Kristian Kanck

TexTraClus

A Spatio-Textual Sub-Trajectory Clustering Framework

Masteroppgave i Informatikk
Veileder: Kjetil Nørvåg

Oktober 2020

Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Kristian Kanck

TexTraClus

A Spatio-Textual Sub-Trajectory Clustering Framework

Masteroppgave i Informatikk
Veileder: Kjetil Nørvåg
Oktober 2020

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Problem Description

Original problem description

A spatio-textual trajectory is a trajectory with some text associated with it. As an example, consider a tweet with retweets from different locations, the locations of the (re-)tweets together with the tweet text is in this case a spatio-textual trajectory. Given a database of spatio-textual trajectories, various data mining tasks can be performed in order to find interesting patterns. A simple example is clustering, which will give tweets that have high textual similarity, and which are retweeted from similar locations. The aim of this project is to 1) study previous work related to this topic, 2) identify interesting and useful data mining operations to be performed on the spatio-textual trajectories, 3) develop algorithms/indexes for efficient execution of one or more of these operations, and 4) evaluate these on a large dataset.

Supervisor: Kjetil Nørvåg

Abstract

The amount of spatial and textual data generated by modern Big Data platforms is growing rapidly. Understanding frequent patterns and identifying new trends becomes more important for marketing, politics and social cohabitation. Challenges are increasing for existing Data Mining methods when attempting to extract knowledge from large spatio-textual data sets, and combining existing methods is increasingly important to keep up with the current rate of technological evolution. This thesis is supported by well established academic methods across disciplines within Big Data Mining to efficiently identify spatio-textual trajectory trends in large data sets.

Sammendrag

Mengden romlig og tekstuell data generert av Big Data-platformer vokser stadig raskere. Å forstå bruksmønster og identifisere nye trender blir viktigere for både markedsføring, politikk og sosialt samhold. Utfordringene øker for eksisterende datautvinningsmetoder når en skal ekstrahere kunnskap fra store rom-tekstlige datasett, og å kombinere eksisterende metoder blir viktig for å effektivt holde tritt med teknologisk utvikling. Dette arbeidet støtter seg på etablerte akademiske metoder på tvers av fagområder innen Big Data Mining for å effektivt identifisere rom-tekstlige bane-trender i store datamengder.

Acknowledgements

I would like first and foremost to thank my supervisor professor Kjetil Nørvåg for supporting me and offering help in completing this master's thesis. I also would like to thank Anette Siverstøl for all her love and support.

Table of Contents

List of Figures

List of Algorithms

List of Tables

1 Introduction

John Naisbitt famously wrote "We are drowning in information but starved for knowledge." in the 1982 book *Megtrends: Ten New Directions Transforming Our Lives* [1]. This statement still holds true today. Smart phones capable of GPS tracking have become household items, which allows software and hardware to record the device's geo-location at unprecedented accuracy and rate.

Geo-tracking moving objects has been an important aspect of data mining in relations to predicting and identifying common animal migration patterns for purposes such as hunting and livestocking, or even predicting hurricane movement patterns. The commercialization of phones and laptops carrying GPS-trackers affords ordinary people without business or research needs to track their daily movement. Application and device designers are then able to collect this data to extract interesting behavioral patterns, such as vehicular movement through traffic or the movement of tourists across points of interest.

Most modern humans carry a GPS-tracking device daily and as their movements are tracked —so are their interactions with the devices themselves. Smartphones and laptops can be used to interact with friends through social media or to publically publish their thoughts on social networks as they interact with the world around them. Where the previous generation of data mining tasks focused heavily on how geo-tracked devices navigated the world around them spatially, the modern big data collecting dichotomy allows researchers to focus on multiple dimensions of interactions.

In the current age of social networks, applications are capable of tracking spatio-textual interactions and efficiently storing them in data warehouses. Such data can be used to analyse the opinions, thoughts and social trends among people. Services such as the Yelp application allow users to publish detailed restaurant reviews for other users' benefit. Social network applications such as Twitter allow users to publically publish their thoughts for users across the world to interact with. Extracting meaning from these vast collections of data requires specialized Data Mining methods. Identifying misinformation campaigns in social media has become a topic of much interest the last decade. Understanding how political language spreads geographically across social sub-networks is an important field of research for effectively combating such online campaigns.

Applying existing methods in the Data Mining field, and effectively developing new methods of locating spatio-textual behavioral patterns are some of the emerging Megatrends of the 21st century. The goal of this paper is to combine spatio-textual clustering methods on social network user trajectories to effectively locate trends in large data sets, where they otherwise would be drowned in noise to the human eye.

1.1 Problem Definition

In traditional spatial trajectory clustering, only spatial distances between data points are considered. In spatio-textual trajectory clustering we must also perform clustering based on the textual dimension. We are interested in finding efficient ways to perform clustering on trajectories in both the spatial and the textual dimension using existing methods.

Research questions:

- **Research Question 1** What approaches exists in context of clustering trajectories?
- **Research Question 2** How can these be extended to spatio-textual trajectories?
- **Research Question 3** How does a spatio-textual trajectory clustering algorithm perform on a large dataset?

