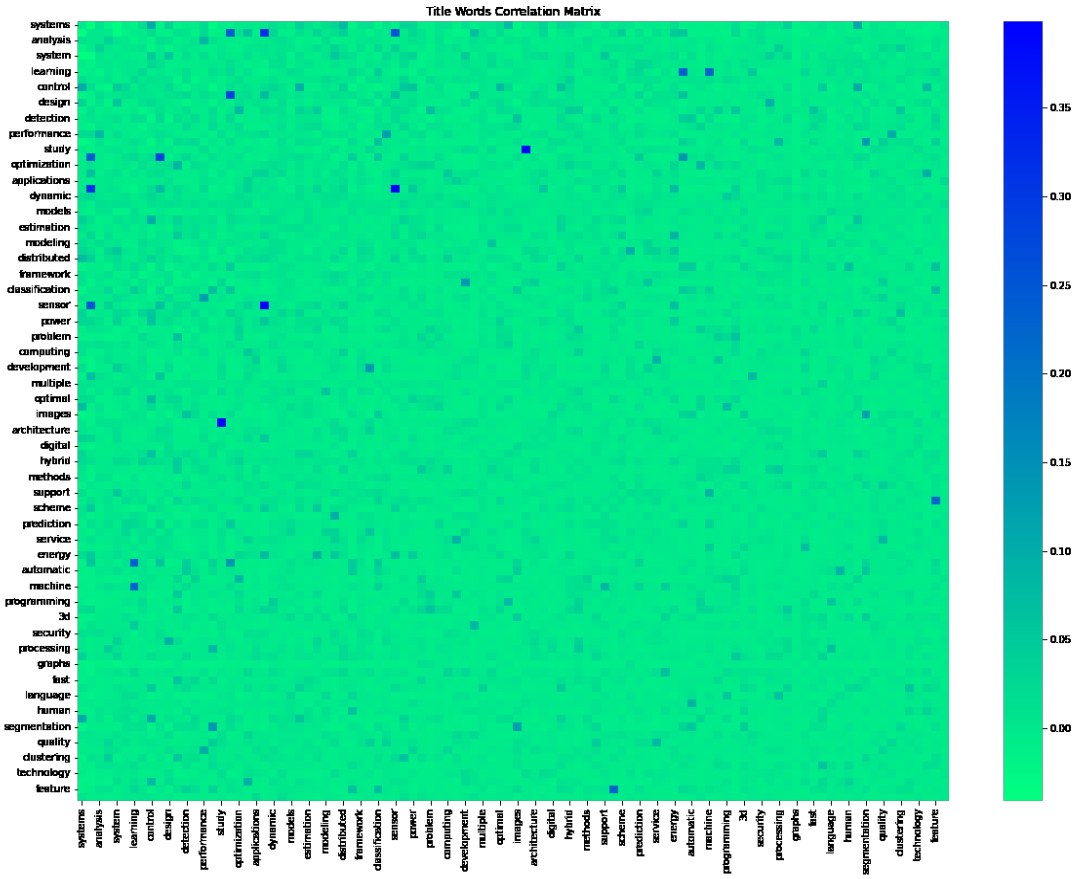

Figure 3.33: Heatmap for the correlation matrix of Title Words

The above figure is the heatmap for the correlation matrix of Title Words. From the heatmap, we can infer that the word pairs like (wireless, sensor), (wireless, network), (deep, learning), (neural, network) etc are correlated strongly when compared to other pairs.

	word1	word2	correlation
2	systems	systems	1
3	systems	networks	-0.04107
4	systems	analysis	0.007395
5	systems	data	-0.02173
6	systems	system	-0.03818
7	systems	model	-0.0073
8	systems	learning	-0.01998
9	systems	approach	0.01584
10	systems	control	0.104539
11	systems	network	-0.02299

Figure 3.34

We also save the correlations in "title_words_correlation.csv" file with three columns namely word1, word2, correlation.

