**MSc Final Project Declaration**

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in 7COM1039-0501-2020 - Computer Science Masters Project at the University of Hertfordshire (UH).

It is my own work except where indicated in the report. I did not use human participants in my MSc Project. I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

****

**University of Hertfordshire School of Computer Science**

MSc Data Science and Analytics with Advance Research

7COM1039-0501-2020 - Advanced Computer Science Masters Project

**Text Semantic Analysis using Latent Semantic Analysis**

Project report

By

Rahul Mannepally

Student Id: 19022966

Supervisor: Deepak Panday

Advisor: Dr Na Helian

**Acknowledgement**

I have received a great amount of support and assistance from my project advisor and supervisor throughout the writing and experimenting of this project. I would like to thank my project advisor DR. Na Helian and project supervisor Mr. Deepak Pandey for their invaluable guidance and assistance through every stage of the process. Lastly, I would like to thank all the staff members of the Faculty of Computer Science at the University of Hertfordshire.

**Table of Contents**

[Abstract: 7](#_Toc88014556)

[Chapter 1 Introduction 8](#_Toc88014557)

[Chapter 2 Literature Review 9](#_Toc88014558)

[2.1 LSA for Semantic analysis 9](#_Toc88014559)

[2.2 Statistical test 11](#_Toc88014560)

[Chapter 3 Methodology 13](#_Toc88014561)

[3.1 Data collection 13](#_Toc88014562)

[3.2 Data cleaning 13](#_Toc88014563)

[3.3 Latent Semantic Analysis (LSA) 15](#_Toc88014564)

[3.4 Word Cloud 18](#_Toc88014565)

[3.5 Wordnet 19](#_Toc88014566)

[3.6 Statistical test – Mantel test 20](#_Toc88014567)

[Chapter 4 Implementation 24](#_Toc88014568)

[4.1 Data Sets and Data Loading 24](#_Toc88014569)

[4.2 Data Cleaning 27](#_Toc88014570)

[4.3 Latent Semantic Analysis (LSA) 28](#_Toc88014571)

[4.4 Word Cloud 32](#_Toc88014572)

[4.5 Wordnet 32](#_Toc88014573)

[4.6 statistical test 35](#_Toc88014574)

[Chapter 5 Experimental Design and Results: 40](#_Toc88014575)

[5.1 Experiment Descriptions 40](#_Toc88014576)

[5.2 Results 41](#_Toc88014577)

[5.2.A Mantel Test Results Comparison (Pyplots) 61](#_Toc88014578)

[Chapter 6 Conclusion 65](#_Toc88014579)

[References 66](#_Toc88014580)

[Chapter 7 Appendix 69](#_Toc88014581)

[7.1 Experiment 1 Code 69](#_Toc88014582)

[7.2 Experiment 2 Code 87](#_Toc88014583)

[7.3 Experiment 3 Code 109](#_Toc88014584)

[7.4 Experiment 4 Code 135](#_Toc88014585)

List of Figures

[Figure 1 Hierarchical structure of LSA (Databricks Academy, 2019) 15](#_Toc88014586)

[Figure 2 Document term matrix 16](#_Toc88014587)

[Figure 3 The pipeline for creating a semantic and stable word cloud layout (Cui et al., 2010) 18](#_Toc88014588)

[Figure 4 Word cloud representation 19](#_Toc88014589)

[Figure 5 Cosine similarity matrix 20](#_Toc88014590)

[Figure 6 5-word Networkx representation 20](#_Toc88014591)

[Figure 7 Top words dataset for mantel test 22](#_Toc88014592)

[Figure 8 sample data from covid-psychology.txt 25](#_Toc88014593)

[Figure 9 Importing libraries 25](#_Toc88014594)

[Figure 10 code for loading data at article level from experiment 4 – “covid-science-large.txt” dataset. 26](#_Toc88014595)

[Figure 11 code for loading data at 2-3 paragraph-level from experiment 4 – “covid-science-large.txt” and “covid-psychology-large.txt” dataset 26](#_Toc88014596)

[Figure 12 Raw data 27](#_Toc88014597)

[Figure 13 code for Pre-processing and output 28](#_Toc88014598)

[Figure 14 Code for modelling LSA and generating top 5 words at article level from “covid-science-large.txt” dataset. 29](#_Toc88014599)

[Figure 15 Code for modelling LSA and generating top 5 words at article level from “covid-psychology-large.txt” dataset. 29](#_Toc88014600)

[Figure 16 Code for modelling LSA and generating top 5 words at 2-3 paragraph-level from “covid-science-large.txt” dataset. 30](#_Toc88014601)

[Figure 17 Code for modelling LSA and generating top 5 words at 2-3 paragraph-level from “covid-psychology-large.txt” 30](#_Toc88014602)

[Figure 18 Code for modelling LSA and generating top 5 words at paragraph-level(all paragraphs) from “covid-science-large.txt” dataset. 30](#_Toc88014603)

[Figure 19 Code for modelling LSA and generating top 5 words at paragraph-level (all paragraphs) from “covid-psychology-large.txt” dataset. 30](#_Toc88014604)

[Figure 20 Code for mantel test 39](#_Toc88014605)

[Figure 21 P-values for Mantel test with 10-word network 61](#_Toc88014606)

[Figure 22 r-statistic for Mantel test with 10-word network 62](#_Toc88014607)

[Figure 23 Z-score for Mantel test with 10-word network 62](#_Toc88014608)

[Figure 24 P-values for Mantel test with 20-word network 63](#_Toc88014609)

[Figure 25 R-values for Mantel test with 20-word network 63](#_Toc88014610)

[Figure 26 z-score for Mantel test with 20-word network 64](#_Toc88014611)

List of Tables

[Table 1 List of top words from LSA model 32](#_Toc88014613)

[Table 2 Word cloud representation of top words 33](#_Toc88014614)

[Table 3 Cosine similarity matrix for top 5 words 34](#_Toc88014615)

[Table 4 WordNet of top 5 words 35](#_Toc88014616)

[Table 5 Mantel test with 10-word network 36](#_Toc88014617)

[Table 6 Vertical comparison mantel test 38](#_Toc88014618)

[Table 7 List of top words from LSA model experiment 1 42](#_Toc88014619)

[Table 8 List of top words from LSA model experiment 2 43](#_Toc88014620)

[Table 9 List of top words from LSA model experiment 3 44](#_Toc88014621)

[Table 10 List of top words from LSA model experiment 4 45](#_Toc88014622)

[Table 11 Word cloud representation of top words from Experiment 1 46](#_Toc88014623)

[Table 12 Word cloud representation of top words from Experiment 2 47](#_Toc88014624)

[Table 13 Word cloud representation of top words from Experiment 3 48](#_Toc88014625)

[Table 14 Word cloud representation of top words from Experiment 4 49](#_Toc88014626)

[Table 15 Cosine Similarity matrices for Experiment 1 50](#_Toc88014627)

[Table 16 Cosine Similarity matrices for Experiment 2 51](#_Toc88014628)

[Table 17 Cosine Similarity matrices for Experiment 3 51](#_Toc88014629)

[Table 18 Cosine Similarity matrices for Experiment 4 52](#_Toc88014630)

[Table 19 Wordnet of top 5 words form experiment 1 53](#_Toc88014631)

[Table 20 Wordnet of top 5 words form experiment 2 54](#_Toc88014632)

[Table 21 wordnet of top 5 words form experiment 3 55](#_Toc88014633)

[Table 22 wordnet of top 5 words form experiment 4 56](#_Toc88014634)

[Table 23 Mantel-Test results for Law and Non-Law datasets (4 articles dataset) 57](#_Toc88014635)

[Table 24 Mantel test results for COVID science – psychology (10 articles dataset) 57](#_Toc88014636)

[Table 25 Mantel test results for COVID science – psychology (15 articles dataset) 58](#_Toc88014637)

[Table 26 Mantel test results for COVID science – psychology (20 articles dataset) 58](#_Toc88014638)

[Table 27 Mantel test results for COVID science – psychology (30 articles dataset) 58](#_Toc88014639)

[Table 28 Mantel test results for COVID science – psychology (40 articles dataset) 58](#_Toc88014640)

[Table 29 Mantel test results for law dataset 59](#_Toc88014641)

[Table 30 mantel test results for non-law dataset 59](#_Toc88014642)

[Table 31 mantel test results for Covid science dataset with 15 articles dataset 60](#_Toc88014643)

[Table 32 mantel test results for Covid psychology dataset with 15 articles dataset 60](#_Toc88014644)

[Table 33 mantel test results for Covid science dataset with 20 articles 60](#_Toc88014645)

[Table 34 mantel test results for Covid psychology dataset with 20 articles dataset 60](#_Toc88014646)

[Table 35 mantel test results for Covid science dataset with 30 articles 61](#_Toc88014647)

[Table 36 mantel test results for Covid psychology dataset with 30 articles dataset 61](#_Toc88014648)

[Table 37 mantel test results for Covid science dataset with 40 articles dataset 61](#_Toc88014649)

[Table 38 mantel test results for Covid psychology dataset with 40 articles dataset 62](#_Toc88014650)

# Abstract:

The semantic meaning of words on the same topic might vary across different subject areas (e.g., The words from scientific articles vary with sematic meaning when compared with words from psychological articles). The project is to identify the semantic difference between two corpuses with different subject areas by group of words generated from the corpuses using Latent Semantic Analysis (LSA).

Traditional Information retrieval systems are designed to retrieve the information from the source by giving a query or group of strings we are searching for. By matching those strings with the source data to retrieve the most relevant document from the internet or any other databases. The way similarity of the document is usually calculated by matching the search string with the information in the documents but retrieving the information by a statistical method can be most relevant and effective. Like searching for a research paper of a specific category like scientific research papers, psychology research papers with a statistical method will be helpful. To retrieve specific categorized information statistically, an application or a method should be developed, which could differentiate the different types of content available in different documents semantically. Likewise in this project developing a statistical method that could differentiate scientific articles from psychology articles using Latent Semantic Analysis.

To avoid this limitation of using pre-defined tags or by direct matching of search string with the document titles or content, a statistical method needs to be developed to derive the semantic value that could differentiate multiple type of content specific documents.

After conducting the experiments and investigating the problem to identify a statistical test to differentiate different type of content or documents, The mantel test results shoes that there is a significant difference between the two corpora of data with the words generated by Latent Semantic Model but varied with multiple parameters like size of the dataset, document level of Latent Semantic Analysis (LSA). The experimental results were more consistent at paragraph level of LSA and moderate dataset sizes (ex: 10 articles, 20 articles, 30 articles) by considering p-values form mantel test statistic. P-values for 10-word network at paragraph level 0.84(10 articles), 0.82(15articles), 0.92(20 articles), 0.95(30 articles), 0.90(40 articles dataset).

# Chapter 1 Introduction

With people spending increasing time online for studying and researching looking for resources like scientific research papers, newspapers, psychology research papers etc. It is hard and time consuming to find resources that we want from the ocean of content available on the internet. But, if different documents available on the internet are tagged according to the content available within the document, It will at least decrease the complexity of searching for a specific type of document to some extent. To achieve this first an application or a method needs to be developed to analyse the content available in the document and differentiate it from other documents statistically.

Latent Semantic Analysis(LSA) is a natural language processing technique based on linear algebra that tries to capture and code the semantics of words and documents at different levels of document like paragraph level, document level, etc. (Pilato and Vassallo, 2015). There are a lot of algorithms to extract lexical information from text corpora. In this area, the focus is to identify what can be extracted from the distribution of words, excluding sub lexical information, perceptual grounding and intimate constraint(Dennis et al., 2003). LSA is a well-known algorithm for computational modelling of the construction of semantic representations of the corpus. LSA is also widely used in information retrieval systems(Wenli, 2016). From(Mohamed and Watada, 2010), an LSA approach to document classification for knowledge application acquisition has been proposed, but the classification of documents is done by generating a set of words with high weights from the document and matching them to a predefined dictionary of categorised words obtained from training the documents. The scope of using a predefined dictionary to the model might not be the ultimate or the best solution to categorise the documents effectively.

The project is to investigate and develop an application to model systematic communication by extraction and representation of the contextual usage of words through the statistical computation applied to large corpora of text for both semantic and syntactic analysis. For my project, I will find the statistical difference between law and science, science, non-science, and psychology papers from a corpus of texts, by extracting the insights from the data using Latent Semantic Analysis (LSA) and then by using statistical tests to compare the data extracted from each model and then find the insights that could differentiate those documents with statistical measure.

To avoid this limitation of using pre-defined words from the training data to classify the corpus of text, a statistical method needs to be developed to derive the semantic distance between the weighted words. Knowing that there are many algorithms to analyse the document semantically, knowing the best algorithm to produce weighted words to analyse the document is a problem. There are many statistical methods to test the results, like the chi-square test, t-test, Mantel test. Choosing one that suites to the project results is a problem. So, the research question raises here.

Hypothesis: The semantic meaning of words on the same topic might vary across different subject areas (e.g., science and psychology).

Research question: Can LSA and mantel test identify the statistical difference between the semantic meaning of words when placed in two different subject areas?

# Chapter 2 Literature Review

Many of the first and most ambitious systemic approaches were proposed by (Von Bertalanffy, 1967), He was interested in mathematical laws that characterized systems in different fields. who as early 1920s tried to develop a general theory of systems to identify and formalize abstract principles that apply to most systems, be they sociological, physical, psychological, and biological? And proved that a small number of systems of differential equations can explain natural and social phenomena in different fields.

Later (Ashby, 1961) carried out the further mathematical formulation of system behaviour. Explained the importance of feedback control, input, output, prediction, and stability while working on aircraft fire control. He generalizes these concepts to biology and social sciences. And in (Brooks-Gordon and Freeman, 2006) mentioned different computational models of the construction of syntactic structures include the Pooled Adjacent Context Model (PAC), the Syntagmatic Paradigmatic model (SP) and computational models of the construction of semantic representations including, Latent Semantic Analysis (LSA), Hyperspace Analogue to Language (HAL), Sparse Random Context Representation (SRCR) and Word Association Space(WAS). These models are used to elucidate the similarity or difference, of word and passage meaning through the statistical analysis of large text corpora. Unsupervised learning techniques can be used to build higher-level representations of word meaning due to statistical redundancies observed in the language(Kintsch, 2001).

Choosing the corpus size plays an important role in obtaining test results. Large, heterogeneous corpora are a risk because they may include more noise or too much specific information from a single area, lowering the accuracy of the resulting models. However, there is no consensus on the predicted corpus size or what constitutes a large or small corpus(Dascalu and McNamara, 2017).

The author's study not only highlights the advantages of employing larger corpora to create LSA and LDA models but also paves the way for future research. The requirement to study LSA and LDA models constructed on large and small corpora that are counterbalanced in terms of domains covered and linguistic complexity is one of the most important of them. Future research should also look into the possibility of LDA models contributing to our knowledge of cognitive processing(Dascalu and McNamara, 2017).

## 2.1 LSA for Semantic analysis

In Latent semantic analysis, the term latent means hidden in the data which cannot be directly measured. These features are essential to the data but are not essential features of the data set. Latent Semantic analysis is a natural processing technique as well as an unsupervised learning technique. The latent semantic analysis aims to create representations of the text data in terms of topics or latent features. Which can produce dimensionality of the original text-based dataset(Landauer, 2014).

Traditional information retrieval system models mainly consist of probability models or space models or vector models. Its basic principle being the keyword retrieval system. It is like matching the words that need to be searched with the words from the index of the document. But the existence of synonyms and polysemy made these systems complex to retrieve the information(Wenli, 2016). Later (Dumais et al., 1988) proposed latent semantic analysis to improve access to textual information.

Latent Semantic Analysis(Landauer et al., 1998), uses free text as an input, with the paragraph as a relevant contextual unit. An analysis of word frequency within the paragraphs represents the meaning and relationship of words as vectors which helps to compare words semantically even at high order co-occurrences(Kontostathis and Pottenger, 2002). (Foltz, 1996) rigorously tested LSA and it is regarded as a very successful theory and better than Hyperspace Analog to Language(Meena and Raj, 2014) which is a method for lexical analysis and a model of semantic analysis. HAL uses a sliding window of co-occurrences words as a measure of context and avoiding higher-order co-occurrences.

(Anwar, 2006) conducted some experiments with latent semantic analysis for tagging and ambiguity correction for words from the large corpus. And provided details about how LSA does better compared to other statistical word tagging and ambiguity correction methods. And the size and the meaning of the corpus plays an important role while using latent semantic analysis(Dascalu and McNamara, 2017). An application(Rosenbusch et al., 2020) has been produced addressing the arbitrary measurements and disconnection between research groups in psychology by detecting semantic overlap between the scales with semantic analysis. He also explained that These similarity or dissimilarity measurements can be shown by cosine similarity and can be represented a space by word network form.

Researchers have recently advanced beyond LSA models of semantic representations to investigate topic-based models of semanticity, the most prevalent of which is Latent Dirichlet Allocation(Blei et al., 2003). Because each word in the semantic vector space model can be represented as a context-free vector, LSA can be thought of as a cognitive statement of meaning. Although individual dimensions of concepts do not have a distinct meaning, the total representation provided by LSA can be thought of as a map of meanings. Positive correlations between LSA similarity scores and human recall using word association lists further support a semantic proximity effect in which LSA resembles human memory(Kintsch, 2001).

To embed words in a vectorial space, corpus-based semantic representations take advantage of statistical features of textual structure. Term pairs with comparable meanings tend to cluster together in this space. These approaches are based on the notion that words with similar meanings are more likely to appear in comparable contexts. This theory is known as the distributional hypothesis, and it provides a useful framework for comprehending and computing semantic relationships between words(Altszyler et al., 2017).

To learn correct word embeddings in small text corpora, the author compares the capabilities of Skip-gram and LSA. To accomplish so, we used nested subsamples of a medium-sized corpus to assess the model's capacity to represent semantic categories (such as drinks, countries, tools, and garments). When models are trained with medium-sized datasets ( 10 million words), Word2vec embeddings beat LSAs. When the corpus size is lowered, however, Word2vec performance suffers significantly, making LSA the more appropriate technique. This discovery adds to the debate over prediction-based vs. counter-based models. They assume that the performance drops of 10 Word2vec in small corpora is because prediction-based models require a large amount of training data to fit their large number of parameters.

We used LSA and Skip-gram capabilities to extract significant semantic word correlations in dream reports as a case study. Even in cases of series with a small number of dreams and a low frequency of target words, we discovered that LSA can accurately identify semantic word relations. This is a step forward in the application of word embeddings to dream content analysis. This branch of study investigates things like "what do we dream about?" and "how do gender, cultural background, and waking life events impact the content of our dreams?" They recommend that LSA be used to investigate word connections in dream reports, potentially shedding new light on an ancient topic of psychology research(Altszyler et al., 2017).

There is a constant proliferation of novel notions and scales in psychological measurement and theory. Psychological science is grappling with arbitrary measurement, construct dilution, and separation between research groups due to the sometimes-redundant nature of new scales. We introduce the Semantic Scale Network, an easy-to-use online tool, to overcome these challenges. The goal of this application is to use latent semantic analysis to find semantic overlap between scales automatically. Authors and reviewers can insert items for a new scale into the app and receive semantic overlap quantifications with related scales in the app's corpus. Unlike typical scale overlap evaluations, the programme can allow expert scale redundancy judgments without access to empirical data or understanding of all possible associated scales. They explain the Semantic Scale Network and discuss best practices for interpreting its outputs after a brief introduction to metrics of semantic similarity in texts(Rosenbusch et al., 2020).

Using two criteria, classify a network-similarity algorithm. First, at what network level does it operate? Second, what kind of comparison is it employing? We identify three levels for the first criterion: micro, meso, and macro. At the micro-level, a method extracts features from nodes or egonets;1 at the mezzo-level, it extracts features from communities; and at the macro-level, it extracts features from the global/network level, as their names suggest. We have three sorts of criteria for the second criterion: vector-based, classifier-based, and matching-based(Soundarajan et al., 2014).

In applications like transfer learning and change detection, quantifying the difference between two networks is crucial. For this goal, several network similarity algorithms have been proposed. The problem's origins can be traced back to the problem of [determining graph isomorphism], for which no polynomial-time algorithm exists. The task of network similarity is significantly broader(Soundarajan et al., 2014).

## 2.2 Statistical test

Latent Semantic Analysis can be used to generate principal components of a text corpus (Slomovitz, 2017). But to compare these principal components with other text corpora, by considering each principal components elements as populations we need a statistical test. A statistical test that could differentiate different types of corpuses need to be defined.

Many statistical tests could be used to compare two populations. Some of them are for paired or matched observation, nominal data - McNemar’s Test, For Ordinal (Ordered categories) - Wilcoxon test, For Quantitative data (Discrete or Non-Nominal) – Wilcoxon, For Quantitative (Normal\*) – Paired t-test.

For comparing two or more groups there are two categories of tests parametric tests and non-parametric tests. In Parametric tests, they are Paired t-test, Unpaired t-test, Pearson correlation, One-way analysis of variance while their equivalent non-parametric tests are Wilcoxon rank-sum test, Mann-Whitney U test, Spearman correlation test, Kruskal Wallis Test(Campbell, 2021).