1.2 Thesis Overview

- **Chapter 2** Gives a brief introduction to the field of data mining and the particular research areas relevant to this thesis.
- **Chapter 3** Explores related research and work in the field of spatio-textual trajectory data mining and provides an overview of the research.
- **Chapter 4** Is a brief introduction to the field of textual analysis, and discusses which methods are well suited in relation to spatial clustering methods while also exploring methods of optimization available.
- **Chapter 5** Explores the data clustering mining technique and discusses which methods of clustering might best suit the task of spatio-textual trajectory clustering.
- **Chapter 6** Will describe the methods used to formulate an algorithm based on the methods previously examined to cluster spatio-textual trajectories.
- **Chapter 7** Presents the results from applying the algorithm onto two large spatio-textual datasets.
- **Chapter 8** Evaluates the results from the algorithm and compares the datasets' results.
- **Chapter 9** Will conclude the findindings of this thesis.

2 Data Mining

In a 1999 article [2], Feyaad described Knowledge Discovery in Databases as "the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". In 2016 Witten defined data mining as the automated process of discovering patterns in large databases, where useful patterns discovered allow the user to make "non-trivial predictions on new data" [3]. Data mining can be summarized as the process of extracting information or knowledge in the absence of a testable hypothesis.

The need to automate the process can be instrumental in the knowledge discovery aspect of data mining. If the data mining process is approached with a bias, then it harms the chances of obtaining new knowledge from the data. Witten also describes the harmful impact of bias when utilizing data mining techniques to discover new knowledge from data. For data mining to discover knowledge in an unbiased fashion, good data mining techniques and algorithms must be utilized.

Anand highlighted the importance of domain knowledge when applying data mining techniques [4]. He acknowledged that the ideal circumstances for knowledge discovery through data mining techniques is when the discovery process is not biased by the user. They outlined the role of the human in terms of domain knowledge and bias information. The paper states about completely unbiased discovery systems that "such a discovery system is not a viable option as the amount of knowledge that can be discovered from a dataset far exceeds the amount of data".

There exists a balance between useful and harmful bias when utilizing data mining techniques. The mining operations must be focused enough that the amount of data returned from the knowledge discovery process is not overwhelming, but still contain some bias so that the knowledge discovered can be useful for future decisions. The user's domain knowledge is the important factor in distinguishing between these two outcomes.

2.1 Mining Operations

Data mining operations and techniques can be invaluable for businesses to learn new trends about their customers for reasons such as expanding their product range, changing marketing strategies or for making informed decisions based on customer history through decision trees.

Data mining techniques can be divided into two categories: supervised, and unsupervised. An example of an supervised mining method is classification. In classification, data entries are ran through some algorithm and assigned to pre-defined output labels. A set of labels are defined by the user and data entries will be allocated a label based on their attributes. For a database containing spatio-temporal data on humidity, cloud density, wind, temperature and sun position one could predict for the user if the weather will be sunny, rainy, windy or cloudy.

A weakness with this type of data mining operation is that the user must already know how to categorize these types of weather based on the input parameters. If other weather types exist, but do not exist as labels in the algorithm, the algorithm will not be able to classify them correctly. Classification algorithms can be invaluable for businesses such as credit card companies for predicting potential customers which are most likely to not be able to default on their payments in the future. However, classification methods are not necessarily well suited for locating new trends in existing data.

Unsupervised data mining techniques are more suited towards identifying unknown trends in data sets. Assosciation based data mining methods, such as the Apriori algorithm [5] are designed to predict likely combinations of items based on assosciation rules. A useful example on how to use the apriori algorithm is to analyze shopping transactions and identifying items that often appear together. By looking at the assosciation relationships between different types of wares, a grocery store could then have its layout arranged to both make future shopping trips more efficient for customers and entice shoppers to buy other closely assosciated wares based on what they originally set out to buy. Figure 2.1 displays the split data mining paradigm and some of the supervised and unsupervised methods.

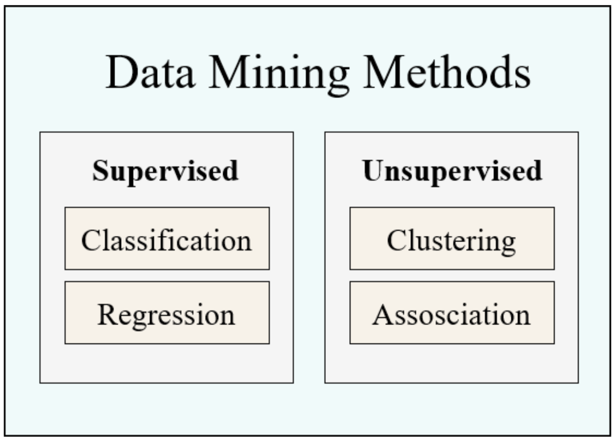


Figure 2.1: Data Mining methods

The focus of this thesis will be the unsupervised clustering data mining method. The goal of clustering methods is to group items based on some distance or similarity. The clustering method is described as unsupervised because there are no expectations on how the data will be grouped, as opposed to classification where the data will be assigned to some label. Most commonly, clustering is performed in a spatial dimension and measures iteratively how close each item is to other items in the data set. Then, items can be grouped based on measures such as density or by simply partitioning the data set. This grouping can be used to identify patterns such as locating traffic congestion for urban planning projects, identifying areas with high crime rates for re-organizing police patrols or grouping similar movies based on user input metrics for a recommendation algorithm. In 2018 a paper was published where the researchers utilized unsupervised clustering methods to separate ”fake news” articles from other news articles [6] by analyzing the structure of the language used.