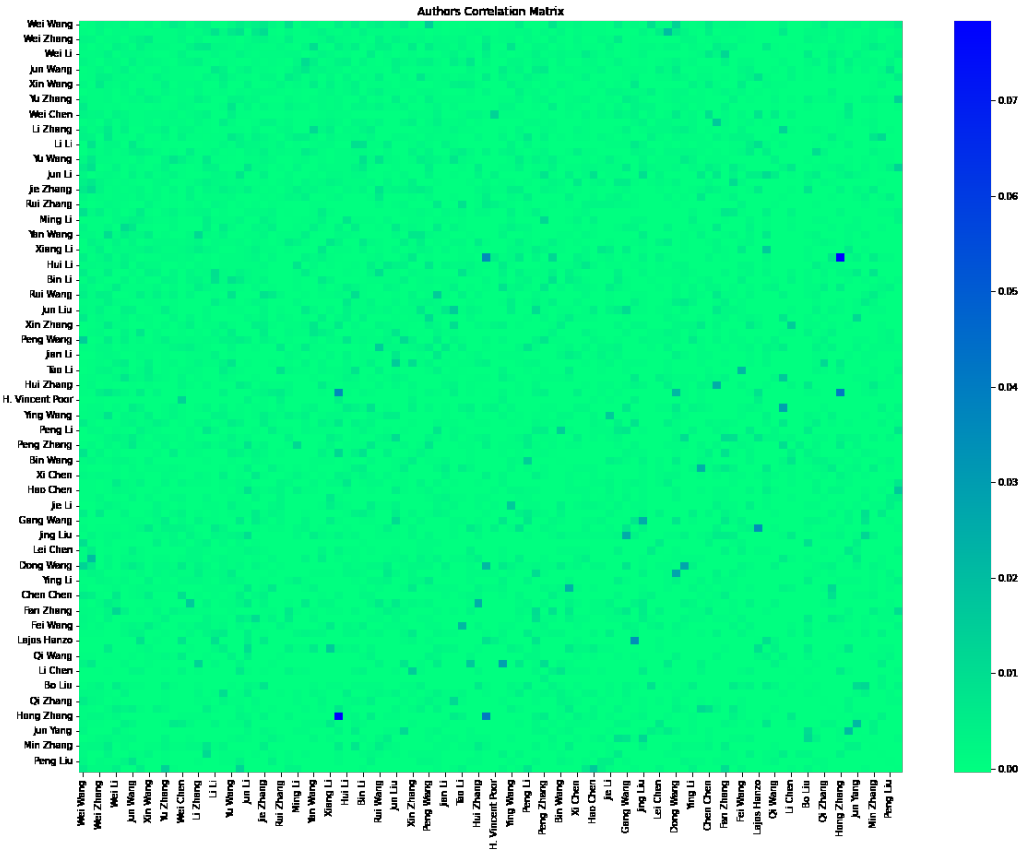

Figure 3.35: Heatmap for the correlation matrix of top 100 Authors

The above figure is the heatmap for the correlation matrix of top 100 Authors with most number of publications. From the heatmap, we can infer that the author pairs like (Chao

Wang, Hong Zhang), (Bo Zhang, Hong Zhang) etc are correlated strongly when compared to other pairs of authors.

	author1	author2	correlation
2	Wei Wang	Wei Wang	1
3	Wei Wang	Yang Liu	0.000539
4	Wei Wang	Wei Zhang	0.002391
5	Wei Wang	Lei Zhang	0.001384
6	Wei Wang	Wei Li	0.001011
7	Wei Wang	Lei Wang	0.002613
8	Wei Wang	Jun Wang	-3.28E-05
9	Wei Wang	Wei Liu	0.000386
10	Wei Wang	Xin Wang	0.001746

Figure 3.36

We also save the correlations in “authors_correlation.csv” file with three columns namely author1, author2, correlation.

3.8 Year wise distribution of publications/conferences

This is a simple representation of year wise publications count.

We extracted data from the 16 GB json file in order to retrieve the the information regarding publication count. The field "year" corresponds to the respective year in which the paper was published. The notation for "year" in the json looks like the following:

$$"year : NumberInt(value)"$$

We read the file line by line iteratively. And for every occurance of "year" in the json file corresponds to one publication. So, we take advantage of this by only parsing the "year" attribute from the file. We maintain a dictionary in which we add a particular year only once when we see it for the first time with a corresponding value 1. Later on, if we find same occurance of year we just increase the count by 1 of that particular year in the dictionary. Likewise, we can obtain the count for no of publications in a particular year.
Below is the diagram representing the year wise publications count.