When sampling from multivariate normal populations, Kullback's statistic for verifying equality of many correlation matrices can be considered a modified likely hood ratio statistic, according to the article(Gupta et al., 2013). Expand data in terms of other random variables and then reverse the expansion term by term to get the asymptotic null distribution of data in series involving independent chi-square variables. An example is also provided to demonstrate the technique to be followed when using the statistic to assess the equivalence of correlation matrices.

The author of this paper(Gupta et al., 2013) looked at the difficulty of assessing how similar two networks are when node-correspondences are unknown. In real-world applications such as transfer learning and change detection, this issue arises regularly. There are a plethora of network similarity approaches to choose from, and it's difficult to know which one to use. The first empirical examination of the relationships between various network similarity approaches is presented. We provide (1) a method for identifying groups of comparable network similarity methods and (2) a method for calculating consensus among a collection of network-similarity methods. By applying our approaches to a variety of real datasets across several areas, we compare and contrast twenty network-similarity strategies. Our findings show that (1) multiple network-similarity approaches are surprisingly well correlated, (2) certain complex network-similarity methods may be approximated by a much simpler method, and (3) a few network similarity methods provide rankings that are extremely near to the consensus ranking.

It is desirable to be able to make statistical comparisons between correlation coefficients recorded on the same individuals in a variety of scenarios in psychological research. For example, an experimenter might want to see if two predictors have the same correlation with a criterion variable. In a different scenario, the investigator might want to see if a whole matrix of correlations has remained stable over time. The current paper examines the literature on such tests, identifies specific statistics to avoid, and describes a number of procedures that can be applied securely with medium to large samples. A number of numerical examples are offered to help illustrate the point(Steiger, 1980).

The former problem is handled in traditional statistical analyses by multivariate approaches that allow one to pay to the correlations among predictor variables; partial regression is a well-known solution to this problem. Path analysis is an interpretive method for conceptually distinguishing causal linkages from spurious associations resulting from chance correlations between variables. However, the second difficulty, namely that autocorrelation in the variables violates the principles of parametric analysis, confounds traditional parametric techniques(Diniz-Filho et al., 2013).

The Mantel test is a method for overcoming some of the difficulties that come with describing species-environment connections. Mantel's test is a regression in which the variables are distance or dissimilarity matrices that sum up pairwise similarities across sample sites(Diniz-Filho et al., 2013).

# Chapter 3 Methodology

## 3.1 Data collection

This is the first and important step of the experiment. The data collected needs to be research specific data. This is the process of assembling, quantifying information on target variables and analysing insights(Morgan and Harmon, 2001). For this project, the data collected need to be both quantitative and qualitative data, for statistical insights and context-specific insights.

Before collecting data, researchers must first determine whether scales are redundant to the new instrument, as they will need participant data to quantify shared variation. Despite the greatest attempts, it is difficult to be aware of every scale that might be important to one's research despite best efforts. Relevant scales are frequently published under various names or in unrelated areas and hence may go unnoticed by researchers. Second, researchers must collect data from the same test subjects for all scales, which might be difficult if there are too many related scales for each participant to complete. Third, the line between strong convergent validity and redundancy does not exist(Kabir, 2016).

In quantitative terms, the datasets collected need to be significantly different and of different sizes to compare the results.

Ex: Covid science dataset of multiple sizes and covid non-science dataset of multiple sizes

In qualitative terms, the datasets collected need to be semantically differentiable.

Ex: One data set with scientific research papers and the other with psychology research papers.

## 3.2 Data cleaning

The data collected from multiple sources consists of a lot of noise in it like unwanted data, symbols etc. So, to perform experiments on the data collected the data need to be cleaned. Data cleaning is a process of detecting, identifying, and correcting data inconsistency, duplicity, redundancy(Broeck et al., 2005).

Data cleaning, also known as data cleansing or scrubbing, is the process of discovering and removing mistakes and inconsistencies from data to improve its quality. Single data collections, such as files and databases, might have data quality issues, such as misspellings during data entry, missing information, or other erroneous data. The necessity for data cleansing increases dramatically when various data sources must be connected, such as in data warehouses, federated database systems, or worldwide web-based information systems. This is due to the fact that the sources frequently provide redundant data in various forms. Consolidation of disparate data formats and the deletion of redundant information is required in order to enable access to correct and consistent data(Broeck et al., 2005).

Text pre-processing is a technique for cleaning text data and preparing it for use in a model. Text data comprises noise in the form of emotions, punctuation, and text in a different case, among other things. When it comes to Human Language, there are many different ways to communicate the same thing, and this is only the beginning of the challenge. Machines cannot understand words; they require numbers; thus, we must convert text to numbers efficiently.

There are many methods developed for data cleaning. But before applying those methods, the data should be overseen once to detect, identify the issues in the data and skim any unwanted data. Different types of data have different techniques for data cleaning. Since our data is textual data, these techniques like Remove Punctuations, remove words and digits containing digits, Remove, stop words, Rephrase Text, Stemming and Lemmatization, Lower case, Remove White spaces need to be performed.

There are three types of stemming algorithms. They are Potter stemmer, Snowball stemmer, Lancaster stemmer. The potter stemmer is considered as the oldest one but easy to implement and it mainly concerns removing the common endings to words so that they can be resolved to a common form. While snowball stemmer is the updated stemmer of snowball stemmer, but it is more aggressive when compared to potter stemmer. In Lancaster stemmer is also one the most aggressive stemming algorithms. Lancaster stemmer is mainly used for customizations and this algorithm can weirdly transforms words into strange stems. Among all of this stemmers potter stemmer better suites for this project.

Here is the sample example for data pre-processing

Data before pre-processing:

['REGINA v SECRETARY OF STATE FOR THE HOMEDEPARTMENT, Ex parte HINDLEY[COURT OF APPEAL][2000] 1 QB 152HEARING-DATES: 5, 6, 7, October 5 November 19985 November 1998CATCHWORDS:Prisons - Prisoners\' rights - Release on licence - Mandatory life sentence prisoner - Tariff element of determinate length provisionally fixed but not communicated to prisoner’]

Applying these pre-processing techniques:

|  |  |  |
| --- | --- | --- |
| **Pre-processing technique** | **Function** | **purpose** |
| Regex tokenizer | RegexpTokenizer(r'\w+') | Used to extract the tokens from string by using regular expression. |
| Stop word list | set(stopwords.words('english')) | Stop words are frequently used in Text Mining and Natural Language Processing (NLP) to exclude terms that are so widely used that they contain little meaningful information. Ex: “a”, “the”, “is”, etc are stop words. |
| Porter Stemmer | PorterStemmer() | The Porter stemmer is a method for removing frequent morphological and inflexional endings from English words. Ex: (play, playing) 🡪 play |

Data after pre-processing output:

['regina', 'secretary', 'state', 'homedepartment', 'parte', 'hindley', 'court', 'appeal', '2000', '152’,’hearing', 'dates', 'october', 'november', '19985', 'november', '1998’, ’catchwords', 'prisons', 'prisoners', 'rights', 'release', 'licence', 'mandatory', 'life', 'sentence', 'prisoner', 'tariff', 'element', 'determinate', 'length', 'provisionally', 'fixed', 'communicated', 'prisoner']

## 3.3 Latent Semantic Analysis (LSA)

The most widely used techniques for determining document proximity can be split into three categories: Adaptive Strategies (AS) such as probabilistic, neural networks, and genetic algorithms, as well as Vector Space Models (VSM) and Latent Semantic Indexing (LSI)(Ibrahimov et al., 2002). For this project, we focus on LSI.

Latent Semantic Analysis (LSA) is a natural processing technique as well as an unsupervised learning technique, that tries to capture and code the semantics of words and documents at different levels. After collecting and cleaning the data the data need to be modelled using LSA. Which generates a weighted matrix by semantically comparing each word to all other words in the corpus. This weighted matrix can be achieved by importing the Latent Semantic Indexing (LSI) model from genism. model’s library and set the data to the model. The LSI model first generates the document term matrix and then performs singular value decomposition on the matrix. From this, the weighted matrix of the words is generated at multiple levels of documents.

Steps involved in LSA:

Diagram

Description automatically generated

Figure 1 Hierarchical structure of LSA (Databricks Academy, 2019)

The dilemma of how to locate relevant documents from search phrases prompted, the development of Latent Semantic Analysis. When we compare words to identify relevant papers, the basic challenge that occurs is to compare the meanings or concepts behind the words. But LSA tries to tackle this problem by putting both words and documents in a "concept" space and comparing them there.

Example:

The aim of the Latent Semantic Analysis is to create the representations of the text data in terms of topics or latent features. Document term matrix is a matrix of the words in Euclidian space. Considering an example of document term matrix.

Raw text words:

‘the quick fox’

‘the quick’

|  |  |  |  |
| --- | --- | --- | --- |
|  | ‘the’ | ‘quick’ | ‘fox’ |
| ‘the fox’ | 1 | 0 | 1 |
| ‘the quick dog’ | 1 | 1 | 0 |

Figure 2 Document term matrix

Step 1 of LSA – Document term matrix

Each row in this matrix represents one of the documents. In this example one of two sentence fragments and then each column represents a word from the dictionary, that is the word that shows up and at least one of the two documents.

Now we consider these two documents as vectors

‘the quick fox’ = (1,0,1)

‘the quick’ = (1,1,0)

Now evaluating the second step of LSA - truncated-Singular Value Decomposition

Let 𝑀 be the real 𝑛×𝑑matrix that we want to decompose.

The SVD theorem states:

(𝑴=𝐔𝑺𝐕𝐓)

𝑴𝒏×𝒅 = 𝑼𝑛×𝒏 𝑺𝑛×𝒅 𝑑×𝑑

where 𝐔 is a column-orthonormal matrix; the columns of the 𝐔matrix are called the left-singular vectors of 𝑀.

𝑺 is a 𝑛×𝑑 diagonal matrix, and the diagonal values in the 𝑺 matrix are known as the singular values of the original matrix 𝑀.

Usually, the singular values are stored in a descending order along the main diagonal in 𝑺.

𝐕is a column-orthonormal matrix; the columns of 𝐕are called the right-singular vectors of 𝑀.

M =

=

det|- = 0

1 = 0

The eigenvalues of are 3, 1, and 0

Then we calculate

𝑴=

S = [1.73 ,1.0 ]

U =

=

Truncated = =

‘The’ 0.8 0.4

‘quick’ 0.4 0.7

‘fox’ -0.5 0.5

The weights of terms in the document are given in order.

## 3.4 Word Cloud

Word Cloud is an application program interface that is used for an easy and clear representation of the words which are related to each other in a corpus. Word Cloud is a process of coupling the trend chart with the words with dynamic word clouds to visualize the material content evolutions in a set of documents(Cui et al., 2010).

Although word clouds are more versatile than traditional time series views when it comes to displaying intricate word associations, they aren't usually built for side-by-side comparison. For example, the size and position of a word in distinct word clouds normally vary, and words regularly appear and depart over time. Exploring temporal trends of documents with varied time stamps is difficult using traditional word clouds. We present a versatile method for creating a word cloud layout tailored to such papers. To suit diverse user requirements, our technique can structure layouts according to multiple semantic-coherence criteria (Cui et al., 2010).

Pipeline for creating a word cloud. A group of words is retrieved from a collection of documents to start the pipeline (see Figure a). Then, based on their properties, we put all extracted words on a 2D plane (see Figure b). filter out any unimportant or unrelated words from the 2D plane (see Figure c) and generates a triangular mesh using Oelau-nay triangulation" on the remaining points, each of which is at the centre of a word. The font size of each word is determined by the frequency of that word at that moment (see Figure d). Finally, to alter point placements and get a suitable layout, we use an adapted force-directed algorithm (see Figure e)(Cui et al., 2010).

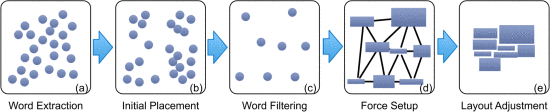


Figure 3 The pipeline for creating a semantic and stable word cloud layout (Cui et al., 2010)

Steps:

1. Initially set of words with different timestamps are generated from the documents.
2. Multidimensional scaling is used to place the extracted words on a 2D plane.
3. Unrelated words are filtered out at this step.
4. The remaining words are set up in a network.
5. Optimising the arrangement using a force-directed method (Cui et al., 2010).

Input: Top words from the corpus

Word cloud:

**Text

Description automatically generated**

Figure 4 Word cloud representation

## 3.5 Wordnet

The term wordnet means a word network. Wordnet is a process of illustrating the relation(similarity or dissimilarity) between the words from different documents, graphically in a network. The network consists of nodes and edges. Nodes represent the words and the edges represent the similarity or dis-similarity metrics between those words in the corpus. The similarity or dissimilarity among words can be calculated by cosine similarity metrics. NetworkX is a python package that is used to generate wordnet.

NetworkX is a Python library that allows you to explore and analyse networks and network methods. Simple graphs, directed graphs, graphs with parallel edges, and graphs with self-loops are all represented by the core package's data structures. NetworkX graphs can have any (hashable) Python object as a node, and edges can have any data as an edge; this flexibility made NetworkX excellent for expressing networks in a variety of scientific domains. Many graph algorithms are implemented in addition to the basic data structures for determining network attributes and structure metrics, such as shortest paths, betweenness centrality, clustering, and degree distribution, among others. NetworkX includes generators for several classic graphs and common graph models, including the Erdoes-Renyi, Small World, and Barabasi-Albert models, as well as the ability to read and write many graph formats for easy interchange with existing data. NetworkX is a strong tool for scientific computations because of the Python programming language's ease of use and flexibility, as well as its connection to the SciPy tools. We share some of our recent work on coupled oscillator synchronisation to show how NetworkX facilitates research in the field of computational networks(Hagberg et al., 2008).

Input: cosine similarity matrix of top 5 words’

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 5 Cosine similarity matrix

Networkx graph:

Diagram

Description automatically generated

Figure 6 5-word Networkx representation

Where:

The nodes represent the words from the documents and the edges represent the weight of similarity between the words in the document.

## 3.6 Statistical test – Mantel test

To compare the correspondence of one set of distances with another, a variety of approaches have been created. Among these are (1) matrix correlation techniques, (2) Spielmen's network matching techniques, (3) matrix dilation and rotation techniques, and (4) Lingoes and Guttman's smallest-space approaches. All of these strategies have trouble determining statistical significance. The issue is that all pairwise distances between k units (in this case semantics) cannot be independent.

One of the most common techniques for evaluating spatial dynamics driving population structure is to compare genetic divergence or genetic distances, estimated by pairwise FST and associated statistics, with geographical distances, as measured by the Mantel test (Diniz-Filho et al., 2013). Using a similar method, we apply the mantel test to the semantic word population and use the mantel test to calculate the geographical distance in the space. To assess the importance of various populations. We must experiment with various populations (top words) derived from the corpus of texts.

When comparing one corpus to another, we need to see if the shape of the global representation of the document collection is stable. The matrix of pairwise similarities across the papers is one way to depict this global form. The Mantel test is a technique for comparing two similarity matrices. This test is a popular way to evaluate the relationships between two distance matrices, or more broadly, two similarity or proximity matrices. It involves using a statistic r to assess the significance of the measure of connection between the elements in two -matrices and then comparing it to the distribution of the values discovered by randomly reallocating the order of the elements in one of the matrices. For two similarity matrices, we used the statistic r, which is the sum of the products of the standardised similarities(Besançon and Rajman, 2002).

“Mantel's test is based on a regression analysis, in which the variables are distance dissimilarity (or similarity) matrices summarizing all pairwise sample combinations”(Diniz-Filho et al., 2013).

Assuming, 2D data consisting of group of top similarity words (Z(s1),Z(s1),…, Z(sN). If the pixel values are correlated or spatially dependent, spatial coordinates that are near to one another will be paired with intensity values that are likewise close to one another. To statistically model the clustering process.

Wij=∥si−sj∥ Describes the spatial proximity of si and sj

And

Uij=|Z(si)−Z(sj)| quantifies the closeness in range Z(si) and Z(sj)

Where i and j represent to integer positions on sampling lattice

Mantel test statistic (Khademi et al., 2009)

M2 =

Example for evaluating Mantel test:

Considering multiple samples data in two different spaces (Biological and Environmental space). Where nodes represent the samples and the edges represent the relation between the samples in terms of their wights, by considering different parametric axis (Species abundance in biological space and mean sea surface temperature, tidal current and depth in environmental space).

Example showing the evaluation of mantel test:

Considering set of four words in two different spaces (space 1 consists of corpus of law words and space 2 consists of corpus of non-law words). And each space is divided into three documents. The set of four words and their weights in their respective document spaces are shown in below table.

**Space 1 Corpus of law words**

**Space 2 Corpus of non-law words**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **‘Word’** | **‘Weight in space 1(law)’** | | | **‘Weight in space 2(non-law)’** | | |
|  | Doc 0 | Doc 1 | Doc 2 | Doc 0 | Doc 1 | Doc 2 |
| ‘judg’ | 0.0593 | -0.0758 | -0.009 | 0.3207 | 0.2275 | 0.0550 |
| ‘evid’ | 0.0095 | 0.0345 | -0.0204 | 0.3092 | 0.2394 | -0.0117 |
| ‘tariff’ | 0.3329 | 0.0379 | 0.1076 | 0.0364 | -0.1337 | 0.2780 |
| ‘secretari’ | 0.3314 | 0.0966 | 0.0830 | 0.0212 | -0.0786 | 0.1560 |

Figure 7 Top words dataset for mantel test

Now evaluating the pairwise distances

Distance =

Pairwise distance =

Pairwise distance(‘judge’,’evid’) in space 1

Judge = (0.0593 -0.0758 -0.0090)

Evid = (0.0095 0.0345 -0.0204)

D(‘judge’,’evid’) =

= 0.1216

Pairwise distance matrix **‘X’** in space 1

judg evid tariff secretari

judg 0 0.1216 0.3184 0.3351

evid 0.1216 0 0.3478 0.3438

tariff 0.3184 0.3478 0 0.0637

secretari 0.3354 0.3438 0.0637 0

Pairwise distance matrix **‘Y’** in space 2

judg evid tariff secretari

judg 0 0.06976 0.5187 0.4475

evid 0.06976 0 0.5511 0.4660

tariff 0.5187 0.5511 0 0.1349

secretari 0.4475 0.4660 0.1349 0

Evaluating z score

Z =

i = 1🡪3 j = 1🡪3

Z =

Z = (0+0.0084+0.1651+0.1499+0.0084+0+0.1916+0.1602+0.1651+0.1916+0+0.0085+0.1500+0.1602+0.0085)

Z = 0.13674

Mantel r statistic formula

M2 =

Elaborate

r-value =

Where X and Y are variables measured at location I and j,

Sx and Sy are standard deviations for X and Y,

N is number of elements I distance matrix

Here N = 4

Text

Description automatically generated with low confidence

=

= (0.1216+0.3184+0.3478+0.3354+0.3438+0.0637)/6

= 0.255

=

= (0.0697+0.5187+0.5511+0.4475+0.4660+0.1349)/6

= 0.3646

Sx =

= 0.1643

Sy =

= 0.0716

r = 0.95

# Chapter 4 Implementation

## 4.1 Data Sets and Data Loading

For this project, 5 different datasets with large corpora of text were collected to perform LSA on them. The first one is law-text-all.txt which consists of 3 articles with legal judgements. In second dataset non-law-all.txt, consists of 4 articles with general conversations. Each article is separated by the word NEW\_ARTICLE.

In third dataset covid-science.txt, consists of covid related scientific research papers collected from multiple sources (ex. IEEE Xplore, Google Scholar). This dataset is then partitioned into multiple sizes, 10 papers(covid-science-small), 20 papers(covid-science), 40 papers(covid-science-large) in order to test LSA performance at multiple levels. Each article is separated by the word “NEW\_PAPER”.

The fourth dataset is covid-non-science.txt, which consists of covid related newspaper articles collected from different news article providers like the Daily Mail and others. Then like the third dataset, the covid-non-science dataset is also partitioned into multiple sizes, 10 papers(covid-non-science-small), 20 papers(covid-non-science). Each article is separated by the word “NEW NEWSPAPER”.

The fifth dataset is covid-psychology.txt, which consists of covid related psychology research papers collected from multiple sources (ex. IEEE Xplore, Google Scholar). Then like the third dataset, the covid-psychology dataset is also partitioned into multiple sizes, 10 papers(covid-psychology-small), 20 papers(covid-psychology), 40 papers(covid-psychology-large). Each article is separated by the word “NEW PAPER”.

After the data is collected from multiple sources, load the dataset. While loading the data set, the data is divided into multiple levels of documents like article level or paragraph level to perform LSA at multiple levels of the document and extract insights from data at multiple levels to understand how LSA performs to the data at multiple levels of the document.