Clustering is a versatile technique that can be applied to many different domains. While the spatial domain is the most obvious, it can also be done textually. The important principle is to convert the difference between items into some measurable distance. Teqchniques such as the Jaccard Similarity Coefficient or Cosine Similarity can be used to measure the difference between two textual items through set theory or by evaluating the items as vectors, respectively. Measuring differences between text documents as distances is a widely used technique utilized daily by web search engines. In a web search context, text documents can be clustered based on similarity to make the storage and retrieval of documents with high degrees of similarity more efficient. Spatial and textual mining operations can be combined to cluster based on both spatial distance and textual similarity. This was demonstrated in a 2017 paper suggesting a spatio-textual variant of the popular k-means clustering method [7]. This paper extended a euclidian distance function to also take into account the Jaccard Similarity Coefficient in its distance function. This allowed the researchers to cluster points not only by spatial distance, but also textual similarity. Textual mining operations and techniques relevant to spatio-textual clustering will be covered in further detail in Chapter 4.

2.1.1 Trajectory data mining methods

An emerging trend in the data mining field is evaluating and analyzing trajectory data. Trajectory data mining could be utilized by analyzing trajectory data to recommend trajectories or predict trajectories. The 2007 sub-trajectory clustering algorithm TraClus [8] used a variation of the DBSCAN clustering algorithm [9] to find common paths among trajectory data sets and evaluated the algorithm on animal migration and hurricane movement trajectory data to predict common sub-paths.

Analyzing trajectories can have large impacts on business strategies. A 2013 study on human movement derived from mobile GPS data [10] analyzed the trajectories constructed from the movement to find common paths. Utilizing such techniques can have large impacts on business strategy, because of the added temporal dimension. Trajectory data mining techniques differ from non-temporal spatial data mining techniques because they take into account the histories of each object. Just like in assosciation mining techniques, the relationships between each spatial element is evaluated. The earlier example of using clustering techniques to locate traffic congestion can be solved in a much more efficient way if the method also takes into account where the vehicle came from and where it is headed. Trajectory mining techniques can allow businesses or bodies of government to obtain a higher degree of knowledge about behavioral patterns and better structure systems and services to serve the needs of citizens or their users in a more efficient manner, potentially reducing costs and improving the users' satisfaction.

3 Related Work

This chapter will present an overview of different papers related to the field of spatio-textual trajectory clustering. Articles describing approaches to solve the current problem will be explored as well as other related research.

3.1 Methodology

A preliminary review of the literature was performed before setting out to explore how to best fulfill the research questions defined for this thesis. Articles reviewed were gathered through online research literature aggregators. As suggested by Oates [11], a set of concepts were first defined then used as search words in the online aggregators for the review. The concepts defined were ”spatio-textual clustering” and ”trajectory clustering”. A broad collection of documents were skimmed through for relevance. Afterwards they were carefully processed to explore how they relate to the field of spatio-textual clustering and identify the current state of the art.

3.2 Review

Data mining on spatio-textual trajectories is an emerging field within data mining. The research is sparse, but much work has been done researching data mining operations on spatio-textual data and trajectories separately. An early paper in the trajectory clustering field was the 1999 study by Gaffney [12], which suggested utilizing linear regression as a means to cluster whole trajectories. In 2007 Lee, Han and Wang published a paper suggesting the TraClus algorithm [8]. TraClus is a density based sub-trajectory clustering algorithm that groups similar line segments found within trajectories and generates representative trajectories highlighting the dominant trend within each grouping. Previously, the field had been primarily focused on clustering whole trajectories, whereas the TraClus algorithm was designed to cluster common sub-trajectories to identify interesting patterns which otherwise could have been overlooked in whole trajectory clustering. The algorithm has since become the standard in density based sub-trajectory clustering and the focus of researchers has since largely shifted away from trajectory clustering in favor of sub-trajectory clustering.

Inspired by the TraClus algorithm, a team of researchers developed the MoveMine system [13] for detecting periodicity and swarming patterns in sub-trajectories. MoveMine utilizes periodic pattern, swarm patterns and movement interactions to classify frequently appearing sub-trajectories. Unlike TraClus, which performs line segment clustering, MoveMine locates frequently occurring check-in points with high similarity across trajectories and generates a representative trajectory based on these points. This system was later revisited by Wu in 2014 and published as MoveMine 2.0 [14] with an extended focus on pairwise relationship patterns, namely attraction/avoidance relationships and follower patterns.

