GRAPH KERNELS AS PREPROCESSING FOR UNSUPERVISED LEARNING: TECHNICAL REPORTS AND SOCIAL MEDIA

by

LEVI C. NICKLAS

A Thesis Submitted to the Faculty of the

DEPARTMENT OF DATA SCIENCE AND BUSINESS ANALYTICS

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE in COMPUTER SCIENCE

DATA SCIENCE TRACK

In the Graduate College

Florida Polytechnic University

2021

GRAPH KERNELS AS PREPROCESSING FOR UNSUPERVISED LEARNING: TECHNICAL REPORTS AND SOCIAL MEDIA

by

Levi C. Nicklas

A Thesis Submitted to the Faculty of the

DEPARTMENT OF DATA SCIENCE AND BUSINESS ANALYTICS

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE in COMPUTER SCIENCE

DATA SCIENCE TRACK In the Graduate College

Florida Polytechnic University

2021

Approved by:

	Signature	**Date**
Dr. Reinaldo Sanchez-Arias (Committee Chair, Advisor)		
Dr. Grisselle Centeno (Committee Member)		
Dr. Harish Chintakunta (Committee Member)		
Dr. Tom Dvorske Vice Provost of Academic Affairs (Graduate Division)		

To my wife Julienne, whose love, support, and encouragement kept me on the right track. I look forward to this life we are making.

Ecclesiastes 9:9

And to my dear baby girl, I hope this work is something you can be proud of, and that this be a reminder that hard work and perseverance are always rewarded.

1 Corinthians 9:24-25

ACKNOWLEDGMENTS

This work would not have been possible without the guidance and support of my mentor, advisor, and friend, Dr. Reinaldo Sanchez-Arias. His example of excellence at what he does, is one to chase; I hope to catch you one day.

Thanks to Dr. Grisselle Centeno and Dr. Harish Chintakunta who graciously agreed to be a part of the committee and gave me relevant feedback to better my work.

I would like to thank Florida Polytechnic University for their support, both in the form of scholarships and in the community that they foster on campus. I look forward to returning to campus and seeing how our small school is making a big impact.

Contents

List of Figures

List of Tables

Abstract

Unsupervised learning methods are applied to text data after applying modern graph based methods as preprocessing. Two datasets, technical reports on ambulance crashes from the NHTSA and social media threads from a reddit forum on mental health, are preprocessed into graphical representations. The graphical representations of the document can then be compared using modern graph kernel methods. The result of applying graph kernels is a matrix of similarity that can be used to perform hierarchical clustering. The datasets both differ in their writing style and their size; the NHTSA documents are large documents ($N = 48$) and the reddit threads are often small but there are many ($N = 1000$). Both datasets are analyzed with the methods described, and their results are compared.

Chapter 1: Introduction

Text mining is the process of extracting insight from text documents using computational and statistical methods. A common task in text mining is document clustering, where the natural language of the documents is reformatted and used to group documents into clusters of similar topics or text content. These type of tasks are known as unsupervised clustering. Unsupervised clustering is the use of algorithms to take unlabeled data and estimate potential clusters, or groupings, of the observations of concern; supervised learning methods, like support vector machines or logistic regression, use labeled data to "learn" how to discern between groups of data. Since the data are guiding the definition of clusters, or categories, this is an unsupervised approach. Popular methods to cluster text include k-means, hierarchical methods, and topic models. Most of these methods utilize term frequency-inverse document frequency measures or bag-of-words methods to do clustering, though these methods often divide words and thus may lose meaning, or context, surrounding the word of concern. In an effort to cluster documents while still preserving the relationship between words, text can be modeled with graphs which better preserve the context surrounding a word. Text can be represented with bigram graphs, where the edge between two vertices is the result of two words appearing adjacent in the text. This idea can be extended to more general n-grams, and further extended to skip-n-grams, resulting in even richer representations of the text. The similarity of two of graphical representations can then be assessed using modern methods called graph kernels. Graph kernels produce a measure of similarity between graphs, thus allowing for further machine learning to take place. With a measure of similarity between two graphs, we can do an assortment of clustering methods.

The chosen graph representation of the observational unit of text is an extension of

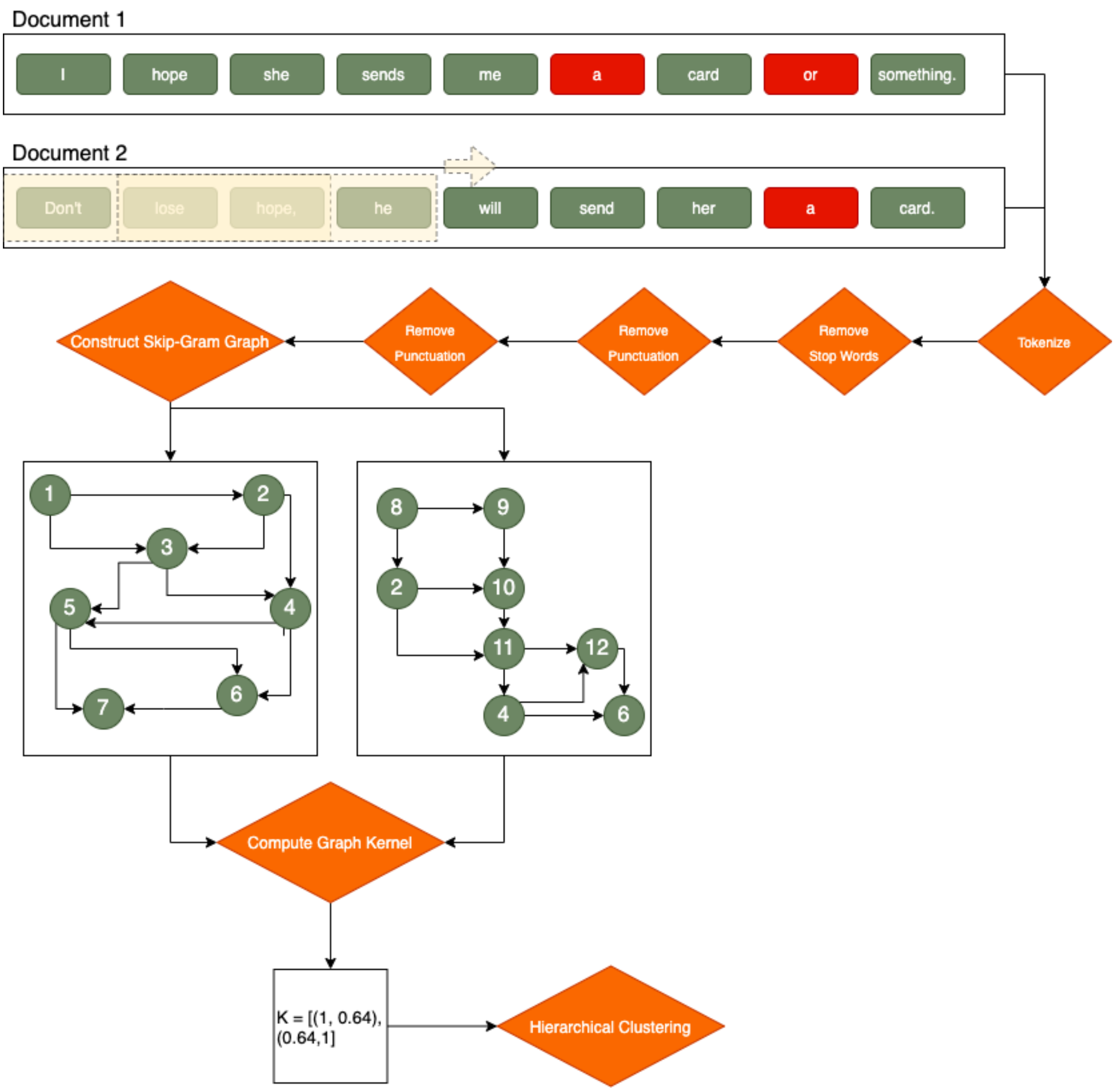

Figure 1.1: A map of concepts used and how the methods used here work together. Example is a skip-gram graph of $k = 2$.

bigram graphs. The use of skip-grams here is a way to connect ideas between words that would have normally attained meaning through the "context" of the sentence. By connecting words that appear close to one another while not being immediately adjacent, we are creating "skip-grams"– an extension of the bigram where the window for which words are considered adjacent is widened. A graphic in Figure 1.1 demonstrates this.

Many methods used in text mining use bag-of-words techniques, but these methods do not always preserve the context and relationships between words. The graph representations for text, used here, aim to preserve the relationship and context between words. The idea of bigrams (pairs of two words) can be extended to "skip-grams"; skip-grams are words which appear within some width w of each other. This leads to more connections, and a richer graph representation of the text. This skip-gram method is similar to that used in the popular machine learning package Word2Vec, which uses the "continuous bag of words" or "continuous skip-grams". In this paper, a graph representation of the text is constructed, using skip-gram methods, for each document in the datasets.

The graph representations are compared with a graph kernel, which is a measure of the similarity of two graphs. In this study, the graph kernel of choice is a "edge histogram kernel", because the kernel utilizes the labels of the vertices to assess the similarity of two graphs– not just the topology. Additionally, this is a computationally cheaper than other graph kernels. For each graph, the kernel produces a measure of similarity to the other $n - 1$ graphs in the dataset. We can use machine learning methods which work nicely with kernels, ~~i.e.~~ e.g., support vector machine (supervised) or hierarchical clustering (unsupervised). The hierarchical clustering methods will be applied here for two datasets, and the results compared.

Alternatively, we could use this similarity measure between two graphs, to assess how similar other graphs in the dataset may be. That is, we can compare the similarity of graphs B and C through their similarity to A. To do this, a KDE (Kernel Density Estimation) to estimate a kernel density curve, the curve is then used to partition the rest of the dataset into clusters. The KDE bandwidth is modified to produce more or less local minima/maxima that can be then used to identify potential clusters. Basic calculus can be used to locate to inflection points that will serve as intervals for which a cluster will be defined. These methods were developed, but were not included in the final analysis; they are available in the appendices and in a GitHub repository.

To test these methods, two different text data sets will be considered. First, reddit comments from subreddits pertaining to mental health. Using the {`redditExtractoR`} package (Rivera & Rivera, 2015), the comments are collected if they match criteria set by the query. From the subreddit `r/mentalhealth`, posts are collected if they have chosen keywords and at least 10 comments. The keywords considered are "anxious", "anxiety", "depressed", "depression", "mental", "illness", "scared", "afraid", "sad", "emotion", "anger", "angry", "upset", "suicide", "abuse", "emotional", "help", and "addiction". These words do not represent all words which define or indicate mental health topics may be present; these words were chosen by the researcher arbitrarily. The result of the queries is thousands of comments. Mental health continues to be a growing focus at the national level, and the discussion on reddit is often candid and open, due to reddit's reputation as an anonymous website. The second data set being considered in the study, is NHTSA report data. These reports are form the Special Crash Investigation (SCI) reports from the NHTSA (National Highway Traffic Safety Administration). The pdf documents that are considered in this study are those that involved ambulance(s) in the incidents highlighted in the reports. These events are considered "edge cases" and could prove useful to those working in autonomous vehicles.

These datasets were chosen because they are two very different styles of writing. The reddit posts are often in contemporary and casual use of language, possibly including "netspeak", while the NHTSA reports feature a technical writing style with plain language, absent of hyperbole or sarcasm. These datasets show two very different sides of text data, and comparisons will be made to how each dataset responds to the methods outlined above.

Throughout the analysis, all the code used and datasets were documented for future use. The code is well documented in the form of explanatory code notebooks (RMarkdown), code comments, and supplementary markdown files. Datasets, scripts, notebooks, and supplementary files are all made available through a public GitHub repository. Part of the primary focus while developing these methods was to produce them a manner which follows good practices for reproducibility and open source software development. The scripts can all be used in a manner that allows a workflow from raw text to clustered documents; the workflow is documented in the form of a "script map" or a process flow diagram, so new users can understand the process more clearly.