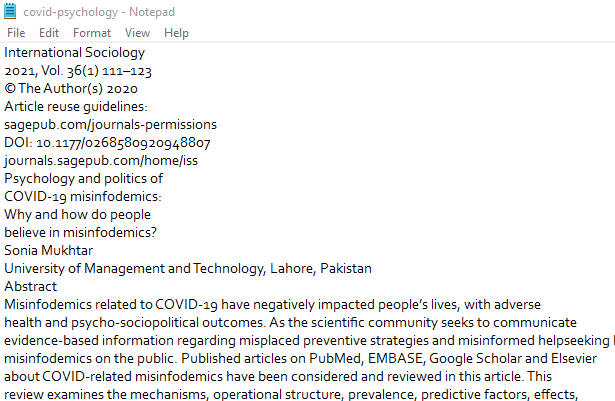


Figure 8 sample data from covid-psychology.txt

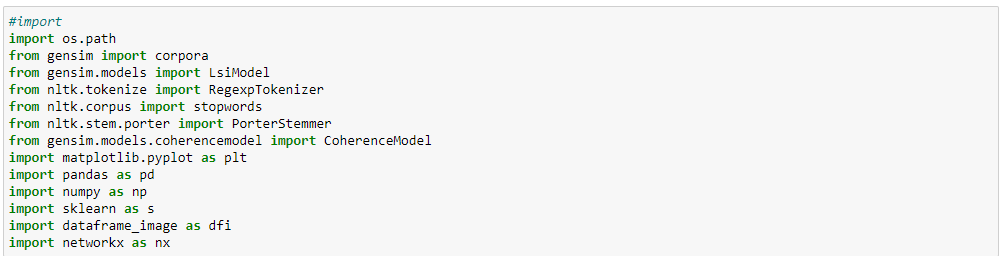


Figure 9 Importing libraries

The first step of writing a code is import libraries. Here in the above code multiple libraries were imported to inherit their classes or methods.



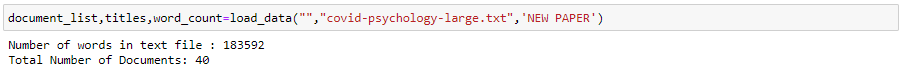


Figure 10 code for loading data at article level from experiment 4 – “covid-science-large.txt” dataset.



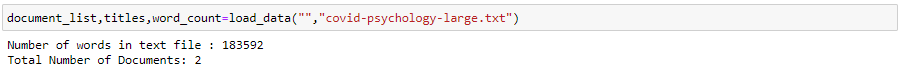


Figure 11 code for loading data at 2-3 paragraph-level from experiment 4 – “covid-science-large.txt” and “covid-psychology-large.txt” dataset

The first 200 words are considered as 2-3 paragraphs from the document.

In the above code the data is loaded by using function *load\_data(path,file\_name).* While loading the data from a text file, the data is divided into multiple documents (one article🡪 one document)

## 4.2 Data Cleaning

The data collected from the different sources contain a lot of noise and might lead to deviated results when experimenting with the data, so to remove the noise multiple pre-processing techniques were applied to the data like tokenizing, removing stop words, and stemming.

Here in this project, three data cleaning techniques were used

|  |  |
| --- | --- |
| **Pre-processing technique** | **Function** |
| Regex tokenizer | RegexpTokenizer(r'\w+') |
| Stop word list | set(stopwords.words('english')) |
| Porter Stemmer | PorterStemmer() |

Graphical user interface, text, application, email

Description automatically generated

Figure 12 Raw data

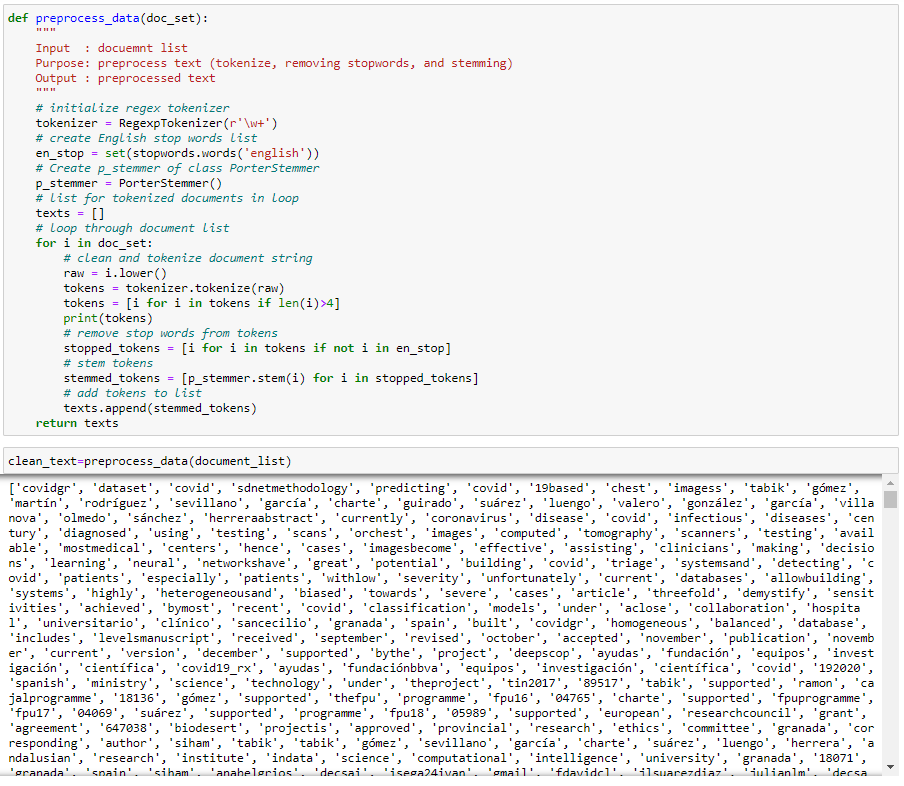


Figure 13 code for Pre-processing and output

The data cleaning techniques are applied to the raw data by function call “preprocess\_data(document list)”. In preprocess\_data function, first the raw text is converted into tokens by using RegexpTokenizer() method. Then the stop words like a, and the etc are removed by using stopwords.words(‘English’) function. Then the potter stemmer is used to remove frequent morphological and inflexional endings from English words. Ex: (play, playing) 🡪 play

## 4.3 Latent Semantic Analysis (LSA)

Preparing corpus by creating the term dictionary for corpus and generating document term matrix to perform LSA using Latent Semantic Indexing model(lsimodel) from python “genism.models” library. And then generating the top 5 words from the corpus.

LSA model:

**from** **gensim.models** **import** LsiModel

model = LsiModel(common\_corpus, id2word=common\_dictionary)

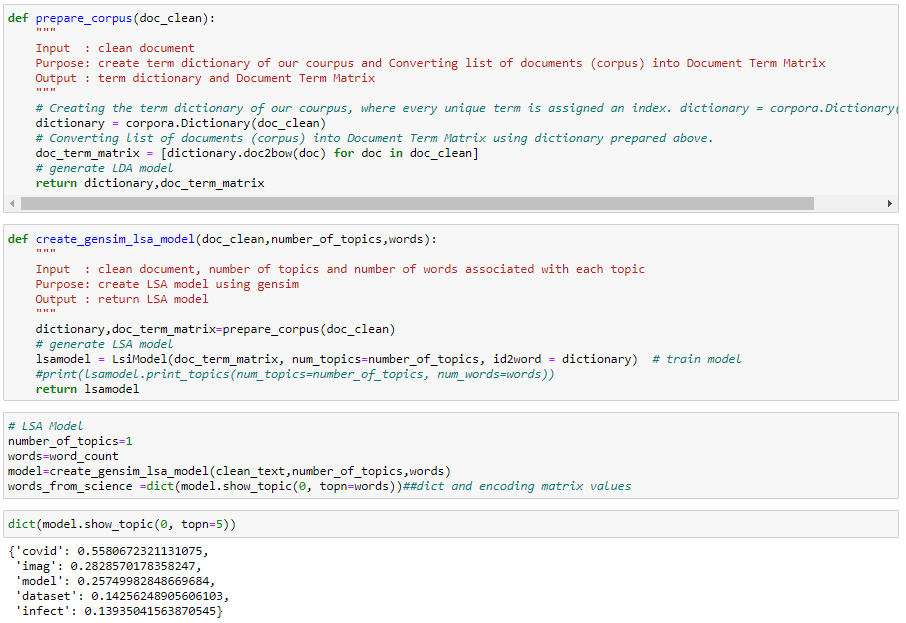


Figure 14 Code for modelling LSA and generating top 5 words at article level from “covid-science-large.txt” dataset.

The first step of LSA is to convert raw data into document term matrix. Here the data is converted to document term matrix by using “prepare\_corpus(doc\_clean)” function. Then in order to use LSA, the LSA model is created using “create\_gensim\_lsa\_model(doc\_clean, number\_of\_topics, words )”. The number of topics parameter is for requesting LSA model to divide the text/words into topics.

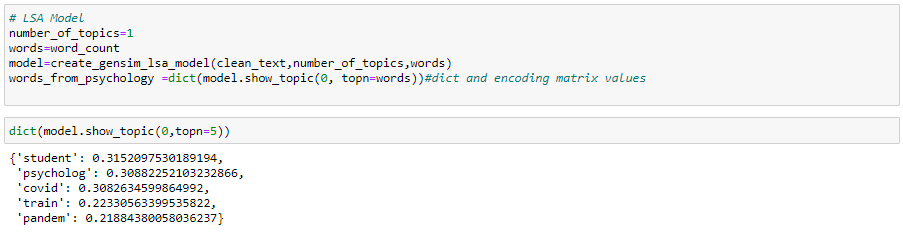


Figure 15 Code for modelling LSA and generating top 5 words at article level from “covid-psychology-large.txt” dataset.

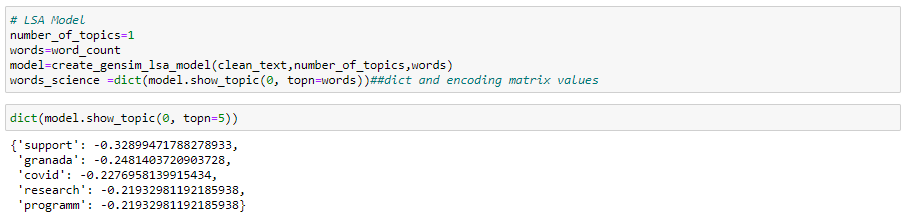


Figure 16 Code for modelling LSA and generating top 5 words at 2-3 paragraph-level from “covid-science-large.txt” dataset.

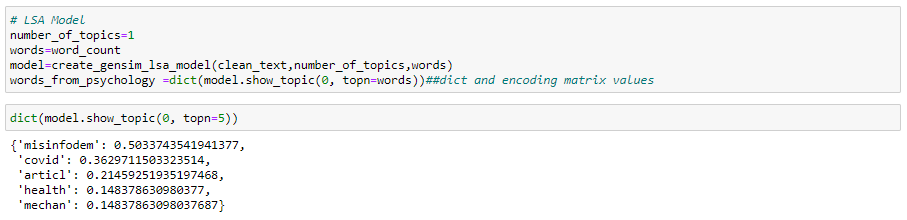


Figure 17 Code for modelling LSA and generating top 5 words at 2-3 paragraph-level from “covid-psychology-large.txt”

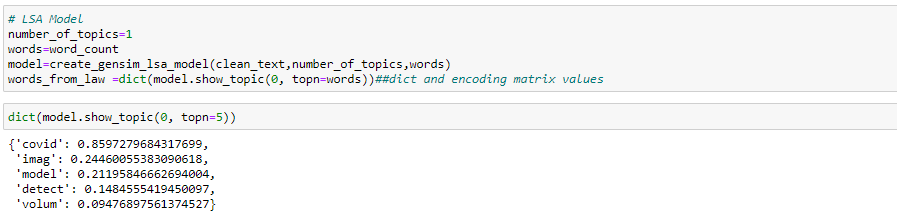


Figure 18 Code for modelling LSA and generating top 5 words at paragraph-level(all paragraphs) from “covid-science-large.txt” dataset.

Graphical user interface, text, application, email

Description automatically generated

Figure 19 Code for modelling LSA and generating top 5 words at paragraph-level (all paragraphs) from “covid-psychology-large.txt” dataset.

**The below table shows the top 5 words and their weights in the document generated by “lsimodel”. The weights for the words are generated by lsi model which represent the frequency of word occurring in the document. Hight weight represent 🡪 high frequency of word occurring in the document vies-verse.**

|  |  |
| --- | --- |
| **law-texts-all** | |
| **Top words** | **weights** |
| **'tariff'** | 0.3329384971656329, |
| **'secretari'** | 0.3314772021127373 |
| **'prison'** | 0.2491591199567464 |
| **'life'** | 0.2361170851923661 |
| **'sentenc'** | 0.2295229637034024 |

Table 1 List of top words from LSA model

## 4.4 Word Cloud

In this step, generating the word cloud for the top 20 words with high weights from the LSA model by using the word cloud application program interface(API) and saving the word cloud to an image.

Function for Word Cloud

|  |  |
| --- | --- |
| [**WordCloud**](https://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.html#wordcloud.WordCloud)([font\_path, width, height, …]) | Word cloud object for generating and drawing. |

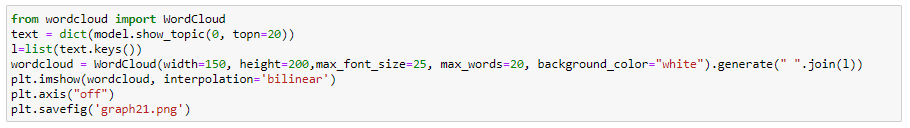


Figure 4.4.1 Code for generating word Cloud

|  |  |
| --- | --- |
| **No. of Documents** | **law-texts-all.txt** |
| 1. **Article level (4 law articles)** | **Text  Description automatically generated** |

Table 2 Word cloud representation of top words

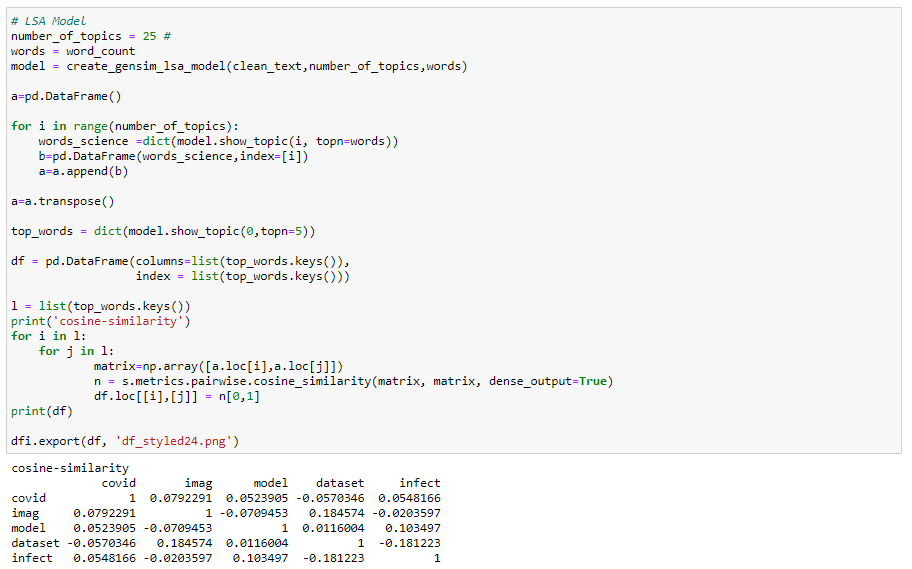
## 4.5 Wordnet

Cosine-similarity is a distance metric used to generate the difference or distance between the words. To generate the distance matrix for the words we use pairwise-cosine-similarity from sklearn. metrics class and to graphically represent and show them in a space we use networkx library.

Networkx:

**import** **networkx** **as** **nx**

**>>>** G = nx.Graph()



|  |  |
| --- | --- |
| **No.of Documents** | **law-texts-all.txt** |
| 1. **Article level (4|4)** | Graphical user interface, text  Description automatically generated |

Figure 4.5.1 Code to generate cosine similarity matrix.

Table 3 Cosine similarity matrix for top 5 words

From the above code by using networkx graphs we can add different colours to the graphs and different widths to the edges representing the relation between the nodes by their weights. In order to establish the connection between the nodes(the top words) the cosine similarity matrix need to be generated using

sklearn.metrics.pairwise.cosine\_similarity(*X*, *Y=None*, *dense\_output=True*)

|  |  |
| --- | --- |
| **Experiment 1 (No. | No.)** | **law-texts-all.txt** |
| 1. **Article level (4|4)** | Diagram  Description automatically generated |

Table 4 WordNet of top 5 words

## 4.6 statistical test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.of Documents** | **covid-science-small** | | **Covid-psychology-small** | |
| **Top words** | **weights** | **Top words** | **weights** |
| **Article level (10|10)** | **'covid'** | 0.46802988644393423, | **'covid'** | 0.6105961140643029, |
| **'model'** | 0.3319825262829552, | **'psycholog'** | 0.31072871004100155, |
| **'imag'** | 0.32671571923604975 | **'pandem'** | 0.25119870827829194, |
| **'class'** | 0.20559425407282925, | **'impact'** | 0.2302237165624363, |
| **'dataset'** | 0.15776417661741118 | **'health'** | 0.17811639545747093 |

Now we conduct the mantel test. In order to conduct the mantel test, need to obtain two pair wise distance matrices x and y form two different spaces. Considering two spaces are science data set and

psychology dataset Like in experiment 2, we have extracted top 5

Table 5 Mantel test with 10-word network

words from science dataset and psychology data set. As those two sets of words are different, first we combined the two sets of words forming 10-word network. Likely,

**‘covid’, ’model’, ‘imag’, ‘class’, ‘dataset’, ‘covid’, ‘psycholog’, ‘pandem’, ‘impact’, ‘health’**

Now generated their respective weights from two spaces. Then we get two sets of top words with similar words but with different weights in different spaces. Then calculated the pairwise distance similarity matrix form 10-word network from each space “dist1”, “dist2”.

Then two distance matrices are passed as parameters to mantel function which returns r statistic, z score, and p value.

Similarly, performed mantel test for vertical sample considering different document levels as different spaces.

10-word network

Table 6 Vertical comparison mantel test

|  |  |  |
| --- | --- | --- |
| **No.of Documents** | **covid-science-small** | |
| **Top words** | **weights** |
| **Article level (10|10)** | **'covid'** | 0.46802988644393423, |
| **'model'** | 0.3319825262829552, |
| **'imag'** | 0.32671571923604975 |
| **'class'** | 0.20559425407282925, |
| **'dataset'** | 0.15776417661741118 |
|  |  |  |
| **2-3 paragraphs** | **'support'** | -0.3289947178827894, |
| **'granada'** | -0.24814037209037246, |
| **'covid'** | -0.2276958139915434, |
| **'programm'** | -0.21932981192185963, |
| **'research'** | -0.21932981192185963 |

Function for mantel test statistic:

**skbio.math.stats.distance.mantel(x, y, method='pearson', permutations=999, alternative='two-sided')**

Code for implementing mantel test: Text

Description automatically generatedText, application

Description automatically generated

A picture containing text

Description automatically generatedTable

Description automatically generated

Figure 20 Code for mantel test

# Chapter 5 Experimental Design and Results:

## 5.1 Experiment Descriptions

4 different experiments were conducted for this research project. Each experiment was conducted in reference to the project implementation and all the four experiments were conducted using different datasets as listed below.

Experiment 1: LSA at the article level, 2-3 paragraph level(200-words), paragraph level (Whole document) for both “law-text-all.txt” and “non-law-all.txt” datasets. Law and Non-Law data sets (each data set consists of 4 articles).

Experiment 2: LSA at the article level, 2-3 paragraph level(200-words), paragraph level (Whole document) for three datasets, “covid-science-small.txt”, “covid-non-science-small.txt” and “covid-psychology-small.txt” containing **10 papers** on each dataset. COVID Science small, Covid non-science small and Covid Psychology small data sets (each data set consists of 10 articles).

Experiment 3: LSA at the article level, 2-3 paragraph level(200-words), paragraph level (Whole document) for three datasets, “covid-science.txt”, “covid-non-science.txt” and “covid-psychology.txt” containing **20 papers** on each dataset. COVID Science, Covid non-science and Covid Psychology data sets (each data set consists of 20 articles)

Experiment 4: LSA at the article level, 2-3 paragraph level(200-words), paragraph level (Whole document) for three datasets, “covid-science-large.txt” and “covid-psychology-large.txt” containing **40 papers** on each dataset. COVID Science large, Covid non-science large and Covid Psychology large data sets (each data set consists of 40 articles).

And 2 more experiments were conducted later to understand mantel test results and to eliminate the large variation dataset sizes between the above 4 experiments. In these two experiments, considered only Covid Science and Psychology datasets.