In 2017 a chasm in the research had been identified by Choi and Chung [7] who published a paper extending the k-Means partion based clustering algorithm to process spatio-textual datasets. The researchers cited a rising need to analyze social media trends more closely as a motivation for the research. The researchers cited other research in the field of spatio-textual query processing as an inspiration, but noted in the paper that the research on applying clustering algorithms to spatio-textual data was sparse. The spatio-textual k-Means algorithm utilized the Jaccard Similarity Coefficient [15] to compare the textual distance between two nodes and required the user to define a weight $\alpha \in [0, 1]$ to inversely proportion textual similarity and euclidian distance when calculating the distance between two nodes.

Nguyen and Shin published in 2017 a paper suggesting the DBSTexC algorithm [16] which adopts the density based DBSCAN clustering algorithm to evaluate textual content in tandem with spatial clustering and evaluated it on geospatio-textual Twitter messages. The algorithm utilized geographical points of interest (POI) mentioned in the textual body of the twitter message and evaluates whether messages are POI-relevant or irrelevant based on the textual content and the geospatial publishing location of the message. This spatio-textual implementation differs from the 2017 spatio-textual k-Means implementation in that the distance function of the DBSCAN algorithm is not modified. The similarity or dissimilarity of the textual contents are not measured as distances and evaluated against the geospatial distances between points. Instead, the researchers redefined the definition of how core points are identified in the DBSCAN algorithm.

To identify which twitter messages are relevant to any particular geospatial point of interest, DBSTexC introduces a threshold measure for when a point is POI-relevant. This threshold is manually set by the user through a variable α. To make this algorithm work, there must already exist some mapping from the POI-names and the geographical locations they are assosciated with. The researchers solved this by performing a keyword search of semantic variations of the POI-name through the database.

A hierarchical variation of the TraClus sub-trajectory clustering algorithm was proposed in 2018 [17]. The researchers stated that "density-based clustering algorithms such as DBSCAN and DENCLUE cannot find hierarchical clusters and they fail to detect clusters of different densities since they use a global density threshold". The researchers utilized HDBSCAN, a hierarchical variation of the DBSCAN algorithm proposed in 2013 [18], which incrementally clusters at different densities through single-linkage producing a hierarchical tree of clusters. This method was combined with TraClus to cluster trajectories using trajectory sequences and semantic information to produce context sensitive "reference spots". These reference spots could then be ordered hierarchically to show periodic patterns occurring in the data.

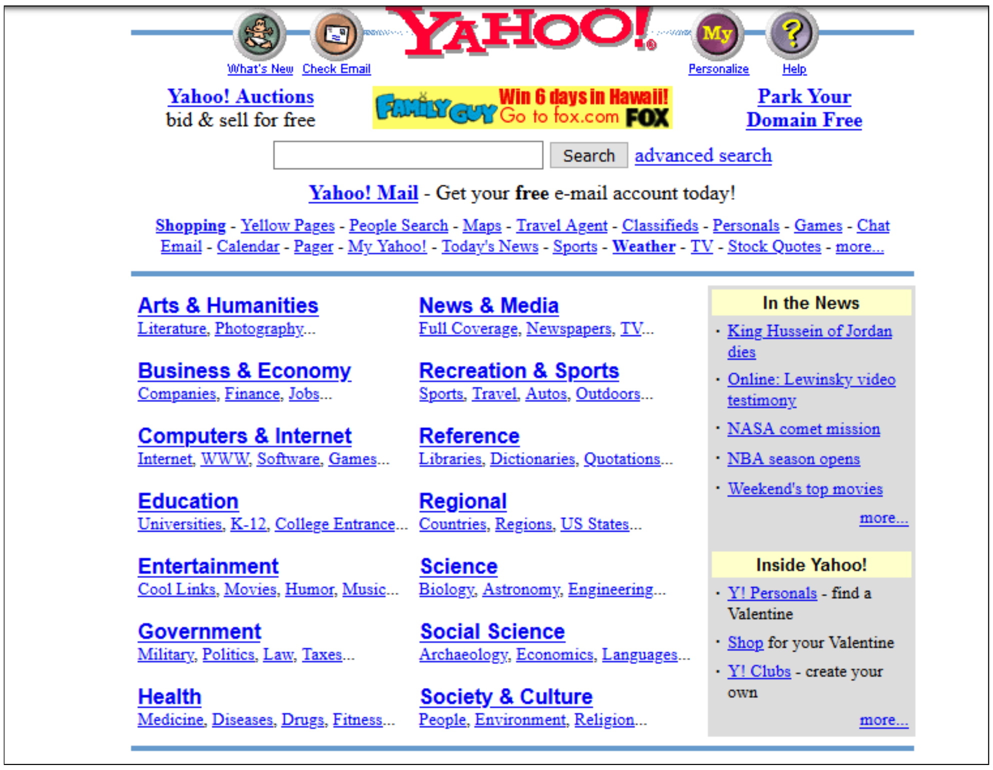
Although some research has been made on spatio-textual clustering and trajectory clustering, the field of spatio-textual trajectory clustering remains a relatively unexplored domain. The goal of this thesis will be combine results from previous research and contribute to future research by demonstrating how current spatio-textual clustering techniques can be extended to existing trajectory clustering methods when applied to social network data.