Chapter 2: Literature Review

The introduction of graph kernels first arose in 2002, from the paper "*Diffusion kernels on graphs and other discrete structures*" (Kondor & Lafferty, 2002). Since then, graph kernels have been used in a variety of fields, and modified to suit different problems. These developments have resulted in applications in biology, chemistry, social media analytics, and machine learning. Graph kernels have since formed into 3 large sub-groups: random walk kernels, sub-graph (also called graphlet) kernels, and tree-based methods. Within these three categories there are a multitude of modifications which include things such as edge or vertex attributes into the calculations (Vishwanathan, Schraudolph, Kondor, & Borgwardt, 2010). The graph kernels of particular interest to this study are those which incorporate vertex attributes into the calculation of the kernel. This inclusion of vertex attributes will allow the assignment of a word from a text document to a vertex.

The use of graph kernels in text processing has been largely focused on applying them to a neural networks for NLP. However, a 2017 paper by Nikolentzos et.al, covered how their use first arrived at the idea to apply graph kernels to what they called "graph-of-words" (Nikolentzos, Meladianos, Rousseau, Stavrakas, & Vazirgiannis, 2017). They utilized a window width to construct a connections between words, effectively "skipping" some words in between; this concept has also been called a "skip-gram" network in other contexts (Cheng, Greaves, & Warren, 2006). Members of this research group, and others close to them, have produced software packages to perform some basic graph kernel methods. In addition, there have been a small number of papers from the group about the application of these methods to text mining (Sugiyama, Ghisu, Llinares-López, & Borgwardt, 2018). Their work centered around use of neural networks for classification tasks.

Table 2.1: Summary of Literature Review

Topic	Author	Title	Year
Graph Kernels	Vishwanathan	Graph Kernels	2010
Graph Kernels	Kriege et al.	A survey on graph kernels	2020
Graph Kernels	Nikolentzos et al.	[graph kernels for doc. similarity]	2017
Graph Kernels	Kondor and Lafferty	Diffusion Kernels on Graphs [...]	2002
Skip-Grams	Cheng et al.	From n-gram to skipgram [...]	2006
Text Mining	Vazirgiannis et al.	GraphRep: [...]	2018
Application	Rosenfeld et al.	Kernel of Truth: [...]	2020
Software	Silge and Robinson	tidytext	2016
Software	Sugiyama et al.	graphkernels	2018
Software	Casardi et al.	igraph	2013

The literature review provided some valuable insights which drove some critical decisions for the methods presented here. Specifically, studies which showed that edge or vertex histogram kernels were the least computationally expensive (Sugiyama et al., 2018), while also being compatible with edge/vertex labels. This saved a lot of time which would have been spent searching for the graph kernel which would scale best to handle the large number of documents. Additionally, the graph kernel paper from Vishwanathan, and the survey paper form Kirege, allowed for clear understanding of the different types of graph kernel methods and their applications in a variety of fields, as well as pros and cons to each kernel.

by Kriege et al. \cite{ }?

Upon reviewing the literature surrounding each of these topics, it was clear there was much more work to do with these methods. Specifically, that the topic could use more published work on the use of graph kernels in conjunction with other machine learning methods, in this case hierarchical clustering. Also, this work provides two very different text datasets that can be used for additional studies on the application of these methods, as well as other natural language processing or text mining efforts. Most importantly, this work provides an end to end framework for text mining while using graphs for the R and the data science communities. These contributions will add to the existing body of work on graph kernels in the context of text mining and will lead to new development of using graph representation of text for machine learning.

is it referenced anywhere on the paper?

Chapter 3: Methods

3.1 Introduction

3.1.1 Skip-grams

As an alternative to natural language processing (NLP) methods, which are reliant on "bag-of-words" methods, the methods used here utilize a graph representation of the text. Consider a bigram, a pair of two words—like "hot dog" or "peanut butter", these bigrams can be constructed for a text document where every pair or adjacent words is a bigram. The bigrams can then be used to make a graph, where each word is a vertex, and each bigram is an edge. This graph representation holds more context than the bag-of-words methods; for example seeing the words "cake" and "carrot" in a bag of words may not show that "carrot cake" was the real intent of the text. This is an important concept for modeling text, as we should strive to achieve a representation of the text that makes for effective modeling that will capture the true meaning of the text in question. Keeping this in mind, with the example of "carrot cake", what about the idiom "beating a dead horse"? Each word individually may mean something other than the idiom. Even the bigrams "beating dead" and "dead horse" do not capture what the idiom means. We can expand the number of words in the n-gram to be 3 or 4 words, or alternatively, we can make more "edges" or connect more words. We can connect words that are not immediately adjacent but perhaps within k words away. These bigrams that appear within k words of each other are called "skip-grams". The skip-gram allows to capture context of larger sequences of words since the graph representation will show how the k wide neighborhood of words was connected. In the idiom example, using skip-grams with window width $k = 2$, and removing common words (e.g. "a", "at", "the"), will produce a graph like:

$$E(G) = \{\text{beat} \longleftrightarrow \text{dead}, \text{dead} \longleftrightarrow \text{horse}, \text{horse} \longleftrightarrow \text{beat}\}$$

This graph representation contains a cycle, of length 3, where most native english speakers will identify the meaning behind the graph representation. As ideas, idioms, figures of speech, and other concepts (that may be explained in a non-literal fashion) grow in size as they include more words, it becomes more difficult to capture the meaning behind the text. However, leveraging the concept of a skip gram can produce such a rich graph representation of the text that the original meaning is more likely to be preserved. Other research has shown that use of skip-grams for text modeling leads to less data sparsity and mitigates issues of variety in large corpuses through modeling text in this way. The skip-grams are shown to preserve more context than traditional bag-of-words methods that use words as the token of choice (?, ?). Fix

3.1.2 Graph Kernels

The next natural question is, "how can we compare these graph representations?", and we address this with graph kernel methods. These methods are generally used to compare the similarity of graphs. These use of a graph kernel to compare graphs was first published in 2003, and since then various applications and adaptations have been made to the methods. In the case of text mining, the graph kernel must assess vertex labels —if one intends to map words to vertices, otherwise they will be assessing the topology alone. In this study, the Edge-Histogram kernel is the kernel used to compute similarity. This kernel was chosen as it uses labels on the graph structure, and is not as computationally intensive as other methods (Sugiyama & Borgwardt, 2015). In the specific implementation used for these studies, the computation time was the shortest when compared with other kernel methods like: graphlet, random walk, and Weisfeiler-Lehman kernel (Sugiyama & Borgwardt, 2015). Since the data sets of concern in the studies feature either large graphs or a large number of graphs, the kernel had to be cheap computationally.

To compute and edge histogram kernel on two graphs, G_1 and G_2, first define the set of edges $E_i = \{(u_1, v_1), (u_2, v_2), ..., (u_n, v_n)\}$ where (u_n, v_n) is the n-th edge connecting u_n to v_n. Then the edge label histogram is defined to be $\vec{g} = \{g_1, g_2, ..., g_s\}$, so that g contains the number of times each edge label appeared. In the case of graphical representations of text, the number of times a skip-gram appears is not considered; it either appeared or did not. For this reason, a Manhattan distance is chosen, as opposed to a euclidean or similar distance metric, since the Manhattan distance measures distance along a grid—like Manhattan city blocks from point A to point B. Since the data are all on a grid in essence, due to the binary nature of either having a label or not, the Manhattan distance is a natural fit here. The kernel is then the sum or the product of each element in the g vectors for each G_1 and G_2 in the case of a linear kernel (Sugiyama & Borgwardt, 2015).

3.1.3 Using Kernel for Clustering

The output of the kernel is useful for a variety of tasks. Some other popular applications have included classification with support vector machines, which are popular with other kernel methods. In this case, the kernel is used for unsupervised clustering. Within the kernel matrix, K, the entry $k_{i,j}$ represents the similarity between graphs i and j. This matrix which contains measures of similarity between points can be used as a distance matrix for hierarchical clustering. Before using the graph kernel as a distance matrix, normalization or standardization takes place, and principal component analysis may be used. The end result is each row is a single graph-document being described by its similarity to all the other graphs, which are the column values. Once the values are transformed or rotated by preprocessing methods, the points are just represented by their similarity to one another, but in a transformed space. Various hyper parameters can be tuned for successful clustering; the graph kernel has a parameter that can be tuned, and the hierarchical clustering can be tried with differing types of linkage.

3.2 Processing Text into Graphs

For both datasets, the reddit comments and the SCI papers, the text data is collected and stored in its raw form for reproducibility. The text is then cleaned and prepared for analysis using a suite of text processing tools from the {`tidytext`} package for R (Silge & Robinson, 2016). Using this package, the text undergoes tokenization into skip-grams, stop word removal, and filtering to remove punctuation, numbers, short words, etc.

fix

Skip-grams are produced for $k = 2, 3, 4$, where k is the window width of the skip-gram for which bigrams are formed. Stop words are gathered from popular lexicons: "snowball", "SMART", and "onix" (?, ?). As stated above, all numeric values, and lingering punctuation were also removed through use of the {`stringR`} package (Wickham, 2010). This processing is an essential preprocessing step for using graph kernels to gauge similarity of documents, because words and strings (like punctuation or numbers) that get repeated frequently across many documents in the set do not actually mean the documents are similar—they just have common reoccurring words. Removing these types of words forces the text data to be more unique across the document set.

The result of the skip-gram tokenizer is a data frame where each observation is a bigram, a pair of words, that the skip-gram window captured. The data frame is then cleaned of stop words through using anti-joins on each bigram; any row with the appearance of a stop word in either position, the first or second word, was removed. The same method was applied for punctuation and numbers. The result was a data frame of bigrams which appeared in a fixed window width, k, of one another and where both words are of a length greater than 3 letters, not a stop word, and do not contain numbers or punctuation. This cleaned dataset is now ready to be converted into a graph object.

Now that the skip-grams are cleaned, the data frame can be converted to a graph object, through use of the {igraph} package (M. G. Csardi, 2013). To do this, each word pair in the data frame is converted to an edge and vertex pair in the graph object. For example, the word pair "data frame" will become two vertices, labeled "data" and "frame", with an undirected edge connecting them. In this study, directed graph edges are not used, but could be considered in future iterations of this work. When this completed for all the skip-gram pairs that were generated, it produces a singular connected graph, however a great deal of filtering occurred and there may be disconnected portions. When words are removed from the graph, a vertex will disappear and can potentially split the graph. This is not too common in the NHTSA dataset for two reasons. First, the text data is quite long, and so if a word is reused at a later time in the text there will be an additional connection to keep it included in the main graph. Secondly, the benefit of the skip-gram, as opposed to plain bigrams, is that the larger window width means words get connected and can "skip" over words that will get removed through the data cleaning process. So at this point, if there is a group of words that are isolated and not connected to the main graph, they are often quite small and are only several words. To keep computation simple, these lingering small isolated graphs are removed. The vertices that are not members of the main graph are removed from the graph object and that leaves a single connected graph for each text document. These graphs can be compared with graph kernels easily now.

3.3 Graph Kernels for Similarity

Graph kernels can be used to compare the similarity of two graphs. The development of these methods arose out of the need to determine if graphs were isomorphic in a faster way. The solution was a graph kernel which produces a scalar value for how similar two, or more, graphs are. The result of a graph kernel is a matrix where the similarity of graphs i and j is in the kernel's i-th row and j-th column entry. This resulting matrix can be used in a variety of ways, but it will be used as a distance matrix in applications here.

Amongst graph kernels which consider edge or vertex labels in their computation, various surveys and studies have found edge label histogram kernels or vertex label histograms to be the most efficient. They may not out perform the other methods, like a Weisfeiler-Lehman or other subgraph methods, but they are computationally cheap. The datasets being studied here are both large: one contains many smaller graphs, and one contains 48 very large graphs. So for this study, computation efficiency was prioritized.

The edge label histogram kernel is the graph kernel that was chosen for these datasets, and it can be computed using either a linear kernel or a Guassian radial basis function (RBF) kernel between the edge label histograms.