Experiment A1 Covid Science\_15 and Covid Psychology\_15 datasets (each data set consists of 15 articles)

Experiment A2 Covid Science\_15 and Covid Psychology\_15 datasets (each data set consists of 30 articles)

## 5.2 Results

**Top 5 words with high weights generated from the datasets(documents) using model LSI (Latent Semantic Indexing).**

**The below table shows the top 5 words and their weights in the document generated by “lsimodel”. The weights for the words are generated by lsi model which represent the frequency of word occurring in the document. Hight weight represent 🡪 high frequency of word occurring in the document vies-verse.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment 1: Document at various levels for short dataset (law: non-law)** | | | | |
| **No.of Documents** | **law-texts-all** | | **non-law-all** | |
| **Top words** | **weights** | **Top words** | **weights** |
| 1. **Article level (4 law /4 non law articles)** | **'tariff'** | 0.3329384971656329, | **'judg'** | 0.3207357836732813 |
| **'secretari'** | 0.3314772021127373 | **'evid'** | 0.3092444069593179 |
| **'prison'** | 0.2491591199567464 | **'west'** | 0.2804028170023648 |
| **'life'** | 0.2361170851923661 | **'trial'** | 0.2772485745197205 |
| **'sentenc'** | 0.2295229637034024 | **'case'** | 0.2281040582305726 |
|  |  |  |  |  |
| 1. **2-3 paragraphs** | **'tariff'** | 0.2852867613391 | **'life'** | 0.424625074653 |
| **'life'** | 0.245096194271 | **'prison'** | 0.366156436155 |
| **'prison'** | 0.2450961942716 | **'sentenc'** | 0.291549439087 |
| **'would'** | 0.2419339607783 | **'hindley'** | 0.257685215145 |
| **'sentenc'** | 0.2041150688308 | **'posit'** | 0.233080800590 |
|  |  |  |  |  |
| 1. **Paragraph level**   **(450 law/ 3038 non law paragraphs)** | **'tariff'** | -0.372423678930 | **'hindley'** | -0.63764826639 |
| **'secretari'** | -0.325658507266 | **'myra'** | -0.58632670557 |
| **'prison'** | -0.262324361300 | **'west'** | -0.23136661906 |
| **'state'** | -0.246189129218 | **'case'** | -0.18241034492 |
| **'life'** | -0.234305014747 | **'trial'** | -0.14821551842 |

Table 7 List of top words from LSA model experiment 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Experiment 2: Document at various levels for a small dataset (COVID-science: non-science: Covid-psychology)** | | | | | | |
| **No.of Documents** | **covid-science-small** | | **Covid-non-science-small** | | **Covid-psychology-small** | |
| **Top words** | **weights** | **Top words** | **weights** | **Top words** | **weights** |
| **Article level**  **(10 articles each)** | **'covid'** | 0.46802988644393423, | **'test'** | 0.6105961140643029, | **'covid'** | 0.41992720259455585, |
| **'model'** | 0.3319825262829552, | **'peopl'** | 0.31072871004100155, | **'psycholog'** | 0.28694986504872455, |
| **'imag'** | 0.32671571923604975 | **'covid'** | 0.25119870827829194, | **'pandem'** | 0.24145037935803, |
| **'class'** | 0.20559425407282925, | **'symptom'** | 0.2302237165624363, | **'impact'** | 0.21712822111271726, |
| **'dataset'** | 0.15776417661741118 | **'fever'** | 0.17811639545747093 | **'health'** | 0.19835033050070103 |
|  |  |  |  |  |  |  |
| **2-3 paragraphs** | **'support'** | -0.3289947178827894, | **'market'** | 0.2986573559242644, | **'misinfodem'** | 0.5033743541941375, |
| **'granada'** | -0.24814037209037246, | **'covid'** | 0.21979867961934094, | **'covid'** | 0.36297115033235156, |
| **'covid'** | -0.2276958139915434, | **'friday'** | 0.18596179123202228, | **'articl'** | 0.21459251935197457, |
| **'programm'** | -0.21932981192185963, | **'infect'** | 0.15212490284470373, | **'commun'** | 0.14837863098037715, |
| **'research'** | -0.21932981192185963 | **'copper'** | 0.14653245307956103 | **'health'** | 0.14837863098037676 |
|  |  |  |  |  |  |  |
| **Paragraph level (7401 science|229 non-science)** | **'covid'** | 0.7920442330385795, | **'trial'** | 0.4673333939365011, | **'pandem'** | 0.3492162493102458, |
| **'model'** | 0.2953962153046564, | **'vaccin'** | -0.3789496907239388, | **'polici'** | 0.33206567897289235, |
| **'imag'** | 0.2745133138045724, | **'stage'** | -0.33285475306510043, | **'health'** | 0.29289264374192664, |
| **'detect'** | 0.16365565281326466, | **'expect'** | -0.2332697529258982, | **'covid'** | 0.26562150610270957, |
| **'class'** | 0.12952267447918445 | **'underway'** | -0.20015564175913142 | **'impact'** | 0.2463281943440908 |

Table 8 List of top words from LSA model experiment 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Experiment 3: Document at various levels for large dataset (20 papers COVID-science: non-science: Covid-psychology)** | | | | | | |
| **No.of Documents** | **covid-science.txt** | | **Covid-non-science.txt** | | **Covid-psychology.txt** | |
| **Top words** | **weights** | **Top words** | **weights** | **Top words** | **weights** |
| **Article level (20 science|20 non-science articles)** | **'covid'** | 0.5195000201940321, | **'test'** | 0.5525012615032745, | **'covid'** | 0.3733564956224059, |
| **'imag'** | 0.3819419924959906, | **'peopl'** | 0.32711032908806575, | **'psycholog'** | 0.26785991090419736, |
| **'model'** | 0.24414357176752507, | **'covid'** | 0.32350304714824757, | **'social'** | 0.2517797817165504, |
| **'dataset'** | 0.17846130209695354, | **'symptom'** | 0.20147297742697395, | **'epidem'** | 0.19223792506781157, |
| **'class'** | 0.15297059818817663 | **'travel'** | 0.15909033106510667 | **'health'** | 0.18194229083665558 |
|  |  |  |  |  |  |  |
| **2-3 paragraphs** | **'support'** | 0.32899471788278933, | **'market'** | 0.2986573559242641, | **'misinfodem'** | -0.5033743541941365, |
| **'granada'** | 0.24814037209037237, | **'covid'** | 0.2197986796193419, | **'covid'** | -0.3629711503323515, |
| **'covid'** | 0.22769581399154337, | **'friday'** | 0.18596179123202256, | **'articl'** | -0.2145925193519742, |
| **'programm'** | 0.21932981192185955, | **'infect'** | 'infect': 0.15212490284470295, | **'commun'** | -0.14837863098037748, |
| **'research'** | 0.21932981192185955 | **'reuter'** | 0.1465324530795613 | **'mechan'** | -0.14837863098037712 |
|  |  |  |  |  |  |  |
| **Paragraph level (14379 science|387non-science paragraphs)** | **'covid'** | 0.8353445537976227, | **'trial'** | -0.46149246082032147, | **'covid'** | 0.44528765261781605, |
| **'imag'** | 0.3551934971949819, | **'vaccin'** | -0.38111970491120056, | **'pandem'** | 0.3598378349771223, |
| **'detect'** | 0.14217453078608513, | **'stage'** | -0.3279128249788164, | **'health'** | 0.33461199438050226, |
| **'model'** | 0.1267002478578127, | **'expect'** | -0.23283573850743006, | **'polici'** | 0.2627823706235591, |
| **'dataset'** | 0.1093369812928102 | **'dose'** | -0.19988003539528476 | **'impact'** | .2152347635291778 |

Table 9 List of top words from LSA model experiment 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment 4: Document at various levels for large dataset (40 papers COVID-science: Covid-psychology)** | | | | |
|  | | | | |
| **No.of Documents** | **covid-science-large.txt** | | **Covid-psychology-large.txt** | |
| **Top words** | **weights** | **Top words** | **weights** |
| **Article level (40|40)** | **'covid'** | 0.5580672321131127, | **'student'** | 0.31520975301891685, |
| **'imag'** | 0.2828570178358147, | **'psycholog'** | 0.3088225210323284, |
| **'model'** | 0.2574998284866968, | **'covid'** | 0.3082634599864977, |
| **'dataset'** | 0.1425624890560568, | **'train'** | 0.22330563399536046, |
| **'infect'** | 0.13935041563870734 | **'pandem'** | 0.21884380058036093 |
|  |  |  |  |  |
| **2-3 paragraphs** | **'support'** | -0.32899471788278967, | **'misinfodem'** | 0.5033743541941371, |
| **'granada'** | -0.24814037209037254, | **'covid'** | 0.3629711503323513, |
| **'covid'** | -0.22769581399154254, | **'articl'** | 0.21459251935197415, |
| **'programm'** | -0.21932981192185977, | **'mechan'** | 0.14837863098037712, |
| **'research'** | -0.21932981192185977 | **'commun'** | 0.14837863098037704 |
|  |  |  |  |  |
| **Paragraph level (274941|183592)** | **'covid'** | 0.8597188592513563, | **'covid'** | 0.5542346968279828, |
| **'imag'** | 0.24459808570501076, | **'psycholog'** | 0.38587594842118456, |
| **'model'** | 0.21196118253925922, | **'pandem'** | 0.36424816089766787, |
| **'detect'** | 0.14846685076222263, | **'health'** | 0.2988945427299109, |
| **'volum'** | 0.09486963239930264 | **'impact'** | 0.15508253422230434 |

Table 10 List of top words from LSA model experiment 4

**Results from Word Cloud for top 20 words from 4 experiments**

|  |  |  |
| --- | --- | --- |
| **Experiment 1: Document at various levels for short dataset (law: non-law)** | | |
| **No. of Documents** | **law-texts-all.txt** | **non-law-all.txt** |
| 1. **Article level (4 law|4 non-law articles)** | **Text  Description automatically generated** | **Text  Description automatically generated** |
| 1. **2-3 paragraphs** | Text  Description automatically generated | Text  Description automatically generated |
| 1. **Paragraph level (450 law|3038 non-law paragraphs)** | Text  Description automatically generated | Text  Description automatically generated |

Table 11 Word cloud representation of top words from Experiment 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment 2: Document at various levels for a small dataset (COVID-science: non-science: Covid-psychology)** | | | |
| **No. of Documents** | **covid-science-small.txt** | **Covid-non-science-small.txt** | **Covid-psychology-small.txt** |
| **Article level (10 science|10 non-science articles)** | Text  Description automatically generated | Text  Description automatically generated | Text  Description automatically generated |
| **2-3 paragraphs** | Text  Description automatically generated | Text  Description automatically generated | Text  Description automatically generated |
| **Paragraph level (7401 science|229 non-science paragraphs)** | Text  Description automatically generated | Text  Description automatically generated with medium confidence | Text  Description automatically generated |

Table 12 Word cloud representation of top words from Experiment 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment 3: Document at various levels for large dataset (20 papers COVID-science: non-science: Covid-psychology)** | | | |
| **No. of Documents** | **covid-science.txt** | **Covid-non-science.txt** | **Covid-psychology.txt** |
| **Article level (20|20)** | Text  Description automatically generated | Text  Description automatically generated with medium confidence | Text  Description automatically generated |
| **2-3 paragraphs** | Text, qr code  Description automatically generated | Text  Description automatically generated | Text  Description automatically generated |
| **Paragraph level (14379|387)** | Text  Description automatically generated | Text  Description automatically generated | Text  Description automatically generated |

Table 13 Word cloud representation of top words from Experiment 3

|  |  |  |
| --- | --- | --- |
| **Experiment 4: Document at various levels for large dataset (40 papers COVID-science: Covid-psychology)** | | |
| **No. of Documents** | **covid-science-large.txt** | **Covid-psychology-large.txt** |
| **Article level (40 science|40 non-science articles)** | Text  Description automatically generated | Text  Description automatically generated |
| **2-3 paragraphs** | Text, qr code  Description automatically generated | Text  Description automatically generated |
| **Paragraph level (274941 science|183592 non-science paragraphs)** | Text  Description automatically generated | Text  Description automatically generated |

Table 14 Word cloud representation of top words from Experiment 4

**Wordnet Results (NetworkX graph for top 5 words) from 4 experiments**

|  |  |  |
| --- | --- | --- |
| Experiment 1: Document at various levels for short dataset (law: non-law) | | |
| **No. of Documents** | **law-texts-all.txt** | **non-law-all.txt** |
| 1. **Article level (4 law|4 non-law articles)** | Graphical user interface, text  Description automatically generated | A screenshot of a computer  Description automatically generated with medium confidence |
| 1. **2-3 paragraphs** | Graphical user interface, text, application, chat or text message  Description automatically generated | Graphical user interface  Description automatically generated |
| 1. **Paragraph level (450 law|3038 non-law paragraphs)** | Text  Description automatically generated | A picture containing background pattern  Description automatically generated |

Table 15 Cosine Similarity matrices for Experiment 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment 2: Document at various levels for a small dataset (COVID-science: non-science: Covid-psychology)** | | | |
| **No. of Documents** | **covid-science-small.txt** | **Covid-non-science-small.txt** | **Covid-psychology-small.txt** |
| **Article level (10 science|10 non-science articles)** | Graphical user interface, text  Description automatically generated | Graphical user interface, text  Description automatically generated | Graphical user interface, text, application, chat or text message  Description automatically generated |
| **2-3 paragraphs** | Graphical user interface, text, application  Description automatically generated | Graphical user interface, text  Description automatically generated | Graphical user interface, text  Description automatically generated |
| **Paragraph level (7401 science|229 non-science paragraphs)** | Graphical user interface, text  Description automatically generated | Graphical user interface, text  Description automatically generated | Graphical user interface, text, application, chat or text message  Description automatically generated |

Table 16 Cosine Similarity matrices for Experiment 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment 3: Document at various levels for large dataset (20 papers COVID-science: non-science: Covid-psychology)** | | | |
| **No. of Documents** | **covid-science.txt** | **Covid-non-science.txt** | **Covid-psychology.txt** |
| **Article level (20 science|20 non-science articles)** | Graphical user interface, text  Description automatically generated | Graphical user interface, text, application  Description automatically generated | Graphical user interface, text, application  Description automatically generated |
| **2-3 paragraphs** | Graphical user interface, text, application  Description automatically generated | Graphical user interface, text  Description automatically generated | Graphical user interface, text, application  Description automatically generated |
| **Paragraph level (14379 science|387 non-science paragraphs)** | Graphical user interface, text  Description automatically generated | Graphical user interface, text  Description automatically generated | Graphical user interface, text, application  Description automatically generated |

Table 17 Cosine Similarity matrices for Experiment 3

|  |  |  |
| --- | --- | --- |
| **Experiment 4: Document at various levels for large dataset (40 papers COVID-science: Covid-psychology)** | | |
| **No.of Document** | **covid-science-large.txt** | **Covid-psychology-large.txt** |
| **Article level (40 science|40 non-science articles)** | Graphical user interface, text, chat or text message  Description automatically generated | Graphical user interface, text, application  Description automatically generated |
| **2-3 paragraphs** | Graphical user interface, text, application  Description automatically generated | Graphical user interface, text  Description automatically generated |
| **Paragraph level (274941 science|183592 non-science paragraphs)** | Graphical user interface, text, application, chat or text message  Description automatically generated | Graphical user interface, text, application  Description automatically generated |

Table 18 Cosine Similarity matrices for Experiment 4

|  |  |  |
| --- | --- | --- |
| **Experiment 1: Document at various level for short dataset (law: non-law)** | | |
| **Experiment 1 (No. | No.)** | **law-texts-all.txt** | **non-law-all.txt** |
| 1. **Article level (4|4)** | Diagram  Description automatically generated | Diagram  Description automatically generated |
| 1. **2-3 paragraphs** | Diagram  Description automatically generated | Chart, diagram  Description automatically generated |
| 1. **Paragraph level (450|3038)** | Chart, diagram  Description automatically generated | Chart, diagram  Description automatically generated |

**NetworkX graphs for 4 experiments**

Table 19 Wordnet of top 5 words form experiment 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment 2: Document at various level for small dataset (COVID-science: non-science: Covid-psychology)** | | | |
| **Experiment 2 (No. | No.)** | **covid-science-small.txt** | **Covid-non-science-small.txt** | **Covid-psychology-small.txt** |
| **Article level (10|10)** | Diagram  Description automatically generated | Chart, diagram, radar chart  Description automatically generated | Chart, radar chart  Description automatically generated |
| **2-3 paragraphs** | Diagram  Description automatically generated | Diagram, radar chart  Description automatically generated | Chart, radar chart  Description automatically generated |
| **Paragraph level (7401|229)** | Chart, radar chart  Description automatically generated with medium confidence | Diagram  Description automatically generated | Chart  Description automatically generated |

Table 20 Wordnet of top 5 words form experiment 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment 3: Document at various level for large dataset (20 papers COVID-science: non-science: Covid-psychology)** | | | |
| **Experiment 3 (No. | No.)** | **covid-science.txt** | **Covid-non-science.txt** | **Covid-psychology.txt** |
| **Article level (20|20)** | Diagram  Description automatically generated | Chart, diagram, radar chart  Description automatically generated | Chart, diagram  Description automatically generated with medium confidence |
| **2-3 paragraphs** | Diagram  Description automatically generated | Diagram  Description automatically generated | Chart, radar chart  Description automatically generated |
| **Paragraph level (14379|387)** | Diagram  Description automatically generated | Chart, diagram, radar chart  Description automatically generated | Radar chart  Description automatically generated |

Table 21 wordnet of top 5 words form experiment 3

|  |  |  |
| --- | --- | --- |
| **Experiment 4: Document at various level for large dataset (40 papers COVID-science: Covid-psychology)** | | |
| **Experiment 4 (No. | No.)** | **covid-science-large.txt** | **Covid-psychology-large.txt** |
| **Article level (40|40)** | Diagram  Description automatically generated | Chart  Description automatically generated |
| **2-3 paragraphs** | Diagram  Description automatically generated | Chart, radar chart  Description automatically generated |
| **Paragraph level (274941|183592)** | Diagram  Description automatically generated with medium confidence | Diagram  Description automatically generated with medium confidence |

Table 22 wordnet of top 5 words form experiment 4

**Mantel Test Results**

Mantel test results for top words in two spaces. The environmental spaces are given below

mantel (X’, Y’)

where X’, Y’ are pairwise distances between top words respectively.

X(Space 1) Y(Space 2)

Law dataset Nonlaw dataset

Covid science Psychology (40 articles)

Covid science Psychology (10 articles)

Covid science Psychology (20 articles)

Results generated from mantel test with 10 words network and 20-word network

10-word network: By comparing Top 5 words from each dataset

20-word network: By comparing Top 10 words from each dataset

Mantel-Test for Law and Non-Law datasets (4 articles dataset)

Table Mantel-Test results for Law and Non-Law datasets (4 articles dataset)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article Level | 0.45342 | 0.0127 | 2.935799 | 0.12178 | 0.1489 | 1.06275 |
| 2-3 Paragraphs | -0.21510 | 0.714285 | -0.68497 | 0.06725 | 0.3282 | 0.40405 |
| Paragraph level | -0.29385 | 0.878 | -1.22018 | -0.31741 | 0.9493 | -1.71181 |

Mantel test for science – psychology (10 articles dataset)

Table Mantel test results for COVID science – psychology (10 articles dataset)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article Level | -0.31881 | 0.87864 | -1.20322 | -0.37001 | 0.96258 | -1.69272 |
| 2-3 Paragraphs | -0.12443 | 0.6625 | -0.52720 | -0.16750 | 0.88753 | -1.14787 |
| Paragraph level | -0.32813 | 0.8417 | -1.00066 | -0.07063 | 0.61801 | -0.32868 |

Mantel test for science – psychology (15 articles dataset)

Table Mantel test results for COVID science – psychology (15 articles dataset)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article Level | -0.55832 | 0.9722 | -1.76688 | -0.41870 | 0.9848 | -1.91222 |
| 2-3 Paragraphs | -0.12443 | 0.6549 | -0.51143 | -0.15851 | 0.8535 | -1.04880 |
| Paragraph level | -0.36722 | 0.8156 | -1.09169 | -0.47758 | 0.9934 | -2.17452 |

Mantel test for science – psychology (20 articles dataset)

Table Mantel test results for COVID science – psychology (20 articles dataset)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article Level | -0.45097 | 0.92318 | -1.40863 | -0.45208 | 0.99278 | -2.32042 |
| 2-3 Paragraphs | -0.12443 | 0.65294 | -0.51520 | -0.15851 | 0.85532 | -1.05703 |
| Paragraph level | -0.47229 | 0.92942 | -1.43762 | -0.26062 | 0.88279 | -1.19640 |

Mantel test for science – psychology (30 articles dataset)

Table Mantel test results for COVID science – psychology (30 articles dataset)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article Level | -0.44085 | 0.9074 | -1.33729 | -0.13052 | 0.7165 | -0.58437 |
| 2-3 Paragraphs | -0.12443 | 0.6558 | -0.51814 | -0.16750 | 0.8834 | -1.15334 |
| Paragraph level | -0.42116 | 0.9493 | -1.29004 | -0.36792 | 0.9549 | -1.65607 |

Mantel test for science – psychology (40 articles dataset)

Table Mantel test results for COVID science – psychology (40 articles dataset)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article Level | -0.10233 | 0.6703 | -0.45536 | 0.16347 | 0.3104 | 0.5929 |
| 2-3 Paragraphs | -0.16750 | 0.8881 | -1.14848 | -0.2556 | 1.0 | -1.1348 |
| Paragraph level | -0.29198 | 0.9005 | -1.29744 | -0.04745 | 0.5642 | -0.17695 |

Mantel test results for top words in two spaces. The environmental spaces are given below

Mantel test results (Vertical comparison of spaces within the same dataset)

X (space 1) Y (space 2)

Article 2-3 p

Article paragraph

2-3 p paragraph

mantel (X”, Y”)

where X”, Y” are pairwise distances between top words respectively.

For each dataset Law dataset, non-Law dataset, Covid science dataset (40 articles and 20 articles), Covid psychology dataset (40 articles and 20 articles.