4 Textual Mining Operations

In this chapter we will discuss text operations that are useful in relation to a spatio-textual clustering algorithm where textual documents are compared in pairs. Starting with basic text comparison algorithms, their strengths and weaknesses and where and when they are best applied. Lastly useful textual pre-processing techniques will be presented.

4.1 Introduction

The field of textual analysis is a well researched field. The process of automating the retrieval of information from textual documents has long been a field of interest for researchers and enterprises alike. The Google search engine was presented in a 1998 paper [19] by Brin and Page who referenced the need to automate the process of retrieving web documents as a substitute to manually curated web indices such as the Yahoo! web portal. Figure 4.1 demonstrates how the Yahoo! web portal manually selected indices for web browsers in 1999.[1] The cumbersome and expensive process of maintaining such indices and the burden of understanding exactly which topics should be curated were identified as problem areas. By automating this process the situation would improve not only for the business, but also the user. The user would be served a wider range of topics in web documents, and the cost of maintaining such indices would be reduced.



Yahoo! Auctions
bid & sell for free

Park Your Domain Free

advanced search

Yahoo! Mail - Get your **free** e-mail account today!

Shopping - Yellow Pages - People Search - Maps - Travel Agent - Classifieds - Personals - Games - Chat
Email - Calendar - Pager - My Yahoo! - Today's News - Sports - **Weather** - TV - Stock Quotes - more...

Arts & Humanities
Literature, Photography...

Business & Economy
Companies, Finance, Jobs...

Computers & Internet
Internet, WWW, Software, Games...

Education
Universities, K-12, College Entrance...

Entertainment
Cool Links, Movies, Humor, Music...

Government
Military, Politics, Law, Taxes...

Health
Medicine, Diseases, Drugs, Fitness...

News & Media
Full Coverage, Newspapers, TV...

Recreation & Sports
Sports, Travel, Autos, Outdoors...

Reference
Libraries, Dictionaries, Quotations...

Regional
Countries, Regions, US States...

Science
Biology, Astronomy, Engineering...

Social Science
Archaeology, Economics, Languages...

Society & Culture
People, Environment, Religion...

In the News
- King Hussein of Jordan dies
- Online: Lewinsky video testimony
- NASA comet mission
- NBA season opens
- Weekend's top movies

more...

Inside Yahoo!
- Y! Personals - find a Valentine
- Shop for your Valentine
- Y! Clubs - create your own

more...

Figure 4.1: Yahoo! web portal snapshot from 8th February 1999

[1] Yahoo! web portal accessed through internet archive service Wayback Machine: `https://web.archive.org/web/19990208021547/https://www.yahoo.com/` (accessed 29th Oct 2021)

At the core of solving these issues is the domain of textual analysis. To retrieve a document for a user based on a search term, keyword or category, some technique must be able to distinguish what makes some documents relevant and other documents less relevant. The process of comparing documents for relevancy can be semantic or agnostic towards semantics.

When the textual analysis is defined as semantic, the meaning and context of the words in the documents are important to measuring similarity. Homonyms can lead to false positives if the similarity measurement does not consider semantic meanings. An example can be how the word crane describes very different things for bird watchers or construction workers. A difficult problem when approaching semantic textual similarity is in the context of slang, or in poetry where words can be metaphors or contain hidden pop-culture references. If semantic meaning is important to the textual analysis process, the context of the words used is critical for the data mining process.

In this paper, the textual data mining operations used will not focus on semantic textual data mining, where typically text is classified to extract intent or meaning. Rather, the goal will be to discover trends in the language used by locating frequently used keywords in tandem with spatial clustering methods to find sub-trajectory patterns where certain keywords are more frequent than others.

4.2 Textual Similarity

Measuring textual similarity has many use cases, from matching search words to documents in online document databases, to measuring differences between documents such as when attempting to detect plagiarism. The methods used to detect such similarities are numerous and versatile, each with their own strengths and weaknesses.

In the context of algorithms designed to measure textual similarity, a textual node is referred to a document containing a set of words. Equation 1 demonstrates how a textual document D in a collection of n documents contains a set of words w .

$$D_n = \{w_1, w_2, \ldots, w_{len_{D_n}}\} \tag{1}$$

One of the most common textual similarity measures in information retrieval is the term frequency and inverse document frequency, also known as TF-IDF [20]. The ideal use case for TF-IDF is in document search engines, where short length search phrases are compared against a collection of documents that are much larger in length. Term frequency is generated by counting the frequency of each term in a document. The idea is that high frequency terms are important for describing key topics unique for that document. The inverse document frequency is used to measure words that are frequent across all documents in the collection. These words typically occur often in almost every document and offer little descriptive value to the contents therein. While the TF-IDF measure is a foundation in the field of information retrieval, extending this method to computationally intensive processes such as data clustering will lead to significant overheads in each computation. This method also offers less value when the documents compared are symmetrical in sizes.