An edge label histogram is defined as $\vec{g} = (g_1, g_2, ...g_i)$ such that $g_i = |\{(u,v) \in E | \phi(u,v) = i\}|$ for each i (Sugiyama & Borgwardt, 2015). Where g_i is a histogram bin for a unique edge label's magnitude, E is the set of edges, and ϕ is a function that maps each label to a scalar value in the range of unique values. The edge label histograms are then passed through a kernel, either a linear kernel or a Gaussian RBF kernel.

Computation using a linear kernel takes two graphs, G and G', and uses their edge label histograms $\vec{g}$ and $\vec{g'}$. The kernel is computed as:

$$K(\vec{g}, \vec{g'}) = \vec{g}^T \vec{g'}$$

The resulting value is stored in the graph kernel matrix as the measure of similarity between the two graphs in the corresponding row and column for the pair (Sugiyama & Borgwardt, 2015).

Alternatively, the Gaussian RBF kernel takes the edge label histograms of G and G', that we call $\vec{g}$ and $\vec{g'}$, and the kernel is computed as:

$$K(\vec{g}, \vec{g'}) = e^{-\left(\frac{||\vec{g}-\vec{g'}||^2}{2\sigma^2}\right)}$$ \left(\right)

The resulting value is stored in the graph kernel matrix as the measure of similarity between the two graphs in the corresponding row and column for the pair.

Through either of these kernels, we obtain a kernel of dimensions $n \times n$ for a list of n graphs. This kernel can then be used for clustering methods. (maybe that way both equations fit in this page)

3.4 Clustering with Graph Kernels

The resulting graph kernel matrix, with dimensions $n \times n$, is high dimensional data. We can take this kernel and perform a principal component analysis (PCA) to reduce the high dimensionality of this data. Each row, can be considered as an observation, and each column as ~~it's~~ its similarity value or in the j-th graph's dimension. PCA then allows us to express the data in a lower dimensionality, hopefully in 2 or 3 dimensions which can be visualized much better.

Then, this transformed kernel data is then used in place of a distance matrix for hierarchical clustering. Since the kernel is a similarity kernel, i.e. larger values mean values are closer or more similar, and not a distance kernel, we need to adjust the kernel. The kernel's elements are passed through the function $f(x) = \frac{1}{x}$ to get the reciprocal value for each similarity value—thus converting the value to a value representative of distance. The hierarchical clustering is performed with the manhattan distance:

a previous eq. needs numbering

$$d(\vec{p}, \vec{q}) = \sum_{i=1}^{n} |p_i - q_i| \tag{3.1}$$

Since the edge label histograms have entries of either 1 or 0, the manhattan distance was a better match for the space. The hierarchical clustering also uses a linkage method of Ward's method, for minimizing variance. Ward's method is defined as:

$$d(\vec{p}, \vec{q}) = ||\vec{p} - \vec{q}||^2 \tag{3.2}$$

The resulting dendrograms are then able to be visualized, analyzed, and cut to form clusters of the documents. In the dendrograms, each document is a "leaf", or end of the tree, and each dendrogram represents the corpus as a whole. Clusters are then formed when the tree is "cut" at some height; the clusters are then what are of interest to the researcher, as they are the result of this unsupervised learning method.

3.5 Development of Methods

All of the methods outlined in preceding chapters were developed in *RStudio* and are stored in a public code repository on *GitHub*. Initial development and testing of all the packages and libraries was completed using *R Markdown* notebooks, which allow for coding in chunks with explanation in markdown text. These netbooks are kept in the code repo to serve as more explanatory documents demonstrating how all the scripts and datasets work in conjunction with one another.

The resulting and final scripts are kept in the code repository as well. The final scripts were developed to use the output from one another and to produce intermediate outputs so that small studies could be conducted. How the scripts work with one another is mapped out in the "Script Map" in Figure 3.1.

The Script Map shows how these R scripts can be used for either standard computation, in the case of the NHTSA dataset, or parallel computation, as was used in the reddit dataset. Parallel computation was completed through use of the {`furrr`}, an R package that makes parallization of functions similar to mapping functions from the popular {`purrr`} package. The {`furrr`} package works by splitting the computation among "workers", or cores, in the CPU. This takes full advantage of the processing capabilities of the machine; instead of letting 1 core do all of the work, we can use many

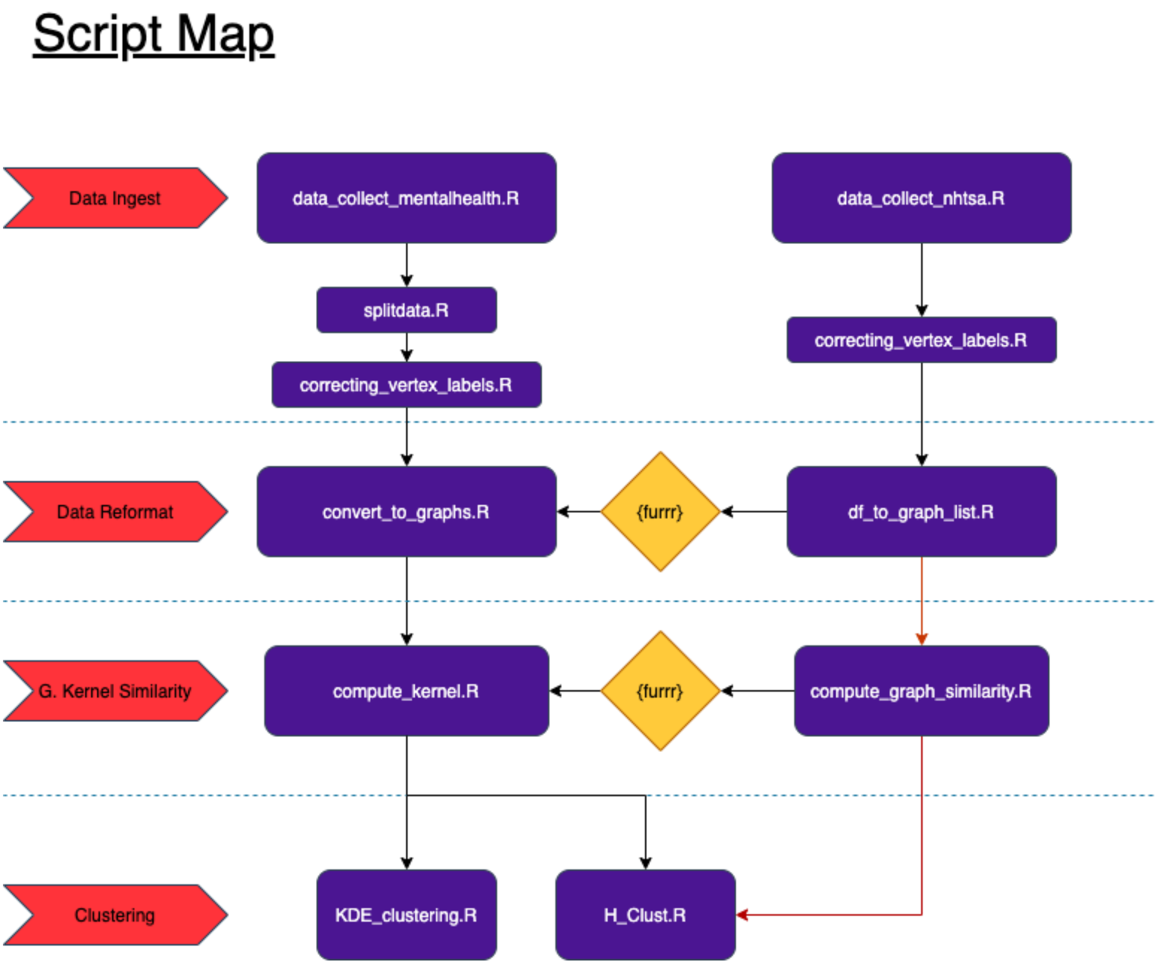

Figure 3.1: Development script map that describes how all modules work together from data ingest to final analysis.

more cores to complete the computation faster. Implementation of a parallel option was necessary because computational time for a list of graph objects grows rapidly as the number of graphs in the list increases. A small study was conducted on the reddit dataset to plot this relationship.

In Figure 3.2, there is strong evidence that the computation time follows $O(e^n)$, or worse. For this reason, the parallel workflow was developed—to handle datasets with a large number of documents to compare. Even though the amount of text per document in the NHTSA dataset is much larger than the amount of text per document in the reddit dataset, the reddit dataset takes longer to compute than the NHTSA dataset.

Referring back to the Script Map, either workflow is decomposed into four steps: Data Ingest, Data Reform, Graph Kernel Similarity, and Clustering. This is the basic work flow developed here. There are some things to note in the Script Map.

First, vertex labels need corrected to work with the {`graphkernels`} package. This is done through reassignment of all text vertex labels to be a scalar value. The labels are mapped from the union of two graphs' labels to values ranging from $1 - n$, where n is the size of the text label set from the union. This script is run within the `convert_to_graphs.R` script, so it operates on two graphs at a time as it iterates through the whole graph set of concern.

Second, the scripts routed through {`furrr`} are functions which are parallelized in the `convert_to_graphs.R` and `compute_kernel.R` scripts. These parallelized scripts are done so through a function being executed with no argument, so the dataset which they consider is hard coded into the script. This is not good from a reproducibility aspect, but it is well documented in the code and in the repository, and has reproducible results.

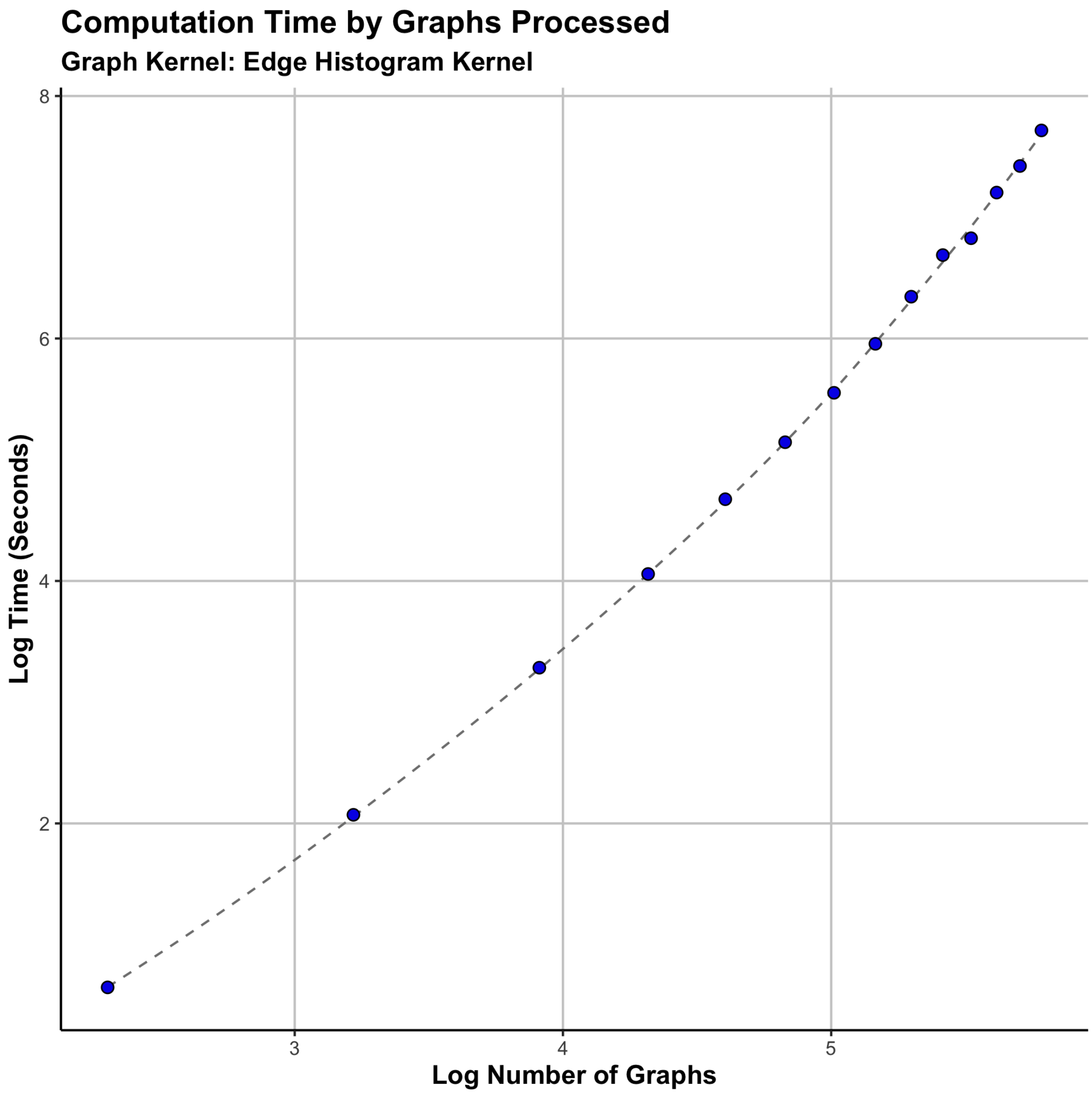

Figure 3.2: Small study on computation time as a function of the number of graphs in list to be processed. Data from reddit used.