Results generated from mantel test with 10 words network and 20-word network

10-word network: By comparing Top 5 words from each dataset

20-word network: By comparing Top 10 words from each dataset

Law dataset

Table Mantel test results for law dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | -0.09916 | 0.5375 | -0.36904 | -0.16939 | 0.9614 | -1.58676 |
| Article ---> Paragraph | 0.30533 | 0.183333 | 1.0028 | 0.29004 | 0.1036 | 1.44017 |
| 2-3 P ---> Paragraph | 0.21549 | 0.14583 | 1.21009 | 0.19494 | 0.0558 | 1.68612 |

Non-Law dataset

Table mantel test results for non-law dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | 0.08840 | 0.2401 | 0.72033 | 0.08802 | 0.2293 | 0.78101 |
| Article ---> Paragraph | 0.43517 | 0.01626 | 1.81199 | 0.55781 | 0.001 | 2.73972 |
| 2-3 P ---> Paragraph | -0.24308 | 0.7751 | -0.8898 | -0.11016 | 0.7182 | -0.56198 |

Covid science dataset (15 articles)

Table mantel test results for Covid science dataset with 15 articles dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | 0.27763 | 0.1765, | 1.06224 | 0.13258 | 0.2438 | 0.72441 |
| Article ---> Paragraph | 0.69915 | 0.01805 | 1.78310 | 0.49915 | 0.0257 | 1.66534 |
| 2-3 P ---> Paragraph | 0.22988 | 0.1795 | 1.02993 | 0.02976 | 0.4411 | 0.18913 |

Covid psychology dataset (15 articles)

Table mantel test results for Covid psychology dataset with 15 articles dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | -0.63556 | 0.9678 | -2.00749 | -0.28840 | 0.9284 | -1.47024 |
| Article ---> Paragraph | 0.37749 | 0.17301 | 0.94661 | 0.43386 | 0.0024 | 1.85060 |
| 2-3 P ---> Paragraph | -0.02156 | 0.4867 | -0.06129 | -0.22712 | 0.8713 | -1.17128 |

Covid science dataset (20 articles)

Table mantel test results for Covid science dataset with 20 articles

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | 0.28155 | 0.1844 | 1.04480 | 0.18832 | 0.1602 | 1.03448 |
| Article ---> Paragraph | 0.78462 | 0.02916 | 1.83809 | 0.48517 | 0.0043 | 1.67499 |
| 2-3 P ---> Paragraph | 0.26279 | 0.1414 | 1.17312 | -0.00810 | 0.5304 | -0.05190 |

Covid psychology dataset (20 articles)

Table mantel test results for Covid psychology dataset with 20 articles dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | -0.12292 | 0.6418 | -0.38989 | -0.22976 | 0.873 | -1.17834 |
| Article ---> Paragraph | 0.65826 | 0.0169 | 1.93129 | 0.69237 | 0.0004 | 2.99234 |
| 2-3 P ---> Paragraph | 0.02425 | 0.481 | 0.06582 | 0.05196 | 0.412 | 0.25059 |

Covid science dataset (30 articles)

Table mantel test results for Covid science dataset with 30 articles

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | 0.27763 | 0.1765 | 1.06224 | 0.13258 | 0.2438 | 0.72441 |
| Article ---> Paragraph | 0.69915 | 0.01805 | 1.78310 | 0.49915 | 0.0257 | 1.66534 |
| 2-3 P ---> Paragraph | 0.22988 | 0.1795 | 1.02993 | 0.02976 | 0.4411 | 0.18913 |

Covid psychology dataset (30 articles)

Table mantel test results for Covid psychology dataset with 30 articles dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | -0.63556 | 0.9678 | -0.38989 | -0.28840 | 0.9284 | -1.47024 |
| Article ---> Paragraph | 0.37749 | 0.17301 | 0.94661 | 0.43386 | 0.0024 | 1.85060 |
| 2-3 P ---> Paragraph | -0.02156 | 0.4867 | -0.06129 | -0.22712 | 0.8713 | -1.17128 |

Covid science dataset (40 articles)

Table mantel test results for Covid science dataset with 40 articles dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | 0.150793 | 0.3091 | 0.57597 | 0.10640 | 0.284 | 0.59211 |
| Article ---> Paragraph | 0.76665 | 0.0047 | 1.93097 | 0.53417 | 0.0091 | 1.80730 |
| 2-3 P ---> Paragraph | 0.18428 | 0.2865 | 0.67298 | 0.00396 | 0.5062 | 0.01858 |

Covid psychology dataset (40 articles)

Table mantel test results for Covid psychology dataset with 40 articles dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mantel Test | 10-word network | | | 20-word network | | |
| r | p-value | Z-score | r | p-value | z-score |
| Article ---> 2-3 P | -0.48586 | 0.9388 | -1.62018 | -0.35639 | 0.9624 | -1.80561 |
| Article ---> Paragraph | 0.23856 | 0.30337 | 0.60114 | 0.51077 | 0.0013 | 1.96954 |
| 2-3 P ---> Paragraph | -0.25881 | 0.762 | -0.87218 | -0.22467 | 0.8612 | -1.11457 |

### 5.2.A Mantel Test Results Comparison (Pyplots)

Chart, line chart

Description automatically generated

Figure 21 P-values for Mantel test with 10-word network

Chart, line chart

Description automatically generated

Figure 22 r-statistic for Mantel test with 10-word network

Chart, line chart

Description automatically generated

Figure 23 Z-score for Mantel test with 10-word network

Chart, line chart

Description automatically generated

Figure 24 P-values for Mantel test with 20-word network

Chart, line chart

Description automatically generated

Figure 25 R-values for Mantel test with 20-word network

Chart, line chart

Description automatically generated

Figure 26 z-score for Mantel test with 20-word network

# Chapter 6 Conclusion

Our experiment results demonstrate that the semantic meaning of words on the same topic might vary across different subject areas (e.g., science and psychology).

In conclusion, from this project experiments the words extracted from two different corpuses (2 different subject areas data) using Latent Semantic Analysis (LSA) demonstrates a clear semantic difference between two types of corpuses, For example, from the four experiments conducted the semantic difference between the COVID Science articles and psychological articles can be derived by words generated by LSA model. But the results vary with the size of the data set and the consideration of level of the document in LSA (Ex: Article level, Paragraph level, n-paragraph level).

From the results the p value is much consistent at paragraph level when compared to article level and 2-3 paragraph level. And shows significant difference between the two datasets for 10-word network (p-values: 0.84, 0.82, 0.92, 0.95, 0.90) and at 20-word network (p-values: 0.62, 0.98, 0.88, 0.95, 0.56). Which means the sample statistics is in the right extreme of the distribution. Most of the test results obtained from 2-3 paragraph level were randomly distributed. The statistical results obtained from 2-3 paragraphs are almost like one-another because the words generated from the articles did not change as the size of the article was changed. At article level the p-values (Ex: p-values at article level for 10-word network 0.88(10 articles), 0.97(15 articles), 0.92(20 articles), 0.95(30 articles), 0.67(40 articles)) are consistent until the large data set (40 articles).

The datasets with 15, 20, 30 articles with 20-word network (p-values: 0.97 - 15 articles, 0.92 - 20 articles, 0.95 - 30 articles) demonstrate clear semantic difference between scientific articles and psychology articles.

Latent Semantic Analysis and Mantel test identify the statistical difference between the semantic meaning of words in different subject areas.

# References

Altszyler, E., Sigman, M., Ribeiro, S., Slezak, D.F., 2017. Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. Consciousness and Cognition 56, 178–187. https://doi.org/10.1016/j.concog.2017.09.004

Anwar, A., 2006. Experiments with latent semantic analysis for word tagging, in: Proceedings of the 5th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing, CSECS’06. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 275–278.

Ashby, W.R., 1961. An introduction to cybernetics. Chapman & Hall Ltd.

Bates, M.J., 2019. Understanding information retrieval systems: management, types, and standards. Auerbach Publications.

Besançon, R., Rajman, M., 2002. Evaluation of a Vector Space Similarity Measure in a Multilingual Framework.

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022.

Broeck, J.V. den, Cunningham, S.A., Eeckels, R., Herbst, K., 2005. Data Cleaning: Detecting, Diagnosing, and Editing Data Abnormalities. PLOS Medicine 2, e267. https://doi.org/10.1371/journal.pmed.0020267

Brooks-Gordon, B., Freeman, M., 2006. Law and Psychology: Current Legal Issues. Oxford University Press UK.

Campbell, M.J., 2021. Statistics at Square One. John Wiley & Sons.

Cui, W., Wu, Y., Liu, S., Wei, F., Zhou, M.X., Qu, H., 2010. Context preserving dynamic word cloud visualization, in: 2010 IEEE Pacific Visualization Symposium (PacificVis). Presented at the 2010 IEEE Pacific Visualization Symposium (PacificVis), pp. 121–128. https://doi.org/10.1109/PACIFICVIS.2010.5429600

Dascalu, M., McNamara, D., 2017. How Important Is Size? An Investigation of Corpus Size and Meaning in both Latent Semantic Analysis and Latent Dirichlet Allocation.

Databricks Academy, 2019. Introduction to Latent Semantic Analysis (1/5).

Dennis, S., Dennis@colorado, S., Edu, 2003. A comparison of statistical models for the extraction of lexical information from text corpora.

Diniz-Filho, J.A.F., Soares, T.N., Lima, J.S., Dobrovolski, R., Landeiro, V.L., de Campos Telles, M.P., Rangel, T.F., Bini, L.M., 2013. Mantel test in population genetics. Genet Mol Biol 36, 475–485. https://doi.org/10.1590/S1415-47572013000400002

Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S., Harshman, R., 1988. Using latent semantic analysis to improve access to textual information, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’88. Association for Computing Machinery, New York, NY, USA, pp. 281–285. https://doi.org/10.1145/57167.57214

Foltz, P.W., 1996. Latent semantic analysis for text-based research. Behavior Research Methods, Instruments, & Computers 28, 197–202. https://doi.org/10.3758/BF03204765

Gensim: topic modelling for humans [WWW Document], n.d. URL https://radimrehurek.com/gensim/models/lsimodel.html (accessed 10.26.21).

Gupta, A., Johnson, B., Nagar, D., 2013. Testing Equality of Several Correlation Matrices. Revista Colombiana de Estadística 36.

Hagberg, A., Swart, P., S Chult, D., 2008. Exploring network structure, dynamics, and function using networkx (No. LA-UR-08-05495; LA-UR-08-5495). Los Alamos National Lab. (LANL), Los Alamos, NM (United States).

Ibrahimov, O., Sethi, I., Dimitrova, N., 2002. The performance analysis of a Chi-square similarity measure for topic related clustering of noisy transcripts, in: Object Recognition Supported by User Interaction for Service Robots. Presented at the 16th International Conference on Pattern Recognition, IEEE Comput. Soc, Quebec City, Que., Canada, pp. 285–288. https://doi.org/10.1109/ICPR.2002.1047452

Kabir, S.M., 2016. METHODS OF DATA COLLECTION. pp. 201–275.

Khademi, A., Hosseinzadeh, D., Venetsanopoulos, A., Moody, A., 2009. Nonparametric statistical tests for exploration of correlation and nonstationarity in images, in: 2009 16th International Conference on Digital Signal Processing. Presented at the 2009 16th International Conference on Digital Signal Processing, pp. 1–6. https://doi.org/10.1109/ICDSP.2009.5201186

Kintsch, W., 2001. Predication. Cognitive Science 25, 173–202. https://doi.org/10.1207/s15516709cog2502\_1

Kontostathis, A., Pottenger, W., 2002. Detecting Patterns in the LSI Term-Term Matrix. undefined.

Kwon, H., Park, Y., 2018. Proactive development of emerging technology in a socially responsible manner: Data-driven problem solving process using latent semantic analysis. Journal of Engineering and Technology Management 50, 45–60. https://doi.org/10.1016/j.jengtecman.2018.10.001

Landauer, T., Foltz, P., Laham, D., 1998. An Introduction to Latent Semantic Analysis. Discourse Processes 25, 259–284. https://doi.org/10.1080/01638539809545028

Landauer, T.K. (Ed.), 2014. Handbook of latent semantic analysis, 1. iss. in paperback. ed. Routledge, New York.

Meena, K., Raj, L., 2014. Evaluation of the descriptive type answers using hyperspace analog to language and self-organizing map, in: 2014 IEEE International Conference on Computational Intelligence and Computing Research. Presented at the 2014 IEEE International Conference on Computational Intelligence and Computing Research, pp. 1–5. https://doi.org/10.1109/ICCIC.2014.7238415

Mohamed, R., Watada, J., 2010. An evidential reasoning based LSA approach to document classification for knowledge acquisition, in: 2010 IEEE International Conference on Industrial Engineering and Engineering Management. Presented at the 2010 IEEE International Conference on Industrial Engineering and Engineering Management, pp. 1092–1096. https://doi.org/10.1109/IEEM.2010.5674188

Morgan, G., Harmon, R., 2001. Data Collection Techniques. Journal of the American Academy of Child and Adolescent Psychiatry 40, 973–6. https://doi.org/10.1097/00004583-200108000-00020

Pilato, G., Vassallo, G., 2015. TSVD as a Statistical Estimator in the Latent Semantic Analysis Paradigm. IEEE Transactions on Emerging Topics in Computing 3, 185–192. https://doi.org/10.1109/TETC.2014.2385594

Rosenbusch, H., Wanders, F., Pit, I.L., 2020. The Semantic Scale Network: An online tool to detect semantic overlap of psychological scales and prevent scale redundancies. Psychological Methods 25, 380–392. https://doi.org/10.1037/met0000244

Slomovitz, G., 2017. Latent Semantic Analysis (LSA) for syslog correlation, in: 2017 International Conference on Electronics, Communications and Computers (CONIELECOMP). Presented at the 2017 International Conference on Electronics, Communications and Computers (CONIELECOMP), pp. 1–4. https://doi.org/10.1109/CONIELECOMP.2017.7891819

Soundarajan, S., Eliassi-Rad, T., Gallagher, B., 2014. A Guide to Selecting a Network Similarity Method, in: Proceedings of the 2014 SIAM International Conference on Data Mining. Presented at the Proceedings of the 2014 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, pp. 1037–1045. https://doi.org/10.1137/1.9781611973440.118

Steiger, J.H., 1980. Tests for Comparing Elements of a Correlation Matrix 7.

Von Bertalanffy, L., 1967. General theory of systems : Application to psychology. Social Science Information 6, 125–136. https://doi.org/10.1177/053901846700600610

Wenli, C., 2016. Application Research on Latent Semantic Analysis for Information Retrieval, in: 2016 Eighth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). Presented at the 2016 Eighth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), pp. 118–121. https://doi.org/10.1109/ICMTMA.2016.37

# Chapter 7 Appendix

## 7.1 Experiment 1 Code

Document at various level for 4-artical dataset (law: non-law)

**Law and Non-Law dataset**

# Importing libraries

**import** os.path

**from** gensim **import** corpora

**from** gensim.models **import** LsiModel

**from** nltk.tokenize **import** RegexpTokenizer

**from** nltk.corpus **import** stopwords

**from** nltk.stem.porter **import** PorterStemmer

**from** gensim.models.coherencemodel **import** CoherenceModel

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**import** numpy **as** np

**import** sklearn **as** s

**import** dataframe\_image **as** dfi

**import** networkx **as** nx

**import** mantel

Implementing LSA on law-texts-all.txt (law-articles)

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

function: loading text file

Output : number of words, document list

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

i**=**i**+**1

**if** text **!=**'':

document **=** document**+**text

**if** text **==** 'NEW\_ARTICLE':

documents\_list**.**append(document)

document **=** ''

documents\_list**.**append(document)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list\_law,titles\_law,word\_count\_law **=** load\_data("","law-texts-all.tx

# data preprocessing

**def** preprocess\_data(doc\_set):

"""

Input : list of documents

function: text preprocessing - word tokenize, removing english stopwords, and potter stemming

Output : clean text

"""

*# initialize regex tokenizer*

word\_tokenizer **=** RegexpTokenizer(r'\w+')

*# create English stop words list*

eng\_stop **=** set(stopwords**.**words('english'))

*# Create p\_stemmer of class PorterStemmer*

po\_stemmer **=** PorterStemmer()

*# list for tokenized documents in loop*

texts **=** []

*# loop through document list*

**for** i **in** doc\_set:

*# clean and tokenize document string*

raw **=** i**.**lower()

tokens **=** word\_tokenizer**.**tokenize(raw)

tokens **=** [i **for** i **in** tokens **if** len(i)**>**3]

print(tokens)

*# removing english stop words from tokens*

stopped\_tokens **=** [i **for** i **in** tokens **if** **not** i **in** eng\_stop]

*# stemmed tokens*

stem\_tokens **=** [po\_stemmer**.**stem(i) **for** i **in** stopped\_tokens]

*# list of tokens*

texts**.**append(stem\_tokens)

**return** texts

clean\_text\_law**=**preprocess\_data(document\_list\_law)

**def** prepare\_corpus(doc\_clean):

"""

Input : preprossed document

function: creating term dictionary and document term matrix

Output : term dictionary and Document Term Matrix

"""

*# every unique term is assigned an index. dictionary = corpora.Dictionary(doc\_clean)*

term\_dictionary **=** corpora**.**Dictionary(doc\_clean)

*# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.*

docu\_term\_matrix **=** [term\_dictionary**.**doc2bow(doc) **for** doc **in** doc\_clean]

*# generate LDA model*

**return** term\_dictionary,docu\_term\_matrix

**def** create\_gensim\_lsa\_model(doc\_clean,number\_of\_topics,words):

"""

Input : preprocessed document, number of topics and word count

function: create LSA model using gensim

Output : return LSA model

"""

term\_dictionary,docu\_term\_matrix**=**prepare\_corpus(doc\_clean)

*# LSA model*

lsamodel **=** LsiModel(docu\_term\_matrix, num\_topics**=**number\_of\_topics, id2word **=** term\_dictionary) *# train model*

**return** lsamodel

*# LSA Model*

number\_of\_topics\_law **=** len(document\_list\_law)

words **=** word\_count\_law

model\_law **=** create\_gensim\_lsa\_model(clean\_text\_law,number\_of\_topics\_law,words)

words\_from\_law **=** dict(model\_law**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_law\_articles **=** create\_gensim\_lsa\_model(clean\_text\_law,number\_of\_topics\_law,words)

top\_words\_law **=** dict(model\_law**.**show\_topic(0,topn**=**5))

*#WordCloud\_API*

**from** wordcloud **import** WordCloud

text **=** dict(model\_law**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph.png')

*#Generating Cosine\_similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics\_law):

words\_from\_law **=**dict(model\_law**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_from\_law,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_law**.**keys()),

index **=** list(top\_words\_law**.**keys()))

l **=** list(top\_words\_law**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled.png')

*#Generating NetworkX graph for top 5 words from the corpus*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words\_law**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'green',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig.png')

Implementing LSA on non-law-all.txt

*#loading non-law-all.txt*

document\_list\_nonlaw,titles\_nonlaw,word\_count\_nonlaw **=** load\_data("","non-law-all”)

*#data\_processing function*

clean\_text\_nonlaw**=**preprocess\_data(document\_list\_nonlaw)

*# LSA Model*

number\_of\_topics\_nonlaw**=**len(document\_list\_nonlaw)

words**=**word\_count\_nonlaw

model\_nonlaw**=**create\_gensim\_lsa\_model(clean\_text\_nonlaw,number\_of\_topics\_nonlaw,words)

words\_from\_nonlaw **=**dict(model\_nonlaw**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_nonlaw\_articles**=**create\_gensim\_lsa\_model(clean\_text\_nonlaw,number\_of\_topics\_nonlaw,words)

top\_words\_nonlaw **=** dict(model\_nonlaw**.**show\_topic(0,topn**=**5))

*#generating wordcloud for top 10 words from lsa*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nonlaw**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph1.png')

*#Generating Cosine Similarity matrix for top 5 words from corpus(non\_law)*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics\_nonlaw):

words\_from\_nonlaw **=**dict(model\_nonlaw**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_from\_nonlaw,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_nonlaw**.**keys()),

index **=** list(top\_words\_nonlaw**.**keys()))

l **=** list(top\_words\_nonlaw**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled1.png')

*#Generating NetworkX for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(2)

l **=** list(top\_words\_nonlaw**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig1.png')

*#Defining function for mantel test*

**def** mantel\_test\_n(n):

*#All words from two corpus(mantel spaces)*

words\_law\_article **=** dict(model\_law**.**show\_topic(0, topn**=**word\_count\_law))

words\_nonlaw\_article **=** dict(model\_nonlaw**.**show\_topic(0, topn**=**word\_count\_nonlaw))

*#Top 5 words from two corpus(mantel spaces)*

top\_words\_law\_article **=** dict(model\_law**.**show\_topic(0, topn**=**n))

top\_words\_nonlaw\_article **=** dict(model\_nonlaw**.**show\_topic(0,topn**=**n))

*#Finding out the weights of top 5 law words in non law dataset and updating top words in nonlaw dataset*

c**=**{}

**for** key **in** top\_words\_law\_article:

**try**:

c[key] **=** words\_nonlaw\_article[key]

**except** KeyError:

c[key] **=** 0

top\_words\_nonlaw\_article**.**update(c)