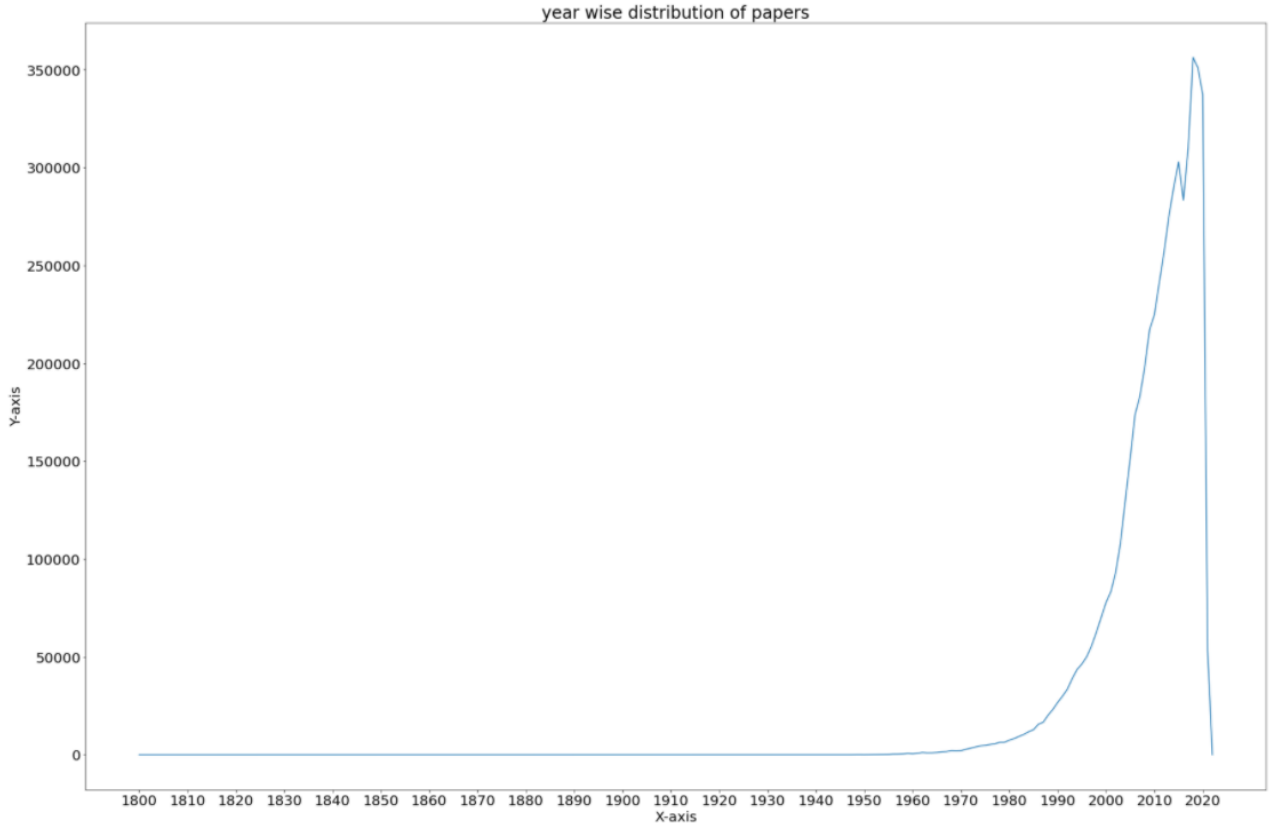

Figure 3.37: Year wise distribution of publications/conferences

4 Conclusion

- In this project analysis is performed on dblp.xml [1] and aminer citation dataset [2] and results have been plotted. Temporal trends in research topical, co-author relationship, geographical distribution publication sources have been analyzed. Association rules for the title words have been mined, Prediction of Co-Authors has been performed.

- From the results obtained we can infer that

  1. China and then USA are the major contributors of research articles. India comes under top 10. Considering citations USA overtakes China. But taking population into account, the density is more in European countries and Singapore.
  2. AI,NLP research currently trending while a lot of work has been done on theoretical computer science, programming languages in 1961-1981.
  3. K means clustering couldn't distinguish well between mobile networks and neural networks.
  4. There is a high probability that authors who have worked together previously will work in the future
  5. The number of publications every year is increasing exponentially.

  The project can be extended by analysing the trends in the research data not only limited to computer science, but including different backgrounds. In the future, estimation of future trends can be done, provided sufficient data, time and resources.

References

[1] Data provided by dblp. https://dblp.uni-trier.de/xml/.

[2] Aminer. https://www.aminer.org/, .

[3] Consolidated dblp data provided by Aminer; Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and Mining of Academic Social Networks. . In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'2008). pp.990-998. [PDF] [Slides] [System] [API], .

[4] Research paper classification systems based on TF-IDF and LDA schemes Sang-Woon Kim Joon-Min Gil . In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'2008). pp.990-998. [PDF] [Slides] [System] [API].

[5] Topic Modeling and Latent Dirichlet Allocation (LDA) in Python . https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf1

[6] Clustering documents with Python . https://towardsdatascience.com/clustering-documents-with-python-97314ad6a78d, .

[7] K means Clustering – Introduction . https://www.geeksforgeeks.org/k-means-clustering-introduction/, .