Third, the `splitdata.R` function is there to simply split up the dataset from reddit, which would be too large to store in the repository on GitHub. This segmentation of the data still allows for workflow and reproducibility, since the datasets can all be read in to the environment all the same.

Lastly, a KDE clustering analysis was not performed on the NHTSA dataset, as that dataset has few observations ($N = 48$) and will not perform well with the density estimation methods developed here; there would be too few observations to generate a number of local maxima/minima needed to apply the clustering methods developed. Results here were not included, but the scripts and methods are available in the Appendix and code repository.

Chapter 4: Results and Analysis

Hierarchical Clustering was performed on both of the reddit and NHTSA datasets. These clustering methods lend themselves to be used with similarity/distance matrices, and the results shown here can be displayed in a variety of formats. One can learn more about the clustering results through dendrograms, n-gram graphs~~,~~ and term-frequency/inverse document-frequency.

4.1 NHTSA Special Crash Investigations

For the NHTSA's Special Crash Investigation dataset, a hierarchical clustering was performed on the resulting graph kernel computed from the skip-gram graphs. Since this dataset was small at $N = 48$, it served as a good subject to study the interactions between modifying the skip window width, kernel hyper parameters, and number of clusters. This study on these interactions informs decisions made on which hyper parameters to use in final analysis of the NHTSA data as well as the reddit thread data.

For the study on these interactions, hyper-parameters were varied and the value of cluster within sum of squares was tracked. Specifically, the values in Figure 4.1 were tested. In Figure 4.1, we see some promising values occurring at more prominent "elbow" points in the plots. These points will be further analyzed and compared. The hyper parameters for these points are:

These hyper-parameter sets, and their corresponding graph kernel matrix, are used in a hierarchical clustering analysis. The results from the analysis perform well, and display prominent clusters, often of comparable sizes.In Figure 4.2, we see the 5 dendrograms produced from hyper-parameter sets A, B, C, D, and E. Upon some examination, one will notice that documents often appear in the same clusters together,

Point	Skip-gram k	Graph Kernel sigma	No. of Clusters
A	3	1000	5
B	2	1100	3
C	3	1100	5
D	3	800	6
E	3	900	7

Table 4.1: Hyper-parameter for Variation Study in Figure X

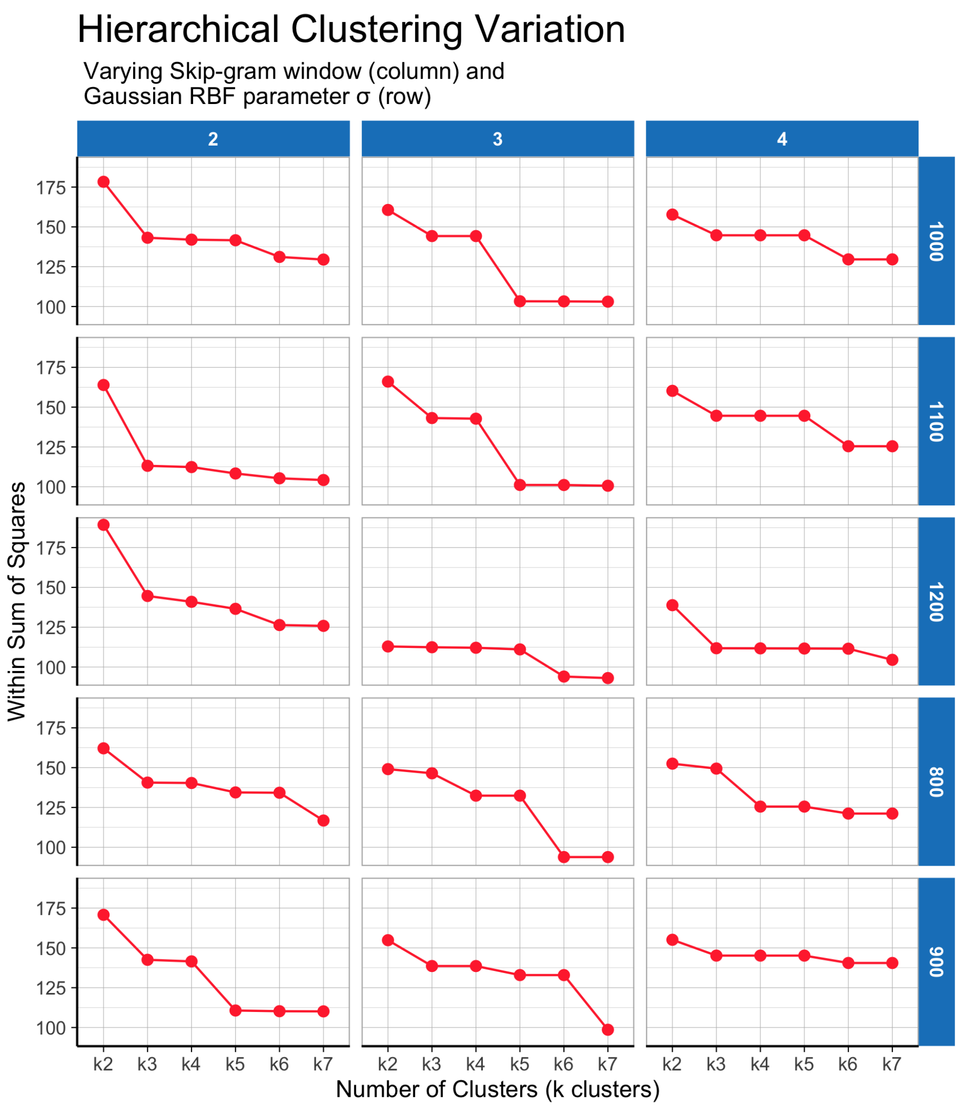

Figure 4.1: Hierarchical Clustering Variation (NHTSA) for differing hyper-parameters.

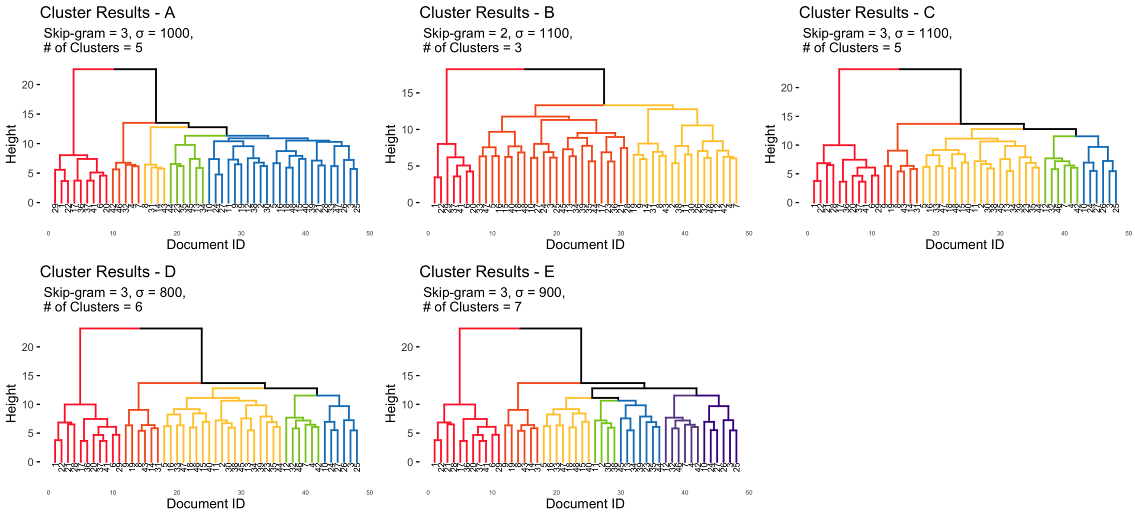

Figure 4.2: Five hyper-parameter sets and their corresponding dendrograms.

regardless of the hyper-parameter sets. This is most encouraging, as it indicates that these clusters are not a result of user-chosen parameters, but that the documents are indeed similar.

To compare how often this co-membership in groups was appearing, a matrix of co-membership was created. In Figure 4.3, we see that each document has a set of other documents with which it is always clustered with, regardless of hyper-parameter configurations. These small document sets are the document sets which we can expect to have the most in common with one another.

We can inspect the results by checking some of the co-members' text. For example, document two (which was an ambulance crash in Angola, Delaware), has only 3 other co-members. Document two's co-members are: document 11, document 30, and document 38. Document 11 was an ambulance crash in Sheridan, Indiana. Document 30 was an ambulance crash in Pasadena, California. Document 38 was an ambulance crash in Ocilla, Georgia.

Now, we can examine some information about these four incidents that may explain their clustering. For example, all of these incidents had fatalities, three of them had roll

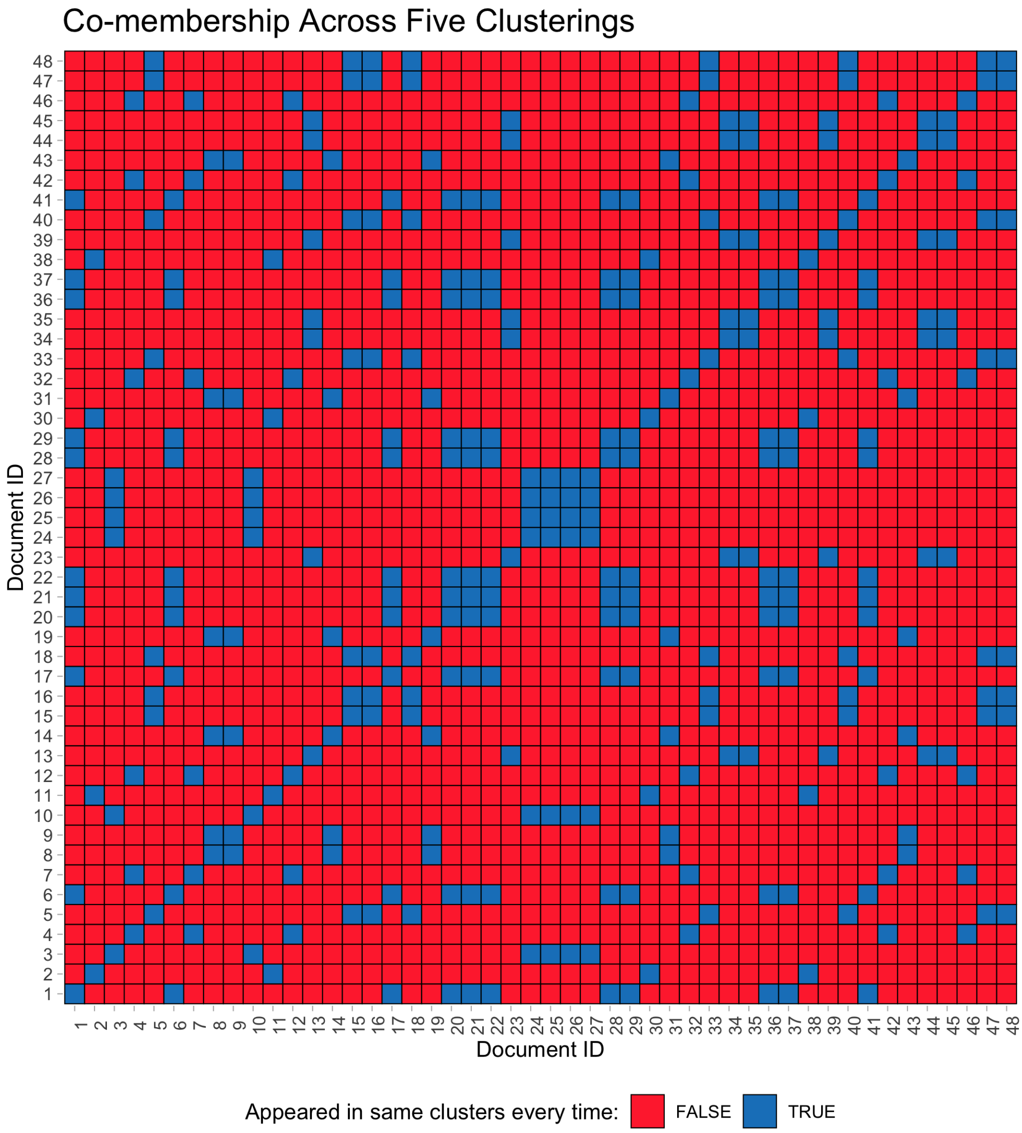

Figure 4.3: Co-membership of documents across hyper-parameter sets A, B, C, D, and E.