*#Finding out the weights of top 5 non\_law words in law dataset and updating top words in law dataset*

b**=**{}

**for** key **in** top\_words\_nonlaw\_article:

**try**:

b[key] **=** words\_law\_article[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_law\_article)

top\_words\_law\_article**=**b

*#Generating pairwise distance matrix for law top words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics\_law):

words\_from\_law **=**dict(model\_law**.**show\_topic(i, topn**=**word\_count\_law))

b**=**pd**.**DataFrame(words\_from\_law,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_law\_article**.**keys()),

index **=** list(top\_words\_law\_article**.**keys()))

l **=** list(top\_words\_law\_article**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**empty((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#Generating pairwise distance matrix for nonlaw top words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics\_nonlaw):

words\_from\_nonlaw **=**dict(model\_nonlaw**.**show\_topic(i, topn**=**word\_count\_nonlaw))

b**=**pd**.**DataFrame(words\_from\_nonlaw,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_nonlaw\_article**.**keys()),

index **=** list(top\_words\_nonlaw\_article**.**keys()))

l **=** list(top\_words\_nonlaw\_article**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**empty((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_law\_article), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_nonlaw\_article), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**10000, method**=**'pearson', tail**=**'upper’)

*#calling mantel test for 10 word network(top 5 words from each dataset)*

mantel\_test\_n(5)

*#calling mantel test for 20 word network(top 10 words from each dataset)*

mantel\_test\_n(10)

2. LSA at 2-3 paragraph Level (Considering each document with 200 words)

**law-texts-all.txt**

*#loading the dataset*

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

function: loading text file

Output : number of words, document list

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** word **in** words:

i**=**i**+**1

document **=** document**+**word**+**' '

**if** i **==** 200:

documents\_list**.**append(document)

document,i **=** '',0

**if** len(documents\_list) **==** 2:

**break**

print("Total Number of Documents:",len(documents\_list))

*#titles.append( text[0:min(len(text),100)] )*

**return** documents\_list,titles,len(words)

document\_list\_law,titles\_law,word\_count\_law **=** load\_data("","law-texts-all.txt”)

*#data preprocessing or cleaning*

clean\_text\_law**=**preprocess\_data(document\_list\_law)

*# LSA Model*

number\_of\_topics**=**len(document\_list\_law)

words**=**word\_count\_law

model\_law**=**create\_gensim\_lsa\_model(clean\_text\_law,number\_of\_topics,words)

words\_from\_law **=**dict(model\_law**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical space comparission mantel test*

model\_law\_p**=**create\_gensim\_lsa\_model(clean\_text\_law,number\_of\_topics,words)

*#top words from law dataset*

top\_words\_law **=** dict(model\_law**.**show\_topic(0, topn**=**5))

*#generating word cloud*

**from** wordcloud **import** WordCloud

text **=** dict(model\_law**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph2.png')

*#Genarating cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_law **=**dict(model\_law**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_from\_law,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_law**.**keys()),

index **=** list(top\_words\_law**.**keys()))

l **=** list(top\_words\_law**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled2.png')

*#generating networkx graph*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words\_law**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'green',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig2.png')

non-law-all.txt

*#loading non law data set*

document\_list\_nonlaw,titles\_non\_law,word\_count\_nonlaw**=**load\_data("","non-law-all”)

*#data preprocessing*

clean\_text\_nonlaw**=**preprocess\_data(document\_list\_nonlaw)

*# LSA Model*

number\_of\_topics**=**len(document\_list\_nonlaw)

words**=**word\_count\_nonlaw

model\_nonlaw**=**create\_gensim\_lsa\_model(clean\_text\_nonlaw,number\_of\_topics,words)

words\_from\_law **=**dict(model\_nonlaw**.**show\_topic(0, topn**=**words))*#dict and encod*ing

*#These variables are for further experimentation in vertical comparission mantel test*

model\_nonlaw\_p**=**create\_gensim\_lsa\_model(clean\_text\_nonlaw,number\_of\_topics\_no)

*#top words from non law*

top\_words\_nonlaw **=** dict(model\_nonlaw**.**show\_topic(0, topn**=**5))

*#word cloud API*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nonlaw**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph3.png')

*#Generating cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_non\_law **=**dict(model\_nonlaw**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_from\_non\_law,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_nonlaw**.**keys()),

index **=** list(top\_words\_nonlaw**.**keys()))

l **=** list(top\_words\_nonlaw**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled3.png')

*#Generating networkx graph*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(2)

l **=** list(top\_words\_nonlaw**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig3.png')

*#mantel test with 10 word network(considering top 5 words from each space/corpus)*

mantel\_test\_n(5)

*#mantel test with 20 word network(considering top 10 words from each space/corpus)*

mantel\_test\_n(10)

**LSA at Paragraph level**

**Each paragraph is considered as each document in LSA**

**law-text-all.txt**

*#Loading dataset*

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

function: loading text file

Output : number of words, document list

"""

documents\_list **=** []

titles**=**[]

**with** open( os**.**path**.**join(path, file\_name) ,"r") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

**if** text **!=**'':

documents\_list**.**append(text)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list\_law,titles\_law,word\_count\_law**=**load\_data("","law-texts-all.txt")

*#data preprocessing/cleaning*

clean\_text\_law**=**preprocess\_data(document\_list\_law)

*# LSA Model*

number\_of\_topics**=**10

words**=**word\_count\_law

model\_law**=**create\_gensim\_lsa\_model(clean\_text\_law,number\_of\_topics,words)

words\_from\_law **=**dict(model\_law**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_law\_paragraph**=**create\_gensim\_lsa\_model(clean\_text\_law,number\_of\_topics,words)

*#top words from law*

top\_words\_law **=** dict(model\_law**.**show\_topic(0, topn**=**5))

*#word cloud API*

**from** wordcloud **import** WordCloud

text **=** dict(model\_law**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph4.png')

*#generating cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_law **=**dict(model\_law**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_from\_law,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_law**.**keys()),

index **=** list(top\_words\_law**.**keys()))

l **=** list(top\_words\_law**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled4.png')

*#generating NetworkX graph*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words\_law**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'green',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig4.png')

LSA for non-law-all.txt at paragraph level

*#Loading dataset*

document\_list\_nonlaw,titles\_nonlaw,word\_count\_nonlaw**=**load\_data("","non-law-al”)

*#Data pre-processing*

clean\_text\_nonlaw**=**preprocess\_data(document\_list\_nonlaw)

*# LSA Model*

number\_of\_topics**=**10

words**=**word\_count\_nonlaw

model\_nonlaw**=**create\_gensim\_lsa\_model(clean\_text\_nonlaw,number\_of\_topics,words)

words\_from\_non\_law **=**dict(model\_nonlaw**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_nonlaw\_paragraph**=**create\_gensim\_lsa\_model(clean\_text\_nonlaw,number\_of\_topics\_nonlaw,words)

In [60]:

*#top 5 words from corpus*

top\_words\_nonlaw**=**dict(model\_nonlaw**.**show\_topic(0, topn**=**5))

In [61]:

*#word cloud API*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nonlaw**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph5.png')

*#generating networkx graph*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_non\_law **=**dict(model\_nonlaw**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_from\_non\_law,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_nonlaw**.**keys()),

index **=** list(top\_words\_nonlaw**.**keys()))

l **=** list(top\_words\_nonlaw**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled5.png')

*#generating networkx graph*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(2)

l **=** list(top\_words\_nonlaw**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig5.png')

*#mantel test for 10 word network*

mantel\_test\_n(5)

*#mantel test for 20 word network*

mantel\_test\_n(10)

Vertical mantel test

Considering two spaces in matel tests as

Article level -----> 2-3 Paragraph level

Article---->Paragraph level,

Paragraph----> 2-3 Paragraph level.

*#function definition for vertical mantel test*

**def** mantel\_test\_verticle(model\_a,model\_b,n):

*# corpus of word from dataset*

words\_law\_article **=** dict(model\_a**.**show\_topic(0, topn**=**word\_count\_law))

words\_nonlaw\_article **=** dict(model\_b**.**show\_topic(0, topn**=**word\_count\_nonlaw))

*# Top words from corpus of dataset*

top\_words\_law\_article **=** dict(model\_a**.**show\_topic(0, topn**=**n))

top\_words\_nonlaw\_article **=** dict(model\_b**.**show\_topic(0,topn**=**n))

c**=**{} *#law words in non law*

**for** key **in** top\_words\_law\_article:

**try**:

c[key] **=** words\_nonlaw\_article[key]

**except** KeyError:

c[key] **=** 0

top\_words\_nonlaw\_article**.**update(c)

b**=**{} *#non law words in law*

**for** key **in** top\_words\_nonlaw\_article:

**try**:

b[key] **=** words\_law\_article[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_law\_article)

top\_words\_law\_article**=**b

*#pairwise distance for law*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics\_law):

words\_from\_law **=**dict(model\_a**.**show\_topic(i, topn**=**word\_count\_law))

b**=**pd**.**DataFrame(words\_from\_law,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_law\_article**.**keys()),

index **=** list(top\_words\_law\_article**.**keys()))

l **=** list(top\_words\_law\_article**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**empty((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distace for non-law*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics\_nonlaw):

words\_from\_nonlaw **=**dict(model\_b**.**show\_topic(i, topn**=**word\_count\_nonlaw))

b**=**pd**.**DataFrame(words\_from\_nonlaw,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_nonlaw\_article**.**keys()),

index **=** list(top\_words\_nonlaw\_article**.**keys()))

l **=** list(top\_words\_nonlaw\_article**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**empty((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_law\_article), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_nonlaw\_article), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**10000, method**=**'pearson', tail**=**'upper')

#### Now permutating the mantel test vertical with different parameters

#### mantel\_test\_vertical(X,Y,Z)

##### Where X is the pairwise distance for top words in space 1

##### Y is the pairwise distance for top words in space 2

##### Z is top words count from the corpus (ike 5 for 10 word network because 5 top words from each network)

mantel\_test\_verticle(model\_law\_articles,model\_law\_p,5)

mantel\_test\_verticle(model\_law\_articles,model\_law\_paragraph,5)

mantel\_test\_verticle(model\_law\_p,model\_law\_paragraph,5)

mantel\_test\_verticle(model\_nonlaw\_articles,model\_nonlaw\_p,5)

mantel\_test\_verticle(model\_nonlaw\_articles,model\_nonlaw\_paragraph,5)

mantel\_test\_verticle(model\_nonlaw\_p,model\_nonlaw\_paragraph,5)

mantel\_test\_verticle(model\_law\_articles,model\_law\_p,10)

mantel\_test\_verticle(model\_law\_articles,model\_law\_paragraph,10)

mantel\_test\_verticle(model\_law\_p,model\_law\_paragraph,10)

mantel\_test\_verticle(model\_nonlaw\_articles,model\_nonlaw\_p,10)

mantel\_test\_verticle(model\_nonlaw\_articles,model\_nonlaw\_paragraph,10)

mantel\_test\_verticle(model\_nonlaw\_p,model\_nonlaw\_paragraph,10)

## 7.2 Experiment 2 Code

Document at various level for small dataset(COVID-science:non-science)

1. Article level

In the file each article is seperated by 'NEW\_PAPER' in 'covid-scienc.txt' and 'NEW NEWSPAPER' in 'covid-non-science'.

*#importing libraries*

**import** os.path

**from** gensim **import** corpora

**from** gensim.models **import** LsiModel

**from** nltk.tokenize **import** RegexpTokenizer

**from** nltk.corpus **import** stopwords

**from** nltk.stem.porter **import** PorterStemmer

**from** gensim.models.coherencemodel **import** CoherenceModel

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**import** numpy **as** np

**import** sklearn **as** s

**import** dataframe\_image **as** dfi

**import** networkx **as** nx

**import** mantel

LSA on covid-science-small.txt at article level

**def** load\_data(path,file\_name,seperator):

"""

Input : path and file\_name

function: loading text file

Output : number of words, document list

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r", encoding**=**"utf8") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

i**=**i**+**1

**if** text **!=**'':

document **=** document**+**text

**if** text **==** seperator:

documents\_list**.**append(document)

document **=** ''

documents\_list**.**append(document)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science-small.txt",'NEW\_PAPER')

**def** preprocess\_data(doc\_set):

"""

Input : list of documents

function: text preprocessing - word tokenize, removing english stopwords, and potter stemming

Output : clean text

"""

*# initialize regex tokenizer*

word\_tokenizer **=** RegexpTokenizer(r'\w+')

*# create English stop words list*

eng\_stop **=** set(stopwords**.**words('english'))

*# Create p\_stemmer of class PorterStemmer*

po\_stemmer **=** PorterStemmer()

*# list for tokenized documents in loop*

texts **=** []

*# loop through document list*

**for** i **in** doc\_set:

*# clean and tokenize document string*

raw **=** i**.**lower()

tokens **=** word\_tokenizer**.**tokenize(raw)

tokens **=** [i **for** i **in** tokens **if** len(i)**>**3]

print(tokens)

*# removing english stop words from tokens*

stopped\_tokens **=** [i **for** i **in** tokens **if** **not** i **in** eng\_stop]

*# stemmed tokens*

stem\_tokens **=** [po\_stemmer**.**stem(i) **for** i **in** stopped\_tokens]

*# list of tokens*

texts**.**append(stem\_tokens)

**return** texts

clean\_text**=**preprocess\_data(document\_list)

*# preparing corpus for LSA*

**def** prepare\_corpus(doc\_clean):

"""

Input : preprossed document

function: creating term dictionary and document term matrix

Output : term dictionary and Document Term Matrix

"""

*# every unique term is assigned an index. dictionary = corpora.Dictionary(doc\_clean)*

term\_dictionary **=** corpora**.**Dictionary(doc\_clean)

*# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.*

docu\_term\_matrix **=** [term\_dictionary**.**doc2bow(doc) **for** doc **in** doc\_clean]

*# generate LDA model*

**return** term\_dictionary,docu\_term\_matrix

**def** create\_gensim\_lsa\_model(doc\_clean,number\_of\_topics,words):

"""

Input : preprocessed document, number of topics and word count

function: create LSA model using gensim

Output : return LSA model

"""

term\_dictionary,docu\_term\_matrix**=**prepare\_corpus(doc\_clean)

*# LSA model*

lsamodel **=** LsiModel(docu\_term\_matrix, num\_topics**=**number\_of\_topics, id2word **=** term\_dictionary) *# train model*

**return** lsamodel

*# LSA Model*

number\_of\_topics**=**10

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_covid\_science **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#dictionary of top 5 words*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#wordcloud API*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph6.png')

*# Generating Cosine\_similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled6.png')

*# Generating NetworkX graph for top 5 words from the corpus*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'g',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig6.png')

LSA on covid-non-science-small.txt at article level

*#loading covid non science dataset*

document\_list,titles,word\_count\_nscience**=**load\_data("","covid-non-science-small.txt",'NEW NEWSPAPER')

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**10

words**=**word\_count\_nscience

model\_nscience**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_non\_science **=**dict(model\_nscience**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*# generated top 5 words using lSA*

dict(model\_nscience**.**show\_topic(0,topn**=**5))

*#wordcloud API*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nscience**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph7.png')

*# Cosine similarity matrix for top 5 words from corpus*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_non\_science **=**dict(model\_nscience**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_non\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_nscience**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled7.png')

*#Generated Networkx for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"y")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig7.png')

LSA on covid-psychology-small.txt at article level

*#loading covid psychology small dataset*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology-SMALL.txt",'NEW PAPER')

*#data preprocessing/cleaning*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**10

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*#top 5 words generated from psychology dataset*

dict(model\_psy**.**show\_topic(0,topn**=**5))

*#wordcloud for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph27.png')

*# Cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psychology **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psychology,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled8.png')

*#generating networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig8.png')

*# defining mantel test function*

**def** mantel\_test(n):

*#all words from corpus*

words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**word\_count\_science))

words\_psy **=** dict(model\_psy**.**show\_topic(0, topn**=**word\_count\_psy))

*#top words from the corpus*

top\_words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**n))

top\_words\_psy **=** dict(model\_psy**.**show\_topic(0,topn**=**n))

*#Finding out the weights of top 5 law words in covid science and updating their weights w.r.t psychology dataset*

c**=**{}

**for** key **in** top\_words\_science:

**try**:

c[key] **=** words\_psy[key]

**except** KeyError:

c[key] **=** 0

top\_words\_psy**.**update(c)

*#Finding out the weights of top 5 law words in covid psychology and updating their weights w.r.t covid science dataset*

b**=**{}

**for** key **in** top\_words\_psy:

**try**:

b[key] **=** words\_science[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_science)

top\_words\_science**=**b

*#pairwise distance for top 10 words in covid science*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_science **=**dict(model\_science**.**show\_topic(i, topn**=**word\_count\_science))

b**=**pd**.**DataFrame(words\_from\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_science**.**keys()),

index **=** list(top\_words\_science**.**keys()))

l **=** list(top\_words\_science**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distace of top 10 words in psychology dataset*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_psy **=**dict(model\_psy**.**show\_topic(i, topn**=**word\_count\_psy))

b**=**pd**.**DataFrame(words\_from\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_psy**.**keys()),

index **=** list(top\_words\_psy**.**keys()))

l **=** list(top\_words\_psy**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_science), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_psy), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**100000, method**=**'pearson', tail**=**'upper')

*#mantel test with 10 word network*

mantel\_test(5)

*#mantel test with 20 word network*

mantel\_test(10)

*# mantel test for covid science and non science dataset(Not necessary for experimental results of project)*

**def** mantel\_test\_n(n):

words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**word\_count\_science))

words\_nscience **=** dict(model\_nscience**.**show\_topic(0, topn**=**word\_count\_nscience))

top\_words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**n))

top\_words\_nscience **=** dict(model\_nscience**.**show\_topic(0,topn**=**n))

c**=**{}

**for** key **in** top\_words\_science:

**try**:

c[key] **=** words\_nscience[key]

**except** KeyError:

c[key] **=** 0

top\_words\_nscience**.**update(c)

b**=**{}

**for** key **in** top\_words\_nscience:

**try**:

b[key] **=** words\_science[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_science)

top\_words\_science**=**b

*#pairwise distance*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_science **=**dict(model\_science**.**show\_topic(i, topn**=**word\_count\_science))

b**=**pd**.**DataFrame(words\_from\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_science**.**keys()),

index **=** list(top\_words\_science**.**keys()))

l **=** list(top\_words\_science**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distace*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_nscience **=**dict(model\_nscience**.**show\_topic(i, topn**=**word\_count\_nscience))

b**=**pd**.**DataFrame(words\_from\_nscience,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_nscience**.**keys()),

index **=** list(top\_words\_nscience**.**keys()))

l **=** list(top\_words\_nscience**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_science), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_nscience), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**100000, method**=**'pearson', tail**=**'upper')

mantel\_test\_n(5)

mantel\_test\_n(10)

2. 2-3 paragraphs

LSA on covid-science-small.txt at 2-3 paragraph level

*#loading dataset*

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

Purpose: loading text file

Output : list of paragraphs/documents and

title(initial 100 words considred as title of document)

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r",encoding**=**"utf8") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** word **in** words:

i**=**i**+**1

document **=** document**+**word**+**' '

**if** i **==** 200:

documents\_list**.**append(document)

document,i **=** '',0

**if** len(documents\_list) **==** 2:

**break**

print("Total Number of Documents:",len(documents\_list))

*#titles.append( text[0:min(len(text),100)] )*

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science-small.txt")

*#data preprocessing/cleaning*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_science **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#top words from the corpus*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#wordcloud for top 5 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph8.png')

*# Cosine similarity for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled9.png')

*# networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'g',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig9.png')

LSA on covid-non-science-small.txt at 2-3 paragraph level

*#loading data*

document\_list,titles,word\_count\_nscience**=**load\_data("","covid-non-science-small.txt")

*# data preprocessing/clenaing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words**=**word\_count\_nscience

model\_nscience**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_non\_science **=**dict(model\_nscience**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#dictionary of top 5 words*

dict(model\_nscience**.**show\_topic(0, topn**=**5))

*#wordcloud of top 5 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nscience**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph9.png')

*# Cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_non\_science **=**dict(model\_nscience**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_non\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_nscience**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled10.png')

*#networkx graph of top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig10.png')

LSA on covid-psychology-small.txt at 2-3 paragraph level

*#loading data*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology-small.txt")

*#data preprocessing/cleaning*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words)) *#dict and encoding matrix values*

*#top 5 words from documents*

dict(model\_psy**.**show\_topic(0, topn**=**5))

*#wordcloud of top 5 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph28.png')

*# Cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psychology **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psychology,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled11.png')

*#NetworkX graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig11.png')

*#mantel test for 10 word network*

mantel\_test(5)

*#mantel test for 20 word network*

mantel\_test(10)

mantel\_test\_n(5)

mantel\_test\_n(10)

Paragraph level

LSA oncovid-science-small.txt at paragraph level

*#data loading*

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

Purpose: loading text file

Output : list of paragraphs/documents and

title(initial 100 words considred as title of document)

"""

documents\_list **=** []

titles**=**[]

**with** open( os**.**path**.**join(path, file\_name) ,"r",encoding**=**'UTF8') **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

**if** text **!=**'':

documents\_list**.**append(text)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science-small.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_law **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#dictionary of top 5 words*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#wordcloud for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph10.png')

*# cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled12.png')

*#Networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'g',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig12.png')

LSA on covid-non-science-small.txt at paragraph level

*#loading dataset*

document\_list,titles,word\_count\_nscience**=**load\_data("","covid-non-science-small.txt")