For this thesis the textual comparisons will typically be between textual sentences that are relatively symmetrical in length. The Cosine Similarity and the Jaccard Similarity

Coefficient are two well established similarity measures in the academic field of textual analysis that perform well in such cases.

4.2.1 Consine Similarity

Cosine similarity [21] treats documents as separate vectors and measures the difference between them as an angular degree. The wider the angle between two documents, the higher degree of dissimilarity. Likewise, the narrower the angle, the higher degree of similarity between them. Equation 2 demonstrates the general formula for calculating the degree difference between two documents.

$$Sim(S_a, S_b) = cos\theta = \frac{\overrightarrow{S_a} \cdot \overrightarrow{S_b}}{||S_a|| \cdot ||S_b||} \tag{2}$$

To convert documents into vectors, a vector object must be constructed that represents all words that can possibly occur across all documents in the collection. The words represented in the vector object structure is the same for all documents, but for each document the frequency of the words are counted in its vector. For large collections of documents the vector object will tend to be very sparsely populated as any word that is not present in the document will be numerically represented as 0. Equation 3 demonstrates how a document vector object $\overrightarrow{D_n}$ in an n-sized collection represents the in-document frequency of any possible word w for the m unique words existing across all documents in the collection.

$$\overrightarrow{D_n} = w_1, w_2, w_3, \ldots, w_m \tag{3}$$

The similarity is calculated through the dot product of two vectors over the product of each vectors euclidian norm. For a vector $\overrightarrow{v} = \{w_1, w_2, w_3, \ldots, w_m\}$, the euclidian norm is calculated as $||\overrightarrow{v}|| = \sqrt{w_1^2 + w_2^2 + w_3^2 + \ldots + w_m^2}$. The dot product is calculated by multiplying the frequency of each unique word and summing them so that for two vectors $\overrightarrow{v_i}$ and $\overrightarrow{v_j}$ the dot product for the two will be equal to $w_{i1} \cdot w_{j1} + w_{i2} \cdot w_{j2}, \ldots, +w_{im} \cdot w_{jm}$.

Cosine similarity is well suited to tasks such as detecting plagiarism as the term frequency have a very high impact on the resulting measurement. A major benefit is also how missing fields have little to no impact on the resulting measurement. However, to effectively compute the similarity between items, large data object structures must be constructed and stored to memory, much of which contain no information at all. For a large collection of short documents with large variance in textual content, each document will be assigned extremely long vector objects containing mostly null fields. For commercial off-the-shelf systems, this can impact processing times tremenduously depending on available memory.

4.2.2 Jaccard Similarity Coefficient

The Jaccard Similarity Coefficient [15] treats documents in a collection as sets and utilizes the intersection and union set operations to calculate the similarity between any two documents. A stark contrast to the Cosine Similarity is that each element is counted only once, as term frequency is ignored. As a consequence Jaccard is rarely used to produce a ranking of documents in relation to a search term, but rather is better suited to quick comparisons between the contents of two documents.

Figure 4.2 demonstrates how two documents, each describing terms relevant to a grocery store and a pet shop share an intersection of terms.

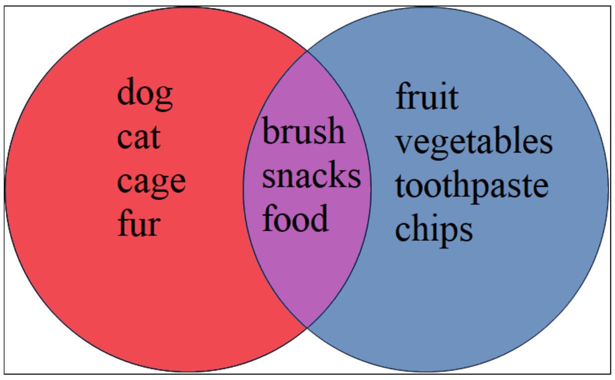


Figure 4.2: Intersection of documents describing different domains

The similarity between two sets of words is calculated as demonstrated in Equation 4. The example illustrated in Figure 4.2 would be calculated by dividing the intersection of the two sets over the union of the two sets. The collection contains 11 words in total with 3 words intersecting, so the jaccard similarity coefficient betwen these two documents in this specific example would be calculated as $\frac{3}{11} = 0.27$.

$$J(S_a, S_b) = \frac{|S_a \cap S_b|}{|S_a \cup S_b|} \tag{4}$$

The Jaccard Similarity Coefficient is well suited to cases where duplication does not matter. Jaccard is also slightly less computationally intensive than Cosine Similarity, as it performs less arithmetic operation for each comparison as well as requiring less storage in lieu of not requiring specialized data objects to represent the data.

Jaccard performs better when comparisons are made between symmetrical items. For two documents compared, where all the words from a smaller document D_1 are present in a much larger document D_2, the similarity measure will not be very high. This is because the Jaccard Similarity Coefficient, in contrast to Cosine Similarity, consders negative matches in its evaluation. This attribute is important to keep in mind when deciding which textual similarity function to apply to a data mining algorithm.