over events, and these four were all front end collisions to the ambulance where they struck another vehicle, or object, head on. While this is just one example of the clustering methods identifying similarities in the text, we can examine similarities across the dataset through use of term-frequency/inverse document frequency.

$$\frac{tf}{idf} = \frac{\frac{\text{Term Frequency in Document}}{\text{Total Words in Document}}}{\log_2(\frac{\text{Total Documents in Corpus}}{DocumentswithTerm})} \tag{4.1}$$

maybe just TF-IDF
(consistent with glossary)

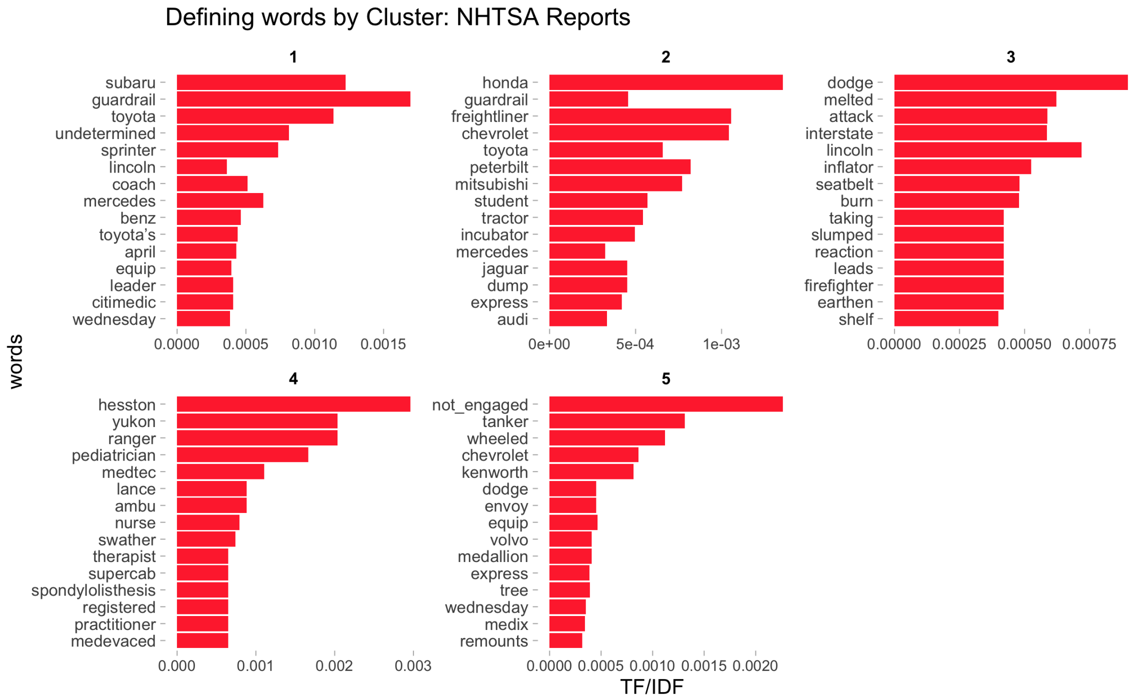

Figure 4.4: Term-Frequency/Inverse Document-Frequency for NHTSA reports, using hyper parameter set A.

In Figure 4.4, we see that words that appear frequently, and are unique to that cluster appear to be vehicle manufacturers, medical terms, and words that describe a crash situation (e.g. guardrail, tree, interstate). We can use these to get an idea of the crash situation. Alternatively, we can also examine portions of the skip-gram graph produced, which was used for the graph kernel calculation.

We can put together some of the ideas and topics discussed in the cluster. For example, in Figure 4.5 we see that "Front Left Passenger", "Rear Facing Seat", and "Patient Compartment" are all skip-grams which appeared frequently in this cluster.

Cluster #2

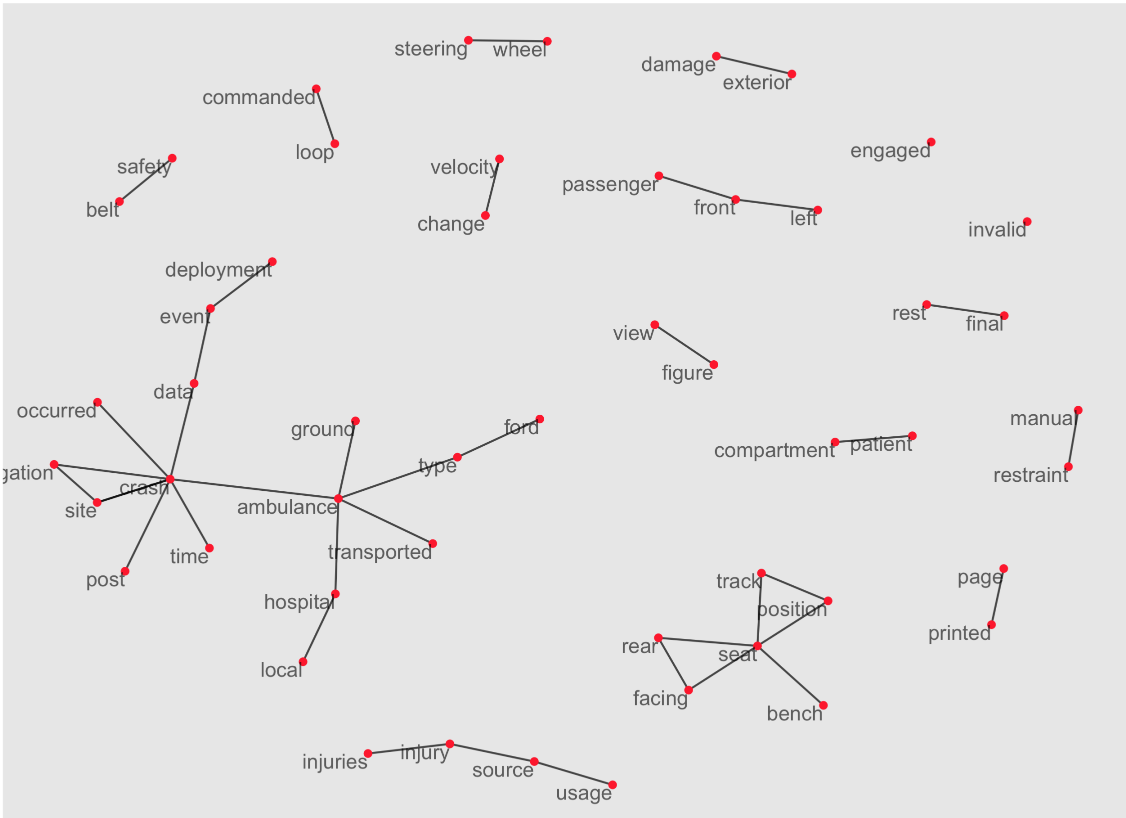

Figure 4.5: Skip-grams which occurred more than 50 times in hierarchical clustering with hyper-parameter set A, in cluster 3.

4.2 reddit Threads

For the reddit threads dataset, analysis was focused on the ability of the clusterings to organize threads based on their query words. The words used to query the `r/MentalHealth` subreddit (`https://www.reddit.com/r/mentalhealth`) all returned threads which were then clustered into new groups. Some of these clusters expressed strong preference for posts that corresponded with specific key words.

This data was collected with the {`redditExtractoR`} package (Rivera & Rivera, 2015), and then we augmented and cleaned with {`tidytext`} (Silge & Robinson, 2016) and other {`tidyverse`} tools (Wickham et al., 2019). These tools made it easy to extract the data from reddit, tidy the data, and perform text processing tasks such as tokenization and parsing for punctuation or numbers. Once the data was in a clean format, it followed the script map to get to a format where it could be analyzed in the same fashion as the NHTSA data was. Since this data was a large set, it utilized the scripts set up with {`furrr`} to speed up computation to under 20 minutes (Bengtsson, 2020).

To assess how many clusters to use, an "elbow plot" was formed similar to those in Figure 4.1, see Figure 4.6. For the reddit threads, the hyper parameters from the NHTSA analysis were used, and the consistently lowest metric parameter set was chosen; skip-grams of $k = 3$ and kernel parameter $\sigma = 1200$ were used in the reddit clustering analysis. In Figure 4.6, we see that 4 clusters is the optimal value, where additional clusters provides diminishing results.

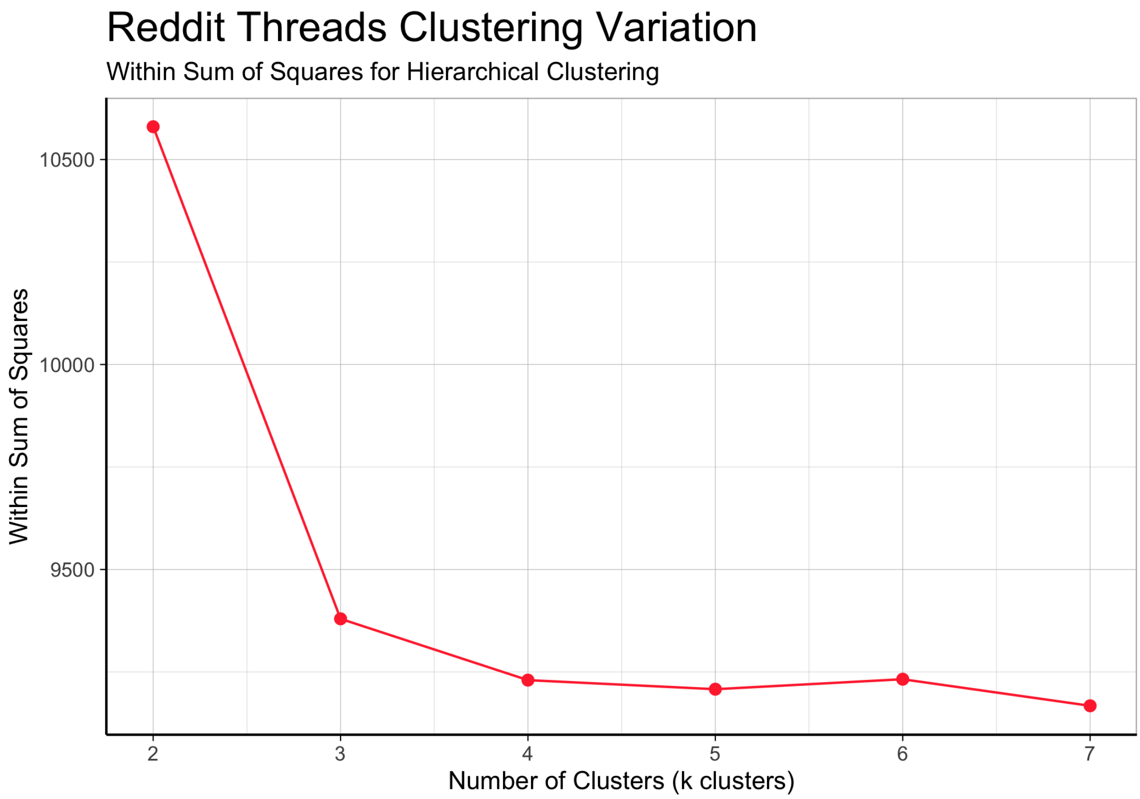

Figure 4.6: Within sum of squares as a function of number of clusters

After computing the kernel and using these parameters, four clusters are produced. Examining the dendrogram, in Figure 4.7, we see that we have 2 very distinct groups, which then break into two smaller groups. These groups are very clear and provide confidence in the clustering results. Along side the dendrogram, there is a heat map of what proportion of documents (or threads) in the cluster came from a query word. In Figure 4.7, we see that clusters 3 and 4 collected posts from the "angry" query word. Additionally cluster 1 displays a strong amount of its posts coming from the "sad" query word.