*#cleaning documents*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_nscience

model\_nscience**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_non\_science **=**dict(model\_nscience**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#dictionary of top 5 words*

dict(model\_nscience**.**show\_topic(0, topn**=**5))

*#wordcloud for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nscience**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph11.png')

*# Cosinesimilarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_non\_science **=**dict(model\_nscience**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_non\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_nscience**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled13.png')

*#networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig13.png')

LSA on covid-psychology-small.txt at paragraph level

*#loading dataset*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology-small.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#dictionary of top 5 words*

dict(model\_psy**.**show\_topic(0, topn**=**5))

*#wordcloud for top 5 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph29.png')

*# cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psychology **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psychology,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled14.png')

*#networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig14.png')

mantel\_test(5)

mantel\_test(10)

mantel\_test\_n(5)

mantel\_test\_n(10)

## 7.3 Experiment 3 Code

Document at various level for datasets(COVID-science : non-science : Psychology)

1. Article level

*#importing libraries*

**import** os.path

**from** gensim **import** corpora

**from** gensim.models **import** LsiModel

**from** nltk.tokenize **import** RegexpTokenizer

**from** nltk.corpus **import** stopwords

**from** nltk.stem.porter **import** PorterStemmer

**from** gensim.models.coherencemodel **import** CoherenceModel

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**import** numpy **as** np

**import** sklearn **as** s

**import** dataframe\_image **as** dfi

**import** networkx **as** nx

**import** mantel

LSA on covid-science.txt at article level

**def** load\_data(path,file\_name,seperator):

"""

Input : path and file\_name

function: loading text file

Output : number of words, document list

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r", encoding**=**"utf8") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

i**=**i**+**1

**if** text **!=**'':

document **=** document**+**text

**if** text **==** seperator:

documents\_list**.**append(document)

document **=** ''

documents\_list**.**append(document)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science.txt",'NEW\_PAPER')

*#data preprocessing/cleaning*

**def** preprocess\_data(doc\_set):

"""

Input : list of documents

function: text preprocessing - word tokenize, removing english stopwords, and potter stemming

Output : clean text

"""

*# initialize regex tokenizer*

word\_tokenizer **=** RegexpTokenizer(r'\w+')

*# create English stop words list*

eng\_stop **=** set(stopwords**.**words('english'))

*# Create p\_stemmer of class PorterStemmer*

po\_stemmer **=** PorterStemmer()

*# list for tokenized documents in loop*

texts **=** []

*# loop through document list*

**for** i **in** doc\_set:

*# clean and tokenize document string*

raw **=** i**.**lower()

tokens **=** word\_tokenizer**.**tokenize(raw)

tokens **=** [i **for** i **in** tokens **if** len(i)**>**3]

print(tokens)

*# removing english stop words from tokens*

stopped\_tokens **=** [i **for** i **in** tokens **if** **not** i **in** eng\_stop]

*# stemmed tokens*

stem\_tokens **=** [po\_stemmer**.**stem(i) **for** i **in** stopped\_tokens]

*# list of tokens*

texts**.**append(stem\_tokens)

**return** texts

clean\_text**=**preprocess\_data(document\_list)

*# preparing corpus for LSA*

**def** prepare\_corpus(doc\_clean):

"""

Input : preprossed document

function: creating term dictionary and document term matrix

Output : term dictionary and Document Term Matrix

"""

*# every unique term is assigned an index. dictionary = corpora.Dictionary(doc\_clean)*

term\_dictionary **=** corpora**.**Dictionary(doc\_clean)

*# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.*

docu\_term\_matrix **=** [term\_dictionary**.**doc2bow(doc) **for** doc **in** doc\_clean]

*# generate LDA model*

**return** term\_dictionary,docu\_term\_matrix

**def** create\_gensim\_lsa\_model(doc\_clean,number\_of\_topics,words):

"""

Input : preprocessed document, number of topics and word count

function: create LSA model using gensim

Output : return LSA model

"""

term\_dictionary,docu\_term\_matrix**=**prepare\_corpus(doc\_clean)

*# LSA model*

lsamodel **=** LsiModel(docu\_term\_matrix, num\_topics**=**number\_of\_topics, id2word **=** term\_dictionary) *# train model*

**return** lsamodel

*# LSA Model*

number\_of\_topics**=**20

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_covid\_science **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_science\_articles **=** create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#top 5 words from lsa*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#wordcloud for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph12.png')

*# cosinesimilarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled15.png')

*#networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'g',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig15.png')

LSA on covid-non-science.txt at article level

*#loading document*

document\_list,titles,word\_count\_nscience**=**load\_data("","covid-non-science.txt",'NEW NEWSPAPER')

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**20

words**=**word\_count\_nscience

model\_nscience**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_non\_science **=**dict(model\_nscience**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*#dictionary of top 5 words*

dict(model\_nscience**.**show\_topic(0,topn**=**5))

*#wordcloud for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nscience**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph7.png')

*# cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_non\_science **=**dict(model\_nscience**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_non\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_nscience**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled7.png')

networkx graph **for** top 5 words

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"y")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig7.png')

lsa on covid-psychology.txt at article level

*#data loading*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology.txt",'NEW PAPER')

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**20

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_psy\_articles**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#top 5 words from lsa*

dict(model\_psy**.**show\_topic(0,topn**=**5))

*#wordcloud for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph27.png')

*# cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psychology **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psychology,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled8.png')

*#networkx for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig8.png')

*# defining mantel test function*

**def** mantel\_test(n):

*#all words from corpus*

words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**word\_count\_science))

words\_psy **=** dict(model\_psy**.**show\_topic(0, topn**=**word\_count\_psy))

*#top words from the corpus*

top\_words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**n))

top\_words\_psy **=** dict(model\_psy**.**show\_topic(0,topn**=**n))

*#Finding out the weights of top 5 law words in covid science and updating their weights w.r.t psychology dataset*

c**=**{}

**for** key **in** top\_words\_science:

**try**:

c[key] **=** words\_psy[key]

**except** KeyError:

c[key] **=** 0

top\_words\_psy**.**update(c)

*#Finding out the weights of top 5 law words in covid psychology and updating their weights w.r.t covid science dataset*

b**=**{}

**for** key **in** top\_words\_psy:

**try**:

b[key] **=** words\_science[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_science)

top\_words\_science**=**b

*#pairwise distance for top 10 words in covid science*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_science **=**dict(model\_science**.**show\_topic(i, topn**=**word\_count\_science))

b**=**pd**.**DataFrame(words\_from\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_science**.**keys()),

index **=** list(top\_words\_science**.**keys()))

l **=** list(top\_words\_science**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distace of top 10 words in psychology dataset*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_psy **=**dict(model\_psy**.**show\_topic(i, topn**=**word\_count\_psy))

b**=**pd**.**DataFrame(words\_from\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_psy**.**keys()),

index **=** list(top\_words\_psy**.**keys()))

l **=** list(top\_words\_psy**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_science), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_psy), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**100000, method**=**'pearson', tail**=**'upper')

mantel\_test(5)

*#mantel test with 20 word network*

mantel\_test(10)

*# mantel test for covid science and non science dataset(Not necessary for experimental results of project)*

**def** mantel\_test\_n(n):

words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**word\_count\_science))

words\_nscience **=** dict(model\_nscience**.**show\_topic(0, topn**=**word\_count\_nscience))

top\_words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**n))

top\_words\_nscience **=** dict(model\_nscience**.**show\_topic(0,topn**=**n))

c**=**{} *#law words in non law*

**for** key **in** top\_words\_science:

**try**:

c[key] **=** words\_nscience[key]

**except** KeyError:

c[key] **=** 0

top\_words\_nscience**.**update(c)

b**=**{}

**for** key **in** top\_words\_nscience:

**try**:

b[key] **=** words\_science[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_science)

top\_words\_science**=**b

*#pairwise distance*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_science **=**dict(model\_science**.**show\_topic(i, topn**=**word\_count\_science))

b**=**pd**.**DataFrame(words\_from\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_science**.**keys()),

index **=** list(top\_words\_science**.**keys()))

l **=** list(top\_words\_science**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distace for non-law*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_nscience **=**dict(model\_nscience**.**show\_topic(i, topn**=**word\_count\_nscience))

b**=**pd**.**DataFrame(words\_from\_nscience,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_nscience**.**keys()),

index **=** list(top\_words\_nscience**.**keys()))

l **=** list(top\_words\_nscience**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_science), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_nscience), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**100000, method**=**'pearson', tail**=**'upper')

mantel\_test\_n(5)

mantel\_test\_n(10)

2. 2-3 paragraphs

LSA on covid-science.txt at 2-3 paragraph level

*#loading dataset*

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

Purpose: loading text file

Output : list of paragraphs/documents and

title(initial 100 words considred as title of document)

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r",encoding**=**"utf8") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** word **in** words:

i**=**i**+**1

document **=** document**+**word**+**' '

**if** i **==** 200:

documents\_list**.**append(document)

document,i **=** '',0

**if** len(documents\_list) **==** 2:

**break**

print("Total Number of Documents:",len(documents\_list))

*#titles.append( text[0:min(len(text),100)] )*

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_science **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_science\_p**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#top 5 words from LSA*

dict(model\_science**.**show\_topic(0, topn**=**5))

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph8.png')

*# Cosine similarity matrix*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled9.png')

*#networkx*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'g',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig9.png')

LSA on covid-non-science.txt at 2-3 paragraph level

*#data loading*

document\_list,titles,word\_count\_nscience**=**load\_data("","covid-non-science.txt")

*#data cleaning*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words**=**word\_count\_nscience

model\_nscience**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_non\_science **=**dict(model\_nscience**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#top 5 words*

dict(model\_nscience**.**show\_topic(0, topn**=**5))

*#wordcloud of top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nscience**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph9.png')

*# cosine similarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_covid\_non\_science **=**dict(model\_nscience**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_covid\_non\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_nscience**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled10.png')

*#networkx*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig10.png')

LSA on covid-psychology.txt at 2-3 paragraph level

*#data losding*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_psy\_p **=** create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#dictionary of top 5 words*

dict(model\_psy**.**show\_topic(0, topn**=**5))

*#wordcloud*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph28.png')

*# Cosinesimilarity matrix for top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psychology **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psychology,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled11.png')

*#networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig11.png')

*#mantel test for 10 word network*

mantel\_test(5)

*#mantel test for 20 word network*

mantel\_test(10)

mantel\_test\_n(5)

mantel\_test\_n(10)

3. Paragraph level

LSA on covid-science.txt at paragraph level

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

Purpose: loading text file

Output : list of paragraphs/documents and

title(initial 100 words considred as title of document)

"""

documents\_list **=** []

titles**=**[]

**with** open( os**.**path**.**join(path, file\_name) ,"r",encoding**=**'UTF8') **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

**if** text **!=**'':

documents\_list**.**append(text)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science.txt")

*#data cleaning*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_law **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_science\_paragraph**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#top 5 words from lsa*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#word cloud of top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph10.png')

*# cosine similarity of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled12.png')

*#networkx of top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***100 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'g',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig12.png')

LSA on covid-non-science-small.txt at 2-3 paragraph level

*#data loading*

document\_list,titles,word\_count\_nscience**=**load\_data("","covid-non-science.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_nscience

model\_nscience**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_non\_science **=**dict(model\_nscience**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#top 5 words from lsa*

dict(model\_nscience**.**show\_topic(0, topn**=**5))

*#word cloud for top 5 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_nscience**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph11.png')

*# cosine similarity of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_non\_science **=**dict(model\_nscience**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_non\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_nscience**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled13.png')

*#networkx for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'y',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig13.png')

LSA on covid-psychology-small.txt at paragraph level

*# data loading*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology.txt")

*#data cleaning*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_psy\_paragraph**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#top 5 words from lsa*

dict(model\_psy**.**show\_topic(0, topn**=**5))

*#word cloud for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph29.png')

*# cosine similarity of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psychology **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psychology,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled14.png')

*#networkx for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***100 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig14.png')

mantel\_test(5)

mantel\_test(10)

mantel\_test\_n(5)

mantel\_test\_n(10)

Vertical mantel test

Considering two spaces in matel tests as

Article level -----> 2-3 Paragraph level ,

Article---->Paragraph level,

Paragraph----> 2-3 Paragraph level

**def** mantel\_test\_verticle(model\_a,model\_b,n):

*#all words from the cleaned documents*

words\_science **=** dict(model\_a**.**show\_topic(0, topn**=**word\_count\_science))

words\_psy **=** dict(model\_b**.**show\_topic(0, topn**=**word\_count\_psy))

*#top 5 words from the documents*

top\_words\_science **=** dict(model\_a**.**show\_topic(0, topn**=**n))

top\_words\_psy **=** dict(model\_b**.**show\_topic(0,topn**=**n))

*#Finding out the weights of top 5 law words in model\_a level and updating their weights w.r.t model b dataset*

c**=**{}

**for** key **in** top\_words\_science:

**try**:

c[key] **=** words\_psy[key]

**except** KeyError:

c[key] **=** 0

top\_words\_psy**.**update(c)

*#Finding out the weights of top 5 law words in model b and updating their weights w.r.t model a dataset*

b**=**{}

**for** key **in** top\_words\_psy:

**try**:

b[key] **=** words\_science[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_science)

top\_words\_science**=**b

*#pairwise distance for top 10 words in model a*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_science **=**dict(model\_a**.**show\_topic(i, topn**=**word\_count\_science))

b**=**pd**.**DataFrame(words\_from\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_science**.**keys()),

index **=** list(top\_words\_science**.**keys()))

l **=** list(top\_words\_science**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distance for top 10 words in model b*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_psy **=**dict(model\_b**.**show\_topic(i, topn**=**word\_count\_psy))

b**=**pd**.**DataFrame(words\_from\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_psy**.**keys()),

index **=** list(top\_words\_psy**.**keys()))

l **=** list(top\_words\_psy**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_science), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_psy), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**10000, method**=**'pearson', tail**=**'upper')

mantel\_test\_verticle(model\_science\_articles,model\_science\_p,5)

mantel\_test\_verticle(model\_science\_articles,model\_science\_paragraph,5)

mantel\_test\_verticle(model\_science\_p,model\_science\_paragraph,5)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_p,5)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_paragraph,5)

mantel\_test\_verticle(model\_psy\_p,model\_psy\_paragraph,5)

mantel\_test\_verticle(model\_science\_articles,model\_science\_p,10)

mantel\_test\_verticle(model\_science\_articles,model\_science\_paragraph,10)

mantel\_test\_verticle(model\_science\_p,model\_science\_paragraph,10)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_p,10)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_paragraph,10)

mantel\_test\_verticle(model\_psy\_p,model\_psy\_paragraph,10)

## 7.4 Experiment 4 Code

Document at various levels for covid science large and covid psychology large datasets

1. Article level

*#importing libraries*

**import** os.path

**from** gensim **import** corpora

**from** gensim.models **import** LsiModel

**from** nltk.tokenize **import** RegexpTokenizer

**from** nltk.corpus **import** stopwords

**from** nltk.stem.porter **import** PorterStemmer

**from** gensim.models.coherencemodel **import** CoherenceModel

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**import** numpy **as** np

**import** sklearn **as** s

**import** dataframe\_image **as** dfi

**import** networkx **as** nx

**import** mantel

LSA on covid-science.txt at article level

**def** load\_data(path,file\_name,seperator):

""""

Input : path and file\_name

function: loading text file

Output : number of words, document list

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r", encoding**=**"utf8") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

i**=**i**+**1

**if** text **!=**'':

document **=** document**+**text

**if** text **==** seperator:

documents\_list**.**append(document)

document **=** ''

documents\_list**.**append(document)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science-large.txt",'NEW\_PAPER')

**def** preprocess\_data(doc\_set):

"""

Input : list of documents

function: text preprocessing - word tokenize, removing english stopwords, and potter stemming

Output : clean text

"""

*# initialize regex tokenizer*

word\_tokenizer **=** RegexpTokenizer(r'\w+')

*# create English stop words list*

eng\_stop **=** set(stopwords**.**words('english'))

*# Create p\_stemmer of class PorterStemmer*

po\_stemmer **=** PorterStemmer()

*# list for tokenized documents in loop*

texts **=** []

*# loop through document list*

**for** i **in** doc\_set:

*# clean and tokenize document string*

raw **=** i**.**lower()

tokens **=** word\_tokenizer**.**tokenize(raw)

tokens **=** [i **for** i **in** tokens **if** len(i)**>**3]

print(tokens)

*# removing english stop words from tokens*

stopped\_tokens **=** [i **for** i **in** tokens **if** **not** i **in** eng\_stop]

*# stemmed tokens*

stem\_tokens **=** [po\_stemmer**.**stem(i) **for** i **in** stopped\_tokens]

*# list of tokens*

texts**.**append(stem\_tokens)

**return** texts

clean\_text**=**preprocess\_data(document\_list)

*# preparing corpus for LSA*

**def** prepare\_corpus(doc\_clean):

"""

Input : preprossed document

function: creating term dictionary and document term matrix

Output : term dictionary and Document Term Matrix

"""

*# every unique term is assigned an index. dictionary = corpora.Dictionary(doc\_clean)*

term\_dictionary **=** corpora**.**Dictionary(doc\_clean)

*# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.*

docu\_term\_matrix **=** [term\_dictionary**.**doc2bow(doc) **for** doc **in** doc\_clean]

*# generate LDA model*

**return** term\_dictionary,docu\_term\_matrix

**def** create\_gensim\_lsa\_model(doc\_clean,number\_of\_topics,words):

"""

Input : preprocessed document, number of topics and word count

function: create LSA model using gensim

Output : return LSA model

"""

term\_dictionary,docu\_term\_matrix**=**prepare\_corpus(doc\_clean)

*# LSA model*

lsamodel **=** LsiModel(docu\_term\_matrix, num\_topics**=**number\_of\_topics, id2word **=** term\_dictionary) *# train model*

**return** lsamodel

*# LSA Model*

number\_of\_topics**=**25

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_science **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_science\_articles **=** create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#top 5 words from model lsa*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#wordcloud of top 10 words from the documents*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph21.png')

*# cosine similarity of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled24.png')

*#networkx for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'green',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig24.png')

LSA on covid-psychology-large.txt at article level

*#loading dataset*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology-large.txt",'NEW PAPER')

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**25

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*#dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_psy\_articles**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#dictionary of top 5 words*

dict(model\_psy**.**show\_topic(0,topn**=**5))

*#wordcloud of top 10 words from lsa*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph22.png')

*#cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psy **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled25.png')

*#networkx garph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig25.png')

*# defining mantel test function*

**def** mantel\_test(n):

*#all words from corpus*

words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**word\_count\_science))

words\_psy **=** dict(model\_psy**.**show\_topic(0, topn**=**word\_count\_psy))

*#top words from the corpus*

top\_words\_science **=** dict(model\_science**.**show\_topic(0, topn**=**n))

top\_words\_psy **=** dict(model\_psy**.**show\_topic(0,topn**=**n))

*#Finding out the weights of top 5 law words in covid science and updating their weights w.r.t psychology dataset*

c**=**{}

**for** key **in** top\_words\_science:

**try**:

c[key] **=** words\_psy[key]

**except** KeyError:

c[key] **=** 0

top\_words\_psy**.**update(c)

*#Finding out the weights of top 5 law words in covid psychology and updating their weights w.r.t covid science dataset*

b**=**{}

**for** key **in** top\_words\_psy:

**try**:

b[key] **=** words\_science[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_science)

top\_words\_science**=**b

*#pairwise distance for top 10 words in covid science*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_science **=**dict(model\_science**.**show\_topic(i, topn**=**word\_count\_science))

b**=**pd**.**DataFrame(words\_from\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_science**.**keys()),

index **=** list(top\_words\_science**.**keys()))

l **=** list(top\_words\_science**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distace of top 10 words in psychology dataset*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_psy **=**dict(model\_psy**.**show\_topic(i, topn**=**word\_count\_psy))

b**=**pd**.**DataFrame(words\_from\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_psy**.**keys()),

index **=** list(top\_words\_psy**.**keys()))

l **=** list(top\_words\_psy**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_science), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_psy), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**10000, method**=**'pearson', tail**=**'upper')

*#mantel test for 10 word network*

mantel\_test(5)

*#mantel test for 20 word network*

mantel\_test(10)

2. 2-3 paragraphs

LSA on covid-science.txt at 2-3 paragraph level

*#loading dataset*

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

Purpose: loading text file

Output : list of paragraphs/documents and

title(initial 100 words considred as title of document)

"""

documents\_list **=** []

titles**=**[]

document **=** ''

i**=**0

**with** open( os**.**path**.**join(path, file\_name) ,"r",encoding**=**"utf8") **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** word **in** words:

i**=**i**+**1

document **=** document**+**word**+**' '

**if** i **==** 200:

documents\_list**.**append(document)

document,i **=** '',0

**if** len(documents\_list) **==** 2:

**break**

print("Total Number of Documents:",len(documents\_list))

*#titles.append( text[0:min(len(text),100)] )*

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science-large.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words\_nscience**=**len(clean\_text[0])**+**len(clean\_text[1])

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words\_nscience)

words\_science **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_science\_p**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words\_nscience)

*#dictionary of top 5 word from lsa*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#word cloud of top 10 words from lsa*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph23.png')