A feature both methods share is that they are not capable of distinguishing lexical or semantic similarity between any documents. Matches are semantically agnostic and only evaluate whether words appear or don't appear. This can be remedied by utilizing semantic dictionaries where terms are assosciated with other terms of similar meaning, or by using techniques such as n-grams for measuring textual similarity.

4.3 Text Preprocessing

Textual preprocessing is a monumentally important task when analysing natural language. Language is an evolving form of communication that is hard to decipher for computers. There exists many textual processing techniques available, such as lexical analysis, stemming, stopword elimination or text enrichment.

Lexical analysis is the process of translating documents into machine processable terms that can be measured in similarity.

Stemming refers to identifying the stem of each term so that they can be easily compared. In this process word suffixes and prefixes are usually stripped to reveal a shared word "stem" with identical semantic meaning.

Stopword removal refers to removing common words that have little functionality in describing the contents of a single document. These are typically found in overabundance across most documents and textual similarity measures containing these will usually yield a high degree of false positives.

Text enrichment refers to identifying the semantic meaning of words in context and either separating them or linking words together as synonyms. An example is how the noun "contest" describes something very different when used as a verb. Text enrichment will typically attempt to identify the context of a word's usage and either match them to words holding similar semantic meaning or negatively impact the similarity measurement between such words.

While some textual pre-processing techniques are optional depending on the task objective, the most important technique in its implementation is lexical analysis. Some techniques such as stemming, stopword removal and text enrichment can be very expensive based on quality demands of the preprocessing required. These techniques can also be harmful to the textual analysis if performed poorly.

4.3.1 Lexical Analysis

Lexical analysis refers to the process of tokenizing documents for computational processing and facilitating for the textual analysis performed. The first and most important operation in lexical analysis to execute, for any algorithm that measures textual similarity, is tokenization. Tokenization is the process of splitting the document body into tokens that can be measured for similarity. In the vast majority of cases, this means splitting the document body into tokens that end and start whenever any whitespace character, such as space or line break, occurrs. This allows the program to evaluate the similarity and dissimilarity of words in pairs or as document vectors.

Capitalization is the an important factor to consider, as most programming languages evaluate letters in similar cases as being different words. For an algorithm evaluating the similarity of the words "Vehicle" and "vehicle" it is important to understand whether the casing of the word is taken into account by the String evaluation. Most programming languages will evaluate these two words as being dissimilar. Lower-casing is typically not a computationally costly operation and will typically have large impacts in the results from textual analysis.

Another important aspect of lexical analysis is language. For an application or algorithm that measures the similarity of two documents, it is important to understand which languages are being utilized. In a textual clustering context, comparing documents containing different languages will lead to very poor results. This is because most word comparisons will result in negative matches, even if their semantic meanings are identical. For textual analysis algorithms it should be considered whether to only evaluate documents containing one singular language or whether to split up the document collection so that documents of different languages can be evaluated separately.

Depending on the context of the textual analysis, it is sometimes important to prune words from special characters. In internet slang and online social networks text can sometimes be dressed up in special characters to add meaning to words. In some cases, some pruning can be skipped depending on the objectives of the textual analysis. However, noise removal should be executed. Consider the term tokens "however" and "however,". If only tokenization is executed without noise removal, special characters such as punctuation marks and commas will not be removed from the text. If choosing to forego noise removal these terms will be regarded as two different terms in textual similarity operations.

4.3.2 Stopword Elimination

Stopword elimination is the process of removing stop words from documents before processing to speed up the textual analysis process and increase the accuracy of positive matches. Most natural language documents will contain vast amounts of words that contain very little discriptive meaning. Examples of stopwords are words such as prepositions like "over", "on", "in" or pronouns such as "I", "you", "it". In considering a Cosine Similarity case as an example, removing such words will result in much smaller vector data objects representing the documents and lead to a lower degree of false positive matches between documents. For the Jaccard Similarity Coefficient stop words will contribute to documents similarity measures also resulting in a large amount of false positives. An option when performing stopword removal is to also remove terms that occur extremely rarely. This can contribute to filtering out misspelled words, but caution should be taken so that relevant terms are not filtered out as that will harm the textual analysis.

Stop words can be identified statistically or through textual dictionaries. Statistically identifying stop words can be done by measuring the term frequency across the document collection for each word and removing words that occur higher than a set threshold. A weakness of this method is that it is hard for humans to manually determine exactly where the threshold should be set. Additionally, it is important to consider how closely related domain-wise the documents are for the statistics. As an example, if the statistics are based on a collection of documents all describing a single contextual domain such as cooking, important terms to the document retrieval might be filtered out if they occur too often.

It is possible to construct dictionaries containing terms that have a high degree of irrelevance regardless of the context they are used in. Prepositions and pronouns are examples of such words. These can be constructed manually, automatically or through a combination of both. Constructing such dictionaries manually is a time consuming and expensive process, but automating the process of building such dictionaries will run into the same problems when filtering stopwords statistically. The more efficient solution in order to achieve better results is to use both methods in tandem to exclude false positives. Such dictionaries should also be built from large amounts of document collections so that the dictionary isn't affected by a domain bias.