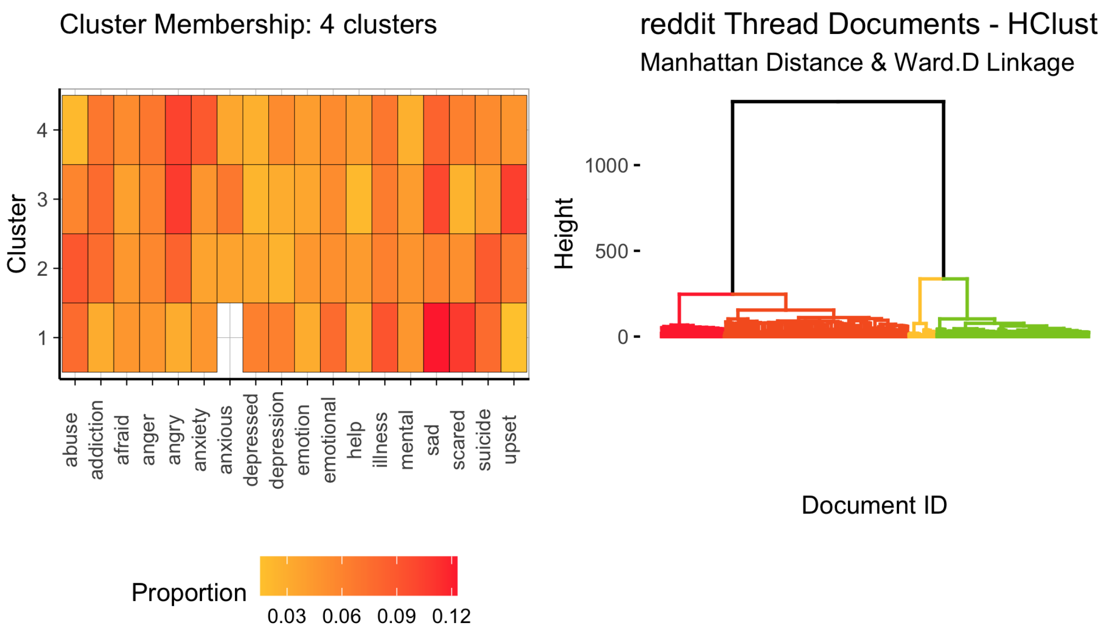

Figure 4.7: Left - Proportion of cluster's threads coming from each query word. Right - Dendrogram for reddit thread analysis, with 4 clusters.

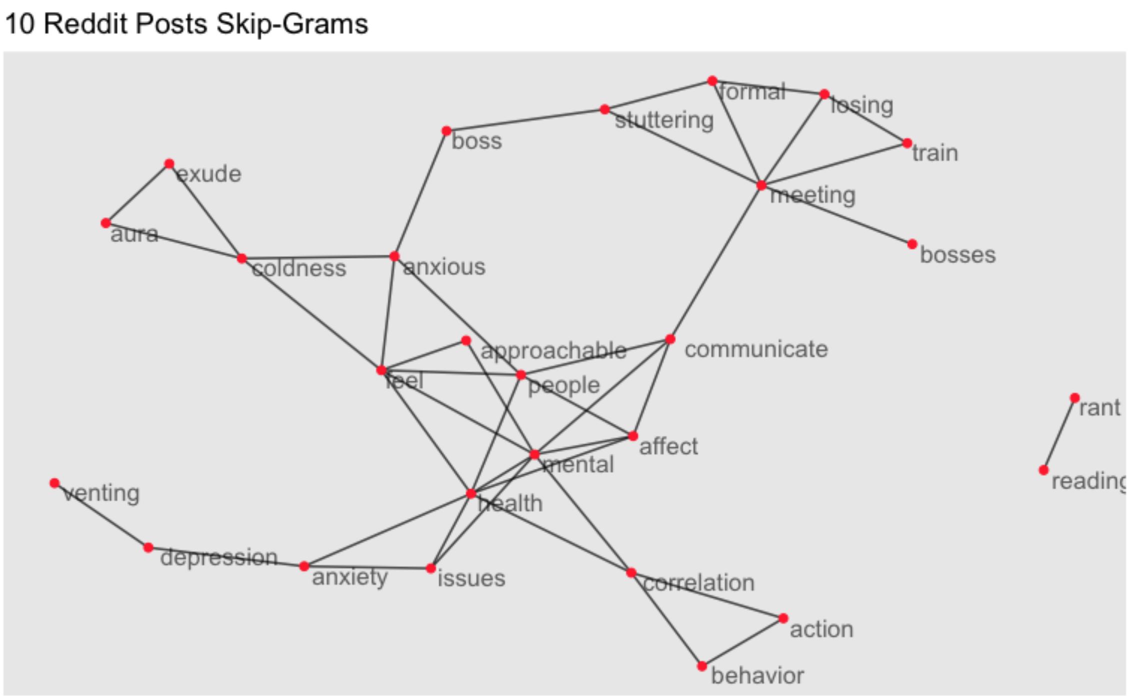

Figure 4.8: 10 Reddit posts and their skip-grams which occurred 8 or more times.

The prominence of some of these query words in the clusters indicates that the tone or language used by the thread authors was picked out by the clustering methods. This is of particular interest, as the goal of these methods was to perform unsupervised text clustering which utilized more of the rich intricacies of language.

4.3 Comparative Analysis

Looking at the results from both hierarchical clusterings on the graph kernels it is apparent that these methods work much better on larger documents. The performance of the graph kernels being applied to the skip-gram graph and then clustering was much better in the NHTSA dataset than the reddit threads dataset. By measuring within sum of squares, the NHTSA data set performed much better, as its WSS to number of observations ratio was 2.08 at its lowest, whereas the reddit dataset had a WSS to number of observation ratio of 9.

As for computational time, the NHTSA dataset took less than 2 minutes to compute, whereas the reddit dataset took just under 20 minutes to compute with the parallel processing. However, the reddit dataset displayed more prominent clusters, which was also indicative of success.

Comparing the elbow plots for both the NHTSA and the reddit threads datasets, it is clear that the optimal number of clusters for these methods will be dependent upon the dataset being analyzed. The results from the NHTSA study that indicated that skip-grams with $k = 3$ did seem to apply to the reddit dataset as well. These hyper parameters will need tuning for any new application.

Overall, the largest difference between the two analyses was that although the reddit threads displayed strong, prominent clusters, the NHTSA dataset was being computed in reasonable amount of time. Since the application of these methods are scaling better for increasing document size instead of increasing document lists, there is a case to be made that aggregating reddit threads into larger documents may be more effective, but to keep the observational unit tied to a reddit thread, this was not completed.

These methods could be applied in any text mining application. Since the methods were able to be parallelized during their most computationally expensive portions of code, the methods could scale with sufficient hardware. In a high performance computing environment, these methods could either be applied in batch/scheduled computations or in a stream/real time application (with some modification). The readiness for HPC environments is important as it allows the code to scale and be used in much larger analytic applications.

4.4 Contributions to the Field

Through this work, several new contributions have been made to the field. First, a specific study on the use of graph kernels as preprocessing for hierarchical clustering; the combination of these ideas has not been well documented within the literature reviewed here. Second, the analysis of such differing datasets will bring new perspective to the topic as it relates to the performance of such methods on different types of text data. Thirdly, the workflow developed here was made open for others to use, edit, and analyze for future applications. All of the scripts, notebooks, supplementary materials, and data being made open encourages further development by the community. Lastly, the results of the analysis will go on to inform their respective subfields about the exciting applications of these methods. In the case of NHTSA, edge cases that can be summarized through these methods can lead to more representative simulations being made that can lead to better trained autonomous vehicles. For the reddit mental health data, the work completed here can be used for semi-supervised applications to classify discussion in these type of subreddits. The classification can inform mental health practitioners about the state of mental health within such communities at scale. They can then use those results to drive mental health awareness campaigns, better target what the community is feeling about treatment, or perhaps identify new language or a new subgroup of people dealing with a specific mental health issue. These contributions will support furthering research in this field, as well as the fields from which the data is of interest to.

Chapter 5: Conclusion

Though applying several different methods in conjunction with one another (skip-grams, graph kernels, and clustering), unsupervised clustering can be done on text while preserving as much of the rich information from the original text as possible.

The NHTSA results, and their clusters, can be used to inform other research initiatives of similar scenarios for ambulance crashes. Identifying these clusters, providing summarized results, like n-gram graphs and term frequency/inverse document frequency lists, allows for information about similar crash reports to be compiled. This compiled information can then be used to generate more "edge-case" scenarios for Florida Polytechnic University's Autonomous Mobility Institute and their work on autonomous vehicles; rare cases are not always available in the datasets used to train autonomous driving models, and this kind of research could help inform their decision making moving forward.

In the reddit dataset analysis, the large samples ($N = 1000$) used in the study here were accelerated through use of parallel programming and made this analysis possible. The results of the analysis showed that prominent clusters were formed. The dendrogram in Figure 4.7 displays very encouraging results, being that the clusters not only stood out from one another, but that the clusters showed that they were collecting some threads according to their query word that was used to gather the original posts/threads.

In summary, the methods outlined here will continue to develop into more mature technologies that could be applied to a wide range of natural language processing tasks, being that the graph kernel matrix could be used for a variety of machine learning tasks—not just hierarchical clustering. These methods can be used in semi-supervised and supervised applications as well. This work opens the door to classification and inference using the graph kernel methods to preprocess text as graph representations. Once these methods are rebuilt into a clean package for use in high performance computing environments, they will see use in industries that process large amounts of text data.

Lastly, building upon the published work from 2020, these methods could be used for intelligent navigation of comments and posts on social media. In a previous paper, “A Framework for Intelligent Navigation Using Latent Dirichlet Allocation on Reddit Posts about Opiates”, \cite{} smart ways to cluster text and how that could be used to filter harmful, or otherwise unwanted, content from a web user were outlined. These same principles could be applied with the methods applied to the reddit threads here. Additionally, the NHTSA analysis will benefit Florida Polytechnic University’s AMI, which study autonomous vehicles. The edge cases which are often absent from training datasets, such as ambulance crashes, can be used to better inform their efforts to improve autonomous vehicles. The many ambulance accidents can be clustered so that instead of recreating 50 ambulance incidents, they can select one or two that represents each cluster.

These methods are all available, publicly, on a GitHub repository at `https://github.com/Levi-Nicklas/GraphDocNLP`. Updates may be made and the software written can be used, adapted, recreated, scrutinized, and more by the software development and data science communities that use GitHub. This open sourcing of the software leave the work conducted here open to examination, and will lead to better open science for the community.