*#cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_science **=**dict(model\_science**.**show\_topic(i, topn **=** words\_nscience))

b**=**pd**.**DataFrame(words\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled26.png')

*#networkx graph for top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'green',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig26.png')

LSA of covid-psycology.txt at 2-3 paragraph level

*#data loading*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology-large.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**2

words\_npsy**=**len(clean\_text[0])**+**len(clean\_text[1])

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words\_npsy)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_psy\_p **=** create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words\_npsy)

*#dictionary of top 5 words*

dict(model\_psy**.**show\_topic(0, topn**=**5))

*#word cloud of top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph24.png')

*# cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psy **=**dict(model\_psy**.**show\_topic(i, topn**=**words\_npsy))

b**=**pd**.**DataFrame(words\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled27.png')

*#Networkx graph of top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***10 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'r',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig27.png')

*#mantel test for 10 word network*

mantel\_test(5)

*#mantel test for 20 word network*

mantel\_test(10)

3. Paragraph level

LSA on covid-science.txt at paragraph level

*#loading dataset*

**def** load\_data(path,file\_name):

"""

Input : path and file\_name

Purpose: loading text file

Output : list of paragraphs/documents and

title(initial 100 words considred as title of document)

"""

documents\_list **=** []

titles**=**[]

**with** open( os**.**path**.**join(path, file\_name) ,"r",encoding**=**'UTF8') **as** fin:

data **=** fin**.**read()

words **=** data**.**split()

fin**.**seek(0)

print('Number of words in text file :', len(words))

**for** line **in** fin**.**readlines():

text **=** line**.**strip()

**if** text **!=**'':

documents\_list**.**append(text)

print("Total Number of Documents:",len(documents\_list))

titles**.**append( text[0:min(len(text),100)] )

**return** documents\_list,titles,len(words)

document\_list,titles,word\_count\_science**=**load\_data("","covid-science-large.txt")

*#data preprocessing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_science

model\_science**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_law **=**dict(model\_science**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_science\_paragraph**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#dictionary of top 5 words*

dict(model\_science**.**show\_topic(0, topn**=**5))

*#word cloud of top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_science**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph25.png')

*# Cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_science **=**dict(model\_science**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_science**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled28.png')

*#networkx graph of top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***100 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'green',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig28.png')

LSA on covid-psychology-large.txt at paragraph level

*#loading data*

document\_list,titles,word\_count\_psy**=**load\_data("","covid-psychology-large.txt")

*#data preprocesing*

clean\_text**=**preprocess\_data(document\_list)

*# LSA Model*

number\_of\_topics**=**100

words**=**word\_count\_psy

model\_psy**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

words\_from\_psychology **=**dict(model\_psy**.**show\_topic(0, topn**=**words))*##dict and encoding matrix values*

*#These variables are for further experimentation in vertical comparission mantel test*

model\_psy\_paragraph**=**create\_gensim\_lsa\_model(clean\_text,number\_of\_topics,words)

*#generating top 5 words form lsa*

dict(model\_psy**.**show\_topic(0, topn**=**5))

*#wordclod for top 10 words*

**from** wordcloud **import** WordCloud

text **=** dict(model\_psy**.**show\_topic(0, topn**=**20))

l**=**list(text**.**keys())

wordcloud **=** WordCloud(width**=**150, height**=**200,max\_font\_size**=**25, max\_words**=**20, background\_color**=**"white")**.**generate(" "**.**join(l))

plt**.**imshow(wordcloud, interpolation**=**'bilinear')

plt**.**axis("off")

plt**.**savefig('graph26.png')

*# Cosine similarity matrix of top 5 words*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_psy **=**dict(model\_psy**.**show\_topic(i, topn**=**words))

b**=**pd**.**DataFrame(words\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

top\_words **=** dict(model\_psy**.**show\_topic(0,topn**=**5))

df **=** pd**.**DataFrame(columns**=**list(top\_words**.**keys()),

index **=** list(top\_words**.**keys()))

l **=** list(top\_words**.**keys())

print('cosine-similarity')

**for** i **in** l:

**for** j **in** l:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

n **=** s**.**metrics**.**pairwise**.**cosine\_similarity(matrix, matrix, dense\_output**=True**)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

dfi**.**export(df, 'df\_styled29.png')

*#networkx graph of top 5 words*

df **=** df**.**apply(pd**.**to\_numeric, errors**=**'coerce')

df**=**df**.**round(4)

l **=** list(top\_words**.**keys())

df\_adj **=** pd**.**DataFrame(df**.**to\_numpy(), index**=**l, columns**=**l)

G **=** nx**.**from\_pandas\_adjacency(df)

pos **=** {l[0]: (0, 0),l[1]: (1, 0), l[2]: (0, 1), l[3]: (1, 1), l[4]: (0.5, 0.8)}

a**=**list(G**.**edges(data**=True**))

b**=**[]

**for** i **in** a:

b**.**append(i[:][2]['weight'])

w **=** [x **\***100 **for** x **in** b]

colors **=** range(4)

nx**.**draw(G,pos, alpha**=**1, width**=**w, with\_labels **=** **True**,node\_size**=**200, edge\_color**=**'g',node\_color**=**'b')

labels **=** nx**.**get\_edge\_attributes(G,'weight')

nx**.**draw\_networkx\_nodes(G, pos, nodelist**=**l, node\_color**=**"w")

nx**.**draw\_networkx\_edge\_labels(G,pos,edge\_labels**=**labels)

plt**.**rcParams["figure.figsize"] **=** (8,5)

plt**.**savefig('fig29.png')

*#mantel test for 10 word network*

mantel\_test(5)

*#mantel test for 20 word network*

mantel\_test(10)

*Vertical mantel test*

*Considering two spaces in matel tests as*

*Article level -----> 2-3 Paragraph level ,*

*Article---->Paragraph level,*

*Paragraph----> 2-3 Paragraph level*

**def** mantel\_test\_verticle(model\_a,model\_b,n):

*#all words from the cleaned documents*

words\_science **=** dict(model\_a**.**show\_topic(0, topn**=**word\_count\_science))

words\_psy **=** dict(model\_b**.**show\_topic(0, topn**=**word\_count\_psy))

*#top 5 words from the documents*

top\_words\_science **=** dict(model\_a**.**show\_topic(0, topn**=**n))

top\_words\_psy **=** dict(model\_b**.**show\_topic(0,topn**=**n))

*#Finding out the weights of top 5 law words in model\_a level and updating their weights w.r.t model b dataset*

c**=**{}

**for** key **in** top\_words\_science:

**try**:

c[key] **=** words\_psy[key]

**except** KeyError:

c[key] **=** 0

top\_words\_psy**.**update(c)

*#Finding out the weights of top 5 law words in model b and updating their weights w.r.t model a dataset*

b**=**{}

**for** key **in** top\_words\_psy:

**try**:

b[key] **=** words\_science[key]

**except** KeyError:

b[key] **=** 0

b**.**update(top\_words\_science)

top\_words\_science**=**b

*#pairwise distance for top 10 words in model a*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_science **=**dict(model\_a**.**show\_topic(i, topn**=**word\_count\_science))

b**=**pd**.**DataFrame(words\_from\_science,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_science**.**keys()),

index **=** list(top\_words\_science**.**keys()))

l **=** list(top\_words\_science**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df1 **=** df**.**to\_numpy()

*#pairwise distance for top 10 words in model b*

a**=**pd**.**DataFrame()

**for** i **in** range(number\_of\_topics):

words\_from\_psy **=**dict(model\_b**.**show\_topic(i, topn**=**word\_count\_psy))

b**=**pd**.**DataFrame(words\_from\_psy,index**=**[i])

a**=**a**.**append(b)

a**=**a**.**transpose()

df **=** pd**.**DataFrame(columns**=**list(top\_words\_psy**.**keys()),

index **=** list(top\_words\_psy**.**keys()))

l **=** list(top\_words\_psy**.**keys())

print('Pairwise-distance')

**for** i **in** l:

**for** j **in** l:

**try**:

matrix**=**np**.**array([a**.**loc[i],a**.**loc[j]])

**except** KeyError:

matrix**=**np**.**zeros((2,3))

n **=** s**.**metrics**.**pairwise**.**nan\_euclidean\_distances(matrix, matrix)

df**.**loc[[i],[j]] **=** n[0,1]

print(df)

df2 **=** df**.**to\_numpy()

dist1 **=** list(df1[np**.**triu\_indices(len(top\_words\_science), k **=** 1)])

dist2 **=** list(df2[np**.**triu\_indices(len(top\_words\_psy), k **=** 1)])

**return** mantel**.**test(dist1, dist2, perms**=**10000, method**=**'pearson', tail**=**'upper')

mantel\_test\_verticle(model\_science\_articles,model\_science\_p,5)

mantel\_test\_verticle(model\_science\_articles,model\_science\_paragraph,5)

mantel\_test\_verticle(model\_science\_p,model\_science\_paragraph,5)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_p,5)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_paragraph,5)

mantel\_test\_verticle(model\_psy\_p,model\_psy\_paragraph,5)

mantel\_test\_verticle(model\_science\_articles,model\_science\_p,10)

mantel\_test\_verticle(model\_science\_articles,model\_science\_paragraph,10)

mantel\_test\_verticle(model\_science\_p,model\_science\_paragraph,10)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_p,10)

mantel\_test\_verticle(model\_psy\_articles,model\_psy\_paragraph,10)

mantel\_test\_verticle(model\_psy\_p,model\_psy\_paragraph,10)

**Graphical Representation of Mantel test results**

**from** matplotlib.lines **import** Line2D

**import** matplotlib.pyplot **as** plt

*# p-value Mantel Test Results(10 Word Network)*

fig **=** plt**.**figure(figsize**=**(8, 6), dpi**=**90)

x\_label **=** ['Sci\_Psy\_10','Sci\_Psy\_15','Sci\_Psy\_20','Sci\_Psy\_30','Sci\_Psy\_40']

plt**.**xticks(range(len(x\_label)), x\_label)

plt**.**title('Mantel Test Results(10 Word Network)',fontsize**=**'15',color**=**'g')

plt**.**xlabel('Data set size',fontsize**=**'15',color**=**'g')

plt**.**ylabel('P-value',fontsize**=**'15',color**=**'g')

p\_article **=** [0.87864,0.9722,0.92318,0.9074,0.6703]

p\_2\_3\_p **=** [0.6625,0.6549,0.6529,0.6558,0.8881]

p\_paragraph **=** [0.8417,0.8156,0.92942,0.9493,0.9005]

zip\_object **=** zip(p\_article,p\_2\_3\_p,p\_paragraph)

plt**.**plot(p\_article,marker**=**'o')

plt**.**plot(p\_2\_3\_p,marker**=**'o')

plt**.**plot(p\_paragraph,marker**=**'o')

i**=**0

**for** a,b,c **in** zip\_object:

plt**.**text(i, a**+**0.008, "%.2f" **%a**, ha="center")

plt**.**text(i, b**+**0.02, "%.2f" **%b**, ha="center")

plt**.**text(i, c**+**0.02, "%.2f" **%c**, ha="center")

i**=**i**+**1

custom\_lines **=** [Line2D([0], [0], color**=**'b', lw**=**4),

Line2D([0], [0], color**=**'g', lw**=**4),

Line2D([0], [0], color**=**'orange', lw**=**4)]

plt**.**legend(custom\_lines, ['Article\_level', 'Paragraph\_level', '2-3 Paragraph level'])

plt**.**show()

Null hypothesis : There is no difference in the distances among objects(top words) in a matrix of covid science dataset to the objects in a matrix of covid psychology dataset Alternantive hypothesis: There is difference in the distances among objects(top words) in a matrix of covid science dataset to the objects in a matrix of covid psychology dataset

*# r-value Mantel Test Results(10 Word Network)*

fig **=** plt**.**figure(figsize**=**(8, 6), dpi**=**90)

x\_label **=** ['Sci\_Psy\_10','Sci\_Psy\_15','Sci\_Psy\_20','Sci\_Psy\_30','Sci\_Psy\_40']

plt**.**xticks(range(len(x\_label)), x\_label)

plt**.**title('Mantel Test Results(10 Word Network)',fontsize**=**'15',color**=**'g')

plt**.**xlabel('Data set size',fontsize**=**'15',color**=**'g')

plt**.**ylabel('R-value',fontsize**=**'15',color**=**'g')

p\_article **=** [**-**0.31881,**-**0.55832,**-**0.45097,**-**0.44085,**-**0.10233]

p\_2\_3\_p **=** [**-**0.12443,**-**0.12443,**-**0.12443,**-**0.12443,**-**0.16750]

p\_paragraph **=** [**-**0.32813,**-**0.36722,**-**0.47229,**-**0.42116,**-**0.29198]

zip\_object **=** zip(p\_article,p\_2\_3\_p,p\_paragraph)

plt**.**plot(p\_article,marker**=**'o')

plt**.**plot(p\_2\_3\_p,marker**=**'o')

plt**.**plot(p\_paragraph,marker**=**'o')

i**=**0

**for** a,b,c **in** zip\_object:

plt**.**text(i, a**+**0.008, "%.2f" **%a**, ha="center")

plt**.**text(i, b**+**0.02, "%.2f" **%b**, ha="center")

plt**.**text(i, c**+**0.02, "%.2f" **%c**, ha="center")

i**=**i**+**1

**from** matplotlib.lines **import** Line2D

custom\_lines **=** [Line2D([0], [0], color**=**'b', lw**=**4),

Line2D([0], [0], color**=**'g', lw**=**4),

Line2D([0], [0], color**=**'orange', lw**=**4)]

plt**.**legend(custom\_lines, ['Article\_level', 'Paragraph\_level', '2-3 Paragraph level'])

plt**.**show()

*# z-score Mantel Test Results(10 Word Network)*

fig **=** plt**.**figure(figsize**=**(8, 6), dpi**=**90)

x\_label **=** ['Sci\_Psy\_10','Sci\_Psy\_15','Sci\_Psy\_20','Sci\_Psy\_30','Sci\_Psy\_40']

plt**.**xticks(range(len(x\_label)), x\_label)

plt**.**title('Mantel Test Results(10 Word Network)',fontsize**=**'15',color**=**'g')

plt**.**xlabel('Data set size',fontsize**=**'15',color**=**'g')

plt**.**ylabel('Z-score',fontsize**=**'15',color**=**'g')

p\_article **=** [2.935799,**-**1.20322,**-**1.76688,**-**1.40863,**-**1.33729,**-**0.455363]

p\_2\_3\_p **=** [**-**0.52720,**-**0.51143,**-**0.51520,**-**0.51814,**-**1.14848]

p\_paragraph **=** [**-**1.00066,**-**1.09169,**-**1.43762,**-**1.29004,**-**1.29744]

zip\_object **=** zip(p\_article,p\_2\_3\_p,p\_paragraph)

plt**.**plot(p\_article,marker**=**'o')

plt**.**plot(p\_2\_3\_p,marker**=**'o')

plt**.**plot(p\_paragraph,marker**=**'o')

i**=**0

**for** a,b,c **in** zip\_object:

plt**.**text(i, a**+**0.008, "%.2f" **%a**, ha="center")

plt**.**text(i, b**+**0.02, "%.2f" **%b**, ha="center")

plt**.**text(i, c**+**0.02, "%.2f" **%c**, ha="center")

i**=**i**+**1

**from** matplotlib.lines **import** Line2D

custom\_lines **=** [Line2D([0], [0], color**=**'b', lw**=**4),

Line2D([0], [0], color**=**'g', lw**=**4),

Line2D([0], [0], color**=**'orange', lw**=**4)]

plt**.**legend(custom\_lines, ['Article\_level', 'Paragraph\_level', '2-3 Paragraph level'])

plt**.**show()

*# p-value Mantel Test Results(20 Word Network)*

fig **=** plt**.**figure(figsize**=**(8, 6), dpi**=**90)

x\_label **=** ['Sci\_Psy\_10','Sci\_Psy\_15','Sci\_Psy\_20','Sci\_Psy\_30','Sci\_Psy\_40']

plt**.**xticks(range(len(x\_label)), x\_label)

plt**.**title('Mantel Test Results(20 Word Network)',fontsize**=**'15',color**=**'g')

plt**.**xlabel('Data set size',fontsize**=**'15',color**=**'g')

plt**.**ylabel('P-value',fontsize**=**'15',color**=**'g')

p\_article **=** [0.96258,0.9848,0.99278,0.7165,0.3104]

p\_2\_3\_p **=** [0.88753,0.8535,0.85532,0.8834,1.0]

p\_paragraph **=** [0.61801,0.9934,0.88279,0.9549,0.5642]

zip\_object **=** zip(p\_article,p\_2\_3\_p,p\_paragraph)

plt**.**plot(p\_article,marker**=**'o')

plt**.**plot(p\_2\_3\_p,marker**=**'o')

plt**.**plot(p\_paragraph,marker**=**'o')

i**=**0

**for** a,b,c **in** zip\_object:

plt**.**text(i, a**+**0.008, "%.2f" **%a**, ha="center")

plt**.**text(i, b**+**0.02, "%.2f" **%b**, ha="center")

plt**.**text(i, c**+**0.02, "%.2f" **%c**, ha="center")

i**=**i**+**1

custom\_lines **=** [Line2D([0], [0], color**=**'b', lw**=**4),

Line2D([0], [0], color**=**'g', lw**=**4),

Line2D([0], [0], color**=**'orange', lw**=**4)]

plt**.**legend(custom\_lines, ['Article\_level', 'Paragraph\_level', '2-3 Paragraph level'])

plt**.**show()

*# r-value Mantel Test Results(20 Word Network)*

fig **=** plt**.**figure(figsize**=**(8, 6), dpi**=**90)

x\_label **=** ['Sci\_Psy\_10','Sci\_Psy\_15','Sci\_Psy\_20','Sci\_Psy\_30','Sci\_Psy\_40']

plt**.**xticks(range(len(x\_label)), x\_label)

plt**.**title('Mantel Test Results(20 Word Network)',fontsize**=**'15',color**=**'g')

plt**.**xlabel('Data set size',fontsize**=**'15',color**=**'g')

plt**.**ylabel('r-value',fontsize**=**'15',color**=**'g')

p\_article **=** [**-**0.37001,**-**0.41870,**-**0.45208,**-**0.13052,0.16347]

p\_2\_3\_p **=** [**-**0.16750,**-**0.15851,**-**0.15851,**-**0.16750,**-**0.2556]

p\_paragraph **=** [**-**0.07063,**-**0.47758,**-**0.26062,**-**0.36792,**-**0.04745]

zip\_object **=** zip(p\_article,p\_2\_3\_p,p\_paragraph)

plt**.**plot(p\_article,marker**=**'o')

plt**.**plot(p\_2\_3\_p,marker**=**'o')

plt**.**plot(p\_paragraph,marker**=**'o')

i**=**0

**for** a,b,c **in** zip\_object:

plt**.**text(i, a**+**0.008, "%.2f" **%a**, ha="center")

plt**.**text(i, b**+**0.02, "%.2f" **%b**, ha="center")

plt**.**text(i, c**+**0.02, "%.2f" **%c**, ha="center")

i**=**i**+**1

custom\_lines **=** [Line2D([0], [0], color**=**'b', lw**=**4),

Line2D([0], [0], color**=**'g', lw**=**4),

Line2D([0], [0], color**=**'orange', lw**=**4)]

plt**.**legend(custom\_lines, ['Article\_level', 'Paragraph\_level', '2-3 Paragraph level'])

plt**.**show()

*# z-score Mantel Test Results(20 Word Network)*

fig **=** plt**.**figure(figsize**=**(8, 6), dpi**=**90)

x\_label **=** ['Sci\_Psy\_10','Sci\_Psy\_15','Sci\_Psy\_20','Sci\_Psy\_30','Sci\_Psy\_40']

plt**.**xticks(range(len(x\_label)), x\_label)

plt**.**title('Mantel Test Results(20 Word Network)',fontsize**=**'15',color**=**'g')

plt**.**xlabel('Data set size',fontsize**=**'15',color**=**'g')

plt**.**ylabel('z-score',fontsize**=**'15',color**=**'g')

p\_article **=** [**-**1.69272,**-**1.91222,**-**2.32042,**-**0.58437,0.5929]

p\_2\_3\_p **=** [**-**1.14787,**-**1.04880,**-**1.05703,**-**1.15334,**-**1.1348]

p\_paragraph **=** [**-**0.32868,**-**2.17452,**-**1.19640,**-**1.65607,**-**0.17695]

zip\_object **=** zip(p\_article,p\_2\_3\_p,p\_paragraph)

plt**.**plot(p\_article,marker**=**'o')

plt**.**plot(p\_2\_3\_p,marker**=**'o')

plt**.**plot(p\_paragraph,marker**=**'o')

i**=**0

**for** a,b,c **in** zip\_object:

plt**.**text(i, a**+**0.008, "%.2f" **%a**, ha="center")

plt**.**text(i, b**+**0.02, "%.2f" **%b**, ha="center")

plt**.**text(i, c**+**0.02, "%.2f" **%c**, ha="center")

i**=**i**+**1

custom\_lines **=** [Line2D([0], [0], color**=**'b', lw**=**4),

Line2D([0], [0], color**=**'g', lw**=**4),

Line2D([0], [0], color**=**'orange', lw**=**4)]

plt**.**legend(custom\_lines, ['Article\_level', 'Paragraph\_level', '2-3 Paragraph level'])

plt**.**show()