Chapter 6: Future Work

Future developments on this project would be focusing on adapting the methods to scale on high performance computing (HPC) environments. The methods currently do no do well as the number of graphs for comparison grows. While the methods do well as the size of the graph scales up, the issue of many graphs can be addressed through parallel processing for applications at scale. The code has already been structured to handle parallel processing, and could be further prepared for a HPC through containerization.

Additionally, supervised and semi-supervised learning methods could be explored on these datasets. Using the clustering results as labels would lead to a semi-supervised model. For classification applications, use of support vector machines in conjunction with these graph kernel methods is well documented, and would be a natural next step following this work. Once the datasets are labeled, it would be an easy study to complete. The NHTSA data set, with 48 observations, could be labeled with a number of different dependent variables that could be gleaned from the crash reports.

The reddit thread dataset, as large as it is, may be a good candidate for a semi-supervised application. The clustering results from this study could be applied in a supervised application on reddit threads from the same subreddit, and efficacy could be assessed.

Lastly, these methods used here consider edge histogram kernels exclusively, because they are computational cheaper than other methods, and exploration of the other kernel methods which utilize edge/vertex labels would be an important study to conduct.

Appendices

Appendix A: Glossary

- Unsupervised Learning: the use of algorithms on unlabeled data to obtain clusters, or groupings, that partition the data.
- Supervised Learning: the use of algorithms on labeled data to classify (or regress) data into categories (or estimate values).
- Hierarchical Clustering: use of agglomerative methods to group observations into a hierarchy; result is often displayed as a tree.
- Bag-of-words: text mining methods which rely on sampling of words from a "bag" and make inferences about how words represent documents.
- Bigram/N-Gram: a pair of two words which appear adjacent to one another in text. Generalized to n words appearing adjacent to one another, in sequence.
- Kernel: a map between two spaces. In the context of this paper, the kernel takes the input set and maps it to a scalar.
- Net-speak: a term referring to the shorthand, slang, or other community specific language used in online settings.
- Vertex/Edge Attribute: a value or label which is associated with a vertex or edge of a graph object.
- Manhattan Distance: a measure of distance that is "the distance travelled along a grid"; this is ~~like~~ similar to expressing distance in city blocks.
- Dendrogram: a tree, in this case the output of a hierarchical clustering algorithm.
- TF-IDF: term frequency to inverse document frequency ratio. The metric measures the "importance" of a word to a document relative to the rest of the corpus.

Appendix B: Session Info - R

R version 4.0.3 (2020-10-10)

Platform: x86_64-apple-darwin17.0 (64-bit)

Running under: macOS Catalina 10.15.6

Keep in margin

Matrix products: default

BLAS:

~~/System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/~~vecLib.framework/V

LAPACK:

/Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

locale:

[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:

[1] stats graphics grDevices utils datasets methods base

other attached packages:

[1] tidytext_0.2.6 graphkernels_1.6 igraph_1.2.6 here_1.0.0

[5] forcats_0.5.0 stringr_1.4.0 dplyr_1.0.2 purrr_0.3.4

[9] readr_1.4.0 tidyr_1.1.2 tibble_3.0.4 ggplot2_3.3.3

[13] tidyverse_1.3.0

loaded via a namespace (and not attached):

[1] Rcpp_1.0.5 cellranger_1.1.0 pillar_1.4.7 compiler_4.0.3

[5] dbplyr_2.0.0 tokenizers_0.2.1 tools_4.0.3 lattice_0.20-41

[9] jsonlite_1.7.1 lubridate_1.7.9.2 lifecycle_0.2.0 gtable_0.3.0

[13] pkgconfig_2.0.3 rlang_0.4.9 Matrix_1.2-18 reprex_0.3.0

```
[17] cli\_2.2.0        DBI\_1.1.0        rstudioapi\_0.13 haven\_2.3.1
[21] xfun\_0.19        janeaustenr\_0.1.5 withr\_2.3.0    xml2\_1.3.2
[25] httr\_1.4.2       knitr\_1.30        fs\_1.5.0       generics\_0.1.0
[29] vctrs\_0.3.5      hms\_0.5.3         rprojroot\_2.0.2 grid\_4.0.3
[33] tidyselect\_1.1.0 glue\_1.4.2        R6\_2.5.0       fansi\_0.4.1
[37] readxl\_1.3.1     modelr\_0.1.8      magrittr\_2.0.1 SnowballC\_0.7.0
[41] backports\_1.2.0 scales\_1.1.1      ellipsis\_0.3.1 rvest\_0.3.6
[45] assertthat\_0.2.1 colorspace\_2.0-0 stringi\_1.5.3 munsell\_0.5.0
[49] broom\_0.7.5      crayon\_1.3.4
```

Appendix C: Major Code Modules Used

C.1 data_collect_mentalhealth.R

```
### Reddit Data Collect
### Levi C. Nicklas
### 10/25/20
###
###
### Notes: This data collected will be from reddit.
###        Target data is that which related to mental
###        health and the open forum discussion of
###        the issues, stigmas, and other challenges
###        surrounding those experiencing mental health
###        issues.
###
###        The package {RedditExtractoR} will be used
###        to obtain the data. This will then be written
###        out to the data folder in an .RDS or .csv format.

library(tidyverse)
library(RedditExtractoR)



mental_health <- c("anxious", "anxiety",
                   "depressed", "depression",
                   "mental", "illness", "scared",
                   "afraid", "sad", "emotion", "anger",
                   "angry", "upset", "suicide", "abuse",
```

```
                "emotional","help", "addiction")

reddit_data <- list()

for(i in 1:length(mental_health)){
  reddit_temp <- RedditExtractoR::get_reddit(search_terms = mental_health[i],
                              #regex_filter = ,
                              subreddit = "mentalhealth",
                              cn_threshold = 10,
                              sort_by = "comments",
                              wait_time = 3)

  reddit_data[[i]] <- reddit_temp
}

write_rds(reddit_data, "Data/RawData/mentalhealth_reddit_20201025.RDS")

}

rm(reddit_thread_kernel)
```

C.2 correcting_vertex_labels.R

```
# convert_vertex_labels
# Levi C. Nicklas
# 1/20/2021 -- *Happy Inauguration Day*
#
#
# Notes: This function converts text labels of two graphs and maps them to
    integer values.
```

```
#       {graphkernels} only works with integer labels on verticies. The
    function finds
#       the union of unique words/labels and then maps each unique label to
    an integer
#       and then assigns the corresponding label to the correct vertex.

convert_vertex_labels <- function(g1, g2){
  # Libs Req.
  require(magrittr)
  require(dplyr)
  require(igraph)
  require(graphkernels)

  # Grab vertex labels (words).
  v.g1 <- unlist(get.vertex.attribute(g1))
  v.g2 <- unlist(get.vertex.attribute(g2))

  # Build a dataframe that maps words to integers (1 to 1).
  map_to_int <- seq(1,length(unique(c(v.g1,v.g2))))
  map_to_int <- data.frame(word = unique(unique(c(v.g1,v.g2))),
                           int = map_to_int)

  # Change g1 vertex names.
  vertex.attributes(g1)$name <- left_join(data.frame(word =
      vertex.attributes(g1)$name), map_to_int, by = "word") %>%
    select(int) %>% pull()

  # Change g2 vertex names.
  vertex.attributes(g2)$name <- left_join(data.frame(word =
      vertex.attributes(g2)$name), map_to_int, by = "word") %>%
    select(int) %>% pull()
```

```
  return(list(g1,g2))
}


### Worked Example -- not functionized ###
# g1 <- reddit_graphs_s[[1]][[1]]
# g2 <- reddit_graphs_s[[2]][[1]]
#
# v.g1 <- unlist(get.vertex.attribute(g1))
# v.g2 <- unlist(get.vertex.attribute(g2))
#
# unique(c(v.g1,v.g2))
#
# map_to_int <- seq(1,length(unique(c(v.g1,v.g2))))
#
# map_to_int <- data.frame(word = unique(unique(c(v.g1,v.g2))),
#                         int = map_to_int)
#
# vertex.attributes(g1)$name <- left_join(data.frame(word =
   vertex.attributes(g1)$name), map_to_int, by = "word") %>%
#   select(int) %>% pull()
#
# vertex.attributes(g2)$name <- left_join(data.frame(word =
   vertex.attributes(g2)$name), map_to_int, by = "word") %>%
#   select(int) %>% pull()
#
# CalculateVertexHistKernel(list(g1,g2))
# CalculateVertexHistKernel(list(reddit_graphs_s[[1]][[1]],
#                                reddit_graphs_s[[2]][[1]]))
```

C.3 convert_to_graphs.R

```r
### Reddit df -> skip_gram graph
### Levi C. Nicklas
### 12/30/20
###
###
### Notes:



convert_to_graphs <- function(){
  # Libraries
  library(here)
  library(furrr)

  plan(multisession, workers = 4)

  ### REDDIT POSTS ###

  post_thread_graphs <- list()

  source(here::here("Development/Scripts/df_to_graph_list.R"))

  text_graphs <- furrr::future_map(.x = post_thread_text_sample$text,
                                   .f = df_to_graph_list)
  return(text_graphs)

  ### NHTSA PAPERS ###
  # papers <- readRDS(here::here("Data/NHTSA/RawData/ConsolidatedPapers.rds"))
  #
  # # import function
```

```r
  # source(here::here("Development/Scripts/df_to_graph_list.R"))
  # text_graphs <- furrr::future_map(.x = papers$text,
  #                                  .f = df_to_graph_list)
  #
  # saveRDS(text_graphs, "Data/NHTSA/ProcessedData/nhtsa_graphs.RDS")
}
```

C.4 compute_kernel.R

```r
# Compute Graph Kernels
# Levi C. Nicklas
# Date: 1/4/2021
#
#
# Notes:
#



compute_kernel <- function(){
  # Libraries
  require(furrr)

  plan(multisession, workers = 8)

  text_kernel <- furrr::future_map(.x = thread_graphs,
                                   .f = compute_graph_similarity,
                                   .options = furrr_options(seed = T))

  #saveRDS(text_kernel, "Data/ProcessedData/reddit_graphkernel_325.RDS")
  return(text_kernel)
```

```r
  saveRDS(text_kernel, "Data/ProcessedData/redditthreads_graphkernel_all.RDS")
}
```

C.5 df_to_graph_list.R

```r
# Process into Graph Objects
# Levi C. Nicklas
# Date: 12/19/2020
#
#
# Notes: This script features a function which takes a text vector
#        and returns a list of graph representations of the text file.

df_to_graph_list <- function(text){
  # Check libraries
  require(dplyr)
  require(tidyr)
  require(tidytext)
  require(igraph)

  n_gram <- 2
  k_skip <- 2

  # Prepare variables and space.
  text_df <- as.data.frame(text)
  colnames(text_df) <- c("text")
  text_df$id <- seq(1,nrow(text_df))
  graph_list <- list()

  # Loop over rows in data.
```

```
for(i in 1:nrow(text_df)){
  # tokenize as skip grams.
  temp_graph <- tidytext::unnest_tokens(text_df,
                                        output = "words" ,
                                        input = text,
                                        token = "skip_ngrams",
                                        n = n_gram,
                                        k = k_skip) %>%
    # filter to only 1 document.
    dplyr::filter(id == i) %>%
    # split bigram
    tidyr::separate(words, c("word1", "word2"), sep = " ") %>%
    # remove stop words
    anti_join(stop_words, by = c("word1" = "word")) %>%
    anti_join(stop_words, by = c("word2" = "word")) %>%
    # toss NA values.
    tidyr::drop_na() %>%
    # clean out punctuation %>%
    mutate(word1_toss = stringr::str_detect(word1, "\\.")) %>%
    mutate(word2_toss = stringr::str_detect(word2, "\\.")) %>%
    mutate(toss_pair = word1_toss|word2_toss) %>%
    filter(toss_pair == F) %>%
    # clean out numebrs
    select(id,word1,word2) %>%
    mutate(word1_toss = stringr::str_detect(word1,"[:digit:]")) %>%
    mutate(word2_toss = stringr::str_detect(word2,"[:digit:]")) %>%
    mutate(toss_pair = word1_toss|word2_toss) %>%
    filter(toss_pair == F) %>%
    select(id,word1,word2) %>%
    # only keep words of > len 3
    mutate(word1_toss = (nchar(word1)<4)) %>%
```

```r
  mutate(word2_toss = (nchar(word2)<4)) %>%
  mutate(toss_pair = word1_toss|word2_toss) %>%
  filter(toss_pair == F) %>%
  select(id, word1, word2) %>%
  # group each bigram.
  group_by(word1, word2) %>%
  # count occurances.
  count() %>%
  # produce Graph.
  igraph::graph_from_data_frame()

# cleaned_words_df <- temp_graph %>% igraph::clusters()
# cleaned_words_df <- as.data.frame(cleaned_words_df$membership)
# cleaned_words_df$words <- rownames(cleaned_words_df)
# colnames(cleaned_words_df) <- c("member", "words")
# reduced_clusters <- cleaned_words_df %>%
#   filter(member != 1)
#
# # Get single largest cluster.
# big_graph <- temp_graph %>%
#   as_edgelist() %>%
#   as.data.frame() %>%
#   anti_join(y = reduced_clusters, by = c("V1" = "words")) %>%
#   anti_join(y = reduced_clusters, by = c("V2" = "words")) %>%
#   graph_from_data_frame()

# Store
#graph_list[[i]] <- list(big_graph, text)
graph_list[[i]] <- list(temp_graph, text)

# if(i %% 1000 == 0){
```

```
    #   print(paste0("On Row: ",i))
    # }
  }



  # Return a list of Graphs
  return(graph_list)

}
```

C.6 compute_graph_similarity.R

```
# Compute Graph Similarity
# Levi C. Nicklas
# Date: 1/4/2021
#
#
# Notes:
#



compute_graph_similarity <- function(input_graph){
  require(graphkernels)
  require(igraph)
  require(here)

  source(here::here("Development/Scripts/convert_vertex_labels.R"))

  # Edit Graph List to Op on.
```

```
#graph_list <- reddit_graphs_s
#graph_list <- nhtsa_graphs
graph_list <- thread_graphs

#Allocate Storage
result <- rep(0,length(graph_list))

#Compute Similarity btwn 1 graph and all others. FOR REDDIT
for(i in 1:length(graph_list)){
  if(length(igraph::vertex.attributes(graph_list[[i]][[1]][[1]])) > 0 &
     length(igraph::vertex.attributes(input_graph[[1]][[1]])) > 0 ){

    # Correct Labels Issue.
    tmp_graph_list <- convert_vertex_labels(graph_list[[i]][[1]][[1]],
        input_graph[[1]][[1]])
    #print(paste0("Calculating Graph: #",i))
    #K <- graphkernels::CalculateEdgeHistKernel(tmp_graph_list)
    K <- graphkernels::CalculateEdgeHistGaussKernel(tmp_graph_list,1200)
    similarity_value <- K[1,2]
    result[i] <- similarity_value
  } else {
    #print(paste0("Skipping Graph: #",i))
    result[i] <- NA
  }
}

# FOR NHTSA
# for(i in 1:length(graph_list)){
#   if(length(igraph::vertex.attributes(graph_list[[i]][[1]])) > 0 &
#      length(igraph::vertex.attributes(input_graph[[1]])) > 0 ){
#
```

```
  #     # Correct Labels Issue.
  #     tmp_graph_list <- convert_vertex_labels(graph_list[[i]][[1]],
    input_graph[[1]])
  #     #print(paste0("Calculating Graph: #",i))
  #     #K <- graphkernels::CalculateEdgeHistKernel(tmp_graph_list)
  #     K <- graphkernels::CalculateEdgeHistGaussKernel(tmp_graph_list,1200)
  #     similarity_value <- K[1,2]
  #     result[i] <- similarity_value
  #   } else {
  #     #print(paste0("Skipping Graph: #",i))
  #     result[i] <- NA
  #   }
  # }

  return(result)
}
```

Appendix D: NHTSA Metadata

Keep in margin
name it "Fatalities"

CASE #	VEHICLE MAKE /MODEL	LOCATION	FATALITIES_DUE_TO_CRASH	ROLLOVER
CA08034	05 FORD E-450	ANGOLA, DE	Yes	No
CA09005	98 FORD E-350	LEXINGTON CO., SC	Yes	No
CA09040	09 FORD F-450	BENNINGTON, VT	Yes	No
CA09075	04 FORD F-450	FORT BRAGG, NC	Yes	No
CA09076	98 FORD E-350	NASHVILLE, TN	Yes	No
CA09080	00 FORD E-350	JOHNSON CO., TN	Yes	No
CA09082	04 FREIGHTLINER 2500	MEMPHIS, TN	Yes	No
CA11004	07 GMC 3500 EXPRESS	BRIGHTON, NY	Yes	No
CA11026	01 FORD F-350	CLARKSBURG, WV	Yes	Yes
CA11027	09 CHEVROLET G4500	ORANGEBURG, SC	Yes	Yes
CA12002	12 FORD E-350	BUCKINGHAM, VA	Yes	Yes
CA12030	01 FORD E-350	MAPLETON DEPOT, PA	Yes	Yes
CA12032	09 CHEVROLET 4500	SARPY, NE	No	Yes
CA12034	05 FORD ECONOLINE	GLENCOE, KY	Yes	Yes
CR12001	07 FORD E-450	ALTOONA, NY	Yes	No
CR12002	11 FORD E-350	NEWARK, NJ	Yes	Yes
CR13004	99 FORD E-350	LEWIS CO., WV	Yes	No
CR13021	2012 Chevrolet G4500	OCILLA, GA	Yes	No
CR14002	2012 FORD E-350	MILFORD, MA	Yes	Yes
CR14003	10 CHEVROLET G4500	WINDSOR, ME	Yes	Yes
CR14057	11 FORD F-350 AMBULANCE	MANSFIELD, OH	Yes	Yes
CR14063	12 FORD E-350 TYPE II	MARION CO., WV	Yes	Yes
CR15019	08 FORD E-SERIES	EAST BRUNSWICK, NJ	Yes	Yes
CR17012	2013 Chevrolet Express 3500 Type II Ambulance	DUANESBURG, NY	Yes	No
CR18003	16 FORD F-450 TYPE I	STAFFORD CO., VA	Yes	Yes
IN10018	99 FORD E-350	ST. LOUIS COUNTY, MN	Yes	Yes
IN10032	03 FORD E-350	MCDOWELL CO., WV	Yes	Yes
IN10036	03 FORD E-350	CLARK CO., IL	Yes	Yes
IN11001	09 FORD E-350	CLARK CO., IN	Yes	No
IN11015	10 CHEVROLET G4500 EXPRESS	SHERIDAN, IN	Yes	Yes
IN11023	10 CHEVROLET G3500 EXPRESS	MILWAUKEE, WI	No	Yes
IN11024	96 FORD E-350	FAIR PLAY, MO	Yes	No
IN13021	90 FORD E-350	ELMWOOD, NE	Yes	No
IN13025	09 FORD E-450	WAUPUN, WI	Yes	Yes
IN14004	12 FORD E-450	BUFFALO, MN	Yes	Yes
IN14012	07 CHEVROLET EXPRESS 3500	CARLSBAD, TX	Yes	Yes
IN14035	10 FORD ECONOLINE E-350	SAN ANTONIO, TX	Yes	No
IN16013	03 FORD ECONOLINE E-350	MARSHALL, MO	Yes	No
IN16025	11 FORD ECONOLINE E-350	CLYDE, OH	Yes	Yes
DS09001	05 FORD E-350	PALO ALTO, CA	Yes	Yes
DS09019	99 FORD E-350	TUCSON, AZ	Yes	Yes
DS11018	07 FORD E-350	ONTARIO, CA	Yes	No
DS13021	00 FORD F-350	CAMBRIDGE, ID	No	No
DS14016	13 MERCEDES-BENZ SPRINTER	SIGNAL HILL, CA	No	Yes
DS15009	09 FORD E-350	PASADENA, CA	Yes	No
DS15014	07 GMC YUKON	SPRINGFIELD, CO	Yes	Yes
DS16010	04 FORD E-350 TYPE II	MORGAN CO., CO	Yes	Yes
DS16014	09 FORD E-350 TYPE II	ALBUQUERQUE, NM	No	Yes

Appendix E: KDE Clustering

As an alternative to popular clustering methods that were used in the studies in this paper, another clustering method was attempted that features use of kernel density estimation. By using a kernel density estimation (KDE) on the kernel values, we can cluster the documents into similar groups, defined by local maxima.

First, the graph kernel matrix, K is taken, and we extract a row, i, and we compute a KDE using R's `density()` function. Now, the default value for bandwidth will likely produce a smooth, unimodal or bimodal distribution, but this is not what the goal is. The goal is to use the KDE to find clusters through their value appearing in a local maxima. So, through producing a KDE with few local maxima, we produce very few clusters. If the number of clusters needs to increase, we can essentially overfit the KDE and manipulation the use of the bandwidth parameter to create a KDE with many more local maxima and minima.

Once a KDE with a sufficient number of local maxima, which is determined by the user, then cluster breaks are located. If we consider the estimated KDE to be a function $k(x)$, where x is a kernel value, and $k(x)$ is the estimated density at a value x, then we can use calculus to locate the break points.

This method need additional development that was beyond the scope of this paper, and will be revisited at a later time. Code developed is available in the GitHub (`https://github.com/Levi-Nicklas/GraphDocNLP`).

References

Akioyamen, P., Nicklas, L. C., & Sanchez-Arias, R. (2020). A framework for intelligent navigation using latent dirichlet allocation on reddit posts about opiates. In *Proceedings of the 2020 the 4th international conference on compute and data analysis* (pp. 190–196).

Bengtsson, H. (2020). A unifying framework for parallel and distributed processing in r using futures. *arXiv preprint arXiv:2008.00553*.

Cheng, W., Greaves, C., & Warren, M. (2006). From n-gram to skipgram to concgram. *International journal of corpus linguistics*, *11*(4), 411–433.

Csardi, G., Nepusz, T., et al. (2006). The igraph software package for complex network research. *InterJournal, complex systems*, *1695*(5), 1–9.

Csardi, M. G. (2013). Package 'igraph'. *Last accessed*, *3*(09), 2013.

Kondor, R. I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning* (Vol. 2002, pp. 315–22).

Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, *5*(1), 1–42.

Nikolentzos, G., Meladianos, P., Rousseau, F., Stavrakas, Y., & Vazirgiannis, M. (2017). Shortest-path graph kernels for document similarity. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 1890–1900).

Rivera, I., & Rivera, M. I. (2015). *Package 'redditextractor'.*

Rosenfeld, N., Szanto, A., & Parkes, D. C. (2020). A kernel of truth: Determining rumor veracity on twitter by diffusion pattern alone. In *Proceedings of the web conference 2020* (pp. 1018–1028).

Silge, J., & Robinson, D. (2016). tidytext: Text mining and analysis using tidy data principles in r. *Journal of Open Source Software*, *1*(3), 37.

Sugiyama, M., & Borgwardt, K. (2015). Halting in random walk kernels. *Advances in* *neural information processing systems*, *28*, 1639–1647.

Sugiyama, M., Ghisu, M. E., Llinares-López, F., & Borgwardt, K. (2018). graphkernels: R and python packages for graph comparison. *Bioinformatics*, *34*(3), 530–532.

Vazirgiannis, M., Malliaros, F. D., & Nikolentzos, G. (2018). Graphrep: boosting text mining, nlp and information retrieval with graphs. In *Proceedings of the 27th acm international conference on information and knowledge management* (pp. 2295–2296).

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., & Borgwardt, K. M. (2010). Graph kernels. *The Journal of Machine Learning Research*, *11*, 1201–1242.

Wickham, H. (2010). Stringr: modern, consistent string processing. *The R Journal*, *2*(2), 38–40.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., ... others (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686.