5 Clustering Operations

This section will introduce well known clustering method paradigms, discuss their strengths and weaknesses and identify algorithms that are well suited for clustering spatio-textual trajectories.

Clustering algorithms are used to identify or classify unrelated objects into object groups in order to locate unknown trends and patterns in the data. The book *Data Mining: Concepts and Techniques* [21] classifies the currently known clustering methods five categories: partition based, hierarchy based, density based, grid based and model based. From reviewing the literature, density based, hierarchy based and partition based methods are the most frequently used techniques when clustering trajectory data.

5.1 Partition Based Clustering

Partition based clustering methods are the simplest clustering methods. As the name suggests, the entire data set will be split up into partitions. These methods typically requires the user to identify the desired number of clusters before processing the data. A manually set parameter is needed, most often denoted as k where $k \leq n$ for n data-points. The result is that the algorithm then will assign each node in the data to one of the k clusters. The most famous example of the partition based clustering method is the algorithm k-means [22].

In the k-means algorithm the locations of k data points will be selected as centroids c and the distance to each other data point will be measured and grouped based on the shortest point-centroid distance. The centroid will then be shifted to the new group center and each point will be progressively evaluated again until an iteration is reached where no centroid will move. The resulting centroids c_k and their points respectively will form the clusters C_k. Algorithm 5.1 will demonstrate the general outline of the k-means algorithm.

A weakness of the k-partition clustering method is that every single data point will be categorized into some cluster. For some data sets and goals this is undesirable, as data points that are extreme outliers will also be included in the resulting clusters. Such outliers are usually referred to as noise. One solution towards reducing noise in partition-based clustering is to use the density of the points as a filtering mechanism, where points in a group that are further away than a distance ε.

Algorithm 5.1: k-means algorithm

```
Input       : $P = \{p_1, \dots, p_n\}$, points to be clustered
Input       : $k$, number of clusters
Output      : $K$, set of clusters
Parameters: $reachedEnd$, boolean initialized as false

1  choose $k$ initial centroids $C = \{c_1, \dots, c_k\}$;
2  foreach $p_i \in P$ do
3      $l(p_i) \longleftarrow minDistance(p_i, c_j), j \in \{1, \dots, k\}$;
4  end
5  repeat
6      foreach $p_i \in P$ do
7          $minDist \longleftarrow minDistance(p_i, c_j), j \in \{1, \dots, k\}$;
8          if $minDist \neq l(p_i)$ then
9              $l(p_i) \longleftarrow minDist$
10         end
11     end
12     if $UpdateCentroidsAndUpdateClusters(C, P) = false$ then
13         $reachedEnd \longleftarrow true$;
14     end
15 until $reachedEnd = true$;
```

An unfortunate consequence of k-partitioning methods is that the clustering will be tend towards circular shapes, which might not necessarily accurately represent the inherent patterns in the data as illustrated in figure 5.1.

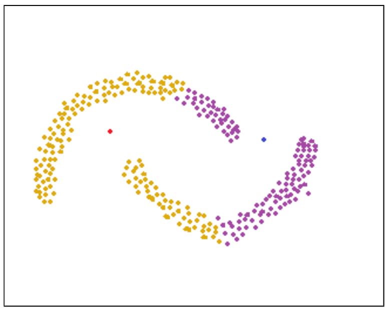


Figure 5.1: Partition based clustering effect on data with inherent pattern

In k-partitioning clustering methods such as k-means, the selection of the initial centroid points will heavily affect the resulting grouping of points. Efforts have been made by researchers to augment and enhance the process of selecting the initial centroids [23], but this incurs an extra computational overhead to an otherwise extremely efficient clustering method. Despite some inherent weaknesses, k-partitioning clustering methods remains incredibly efficient for quickly sorting data into dissimilar categories and algorithms such as the k-means clustering method has stood the test of time in its ease of implementation as well as computational efficiency.

5.2 Density Based Clustering

Hierarchical and partition based clustering algorithms have a tendency towards discovering spherical-shaped clusters. The main strategy of density based clustering methods is to find clusters of arbritrary shapes by clustering regions of high density separated by regions of low density in order to locate natural clusters occurring in the data. Density based clustering is one of the most popular clustering methods in data mining for knowledge discovery due to this property as the process of identifying dense areas implicit in the data set minimizes human bias.

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [9] implements density based clustering by introducing three states for data point. A point can either be a core component, an edge or noise in relation to each cluster. It is not necessary to explicitly specify the desired amounts of resulting clusters, like in partition based methods. Instead two parameters are required: the minimum neighbouring distance ε for two points to be clustered, and the minimum amounts of points needed $minPts$ in a group for it to be clustered.
Density based clustering algorithms are generally much more computationally intensive

Algorithm 5.2: DBSCAN algorithm.

```
Input   : $P = \{p_1, \dots, p_n\}$, points to be clustered
Input   : $\varepsilon$, radius
Input   : $minPts$, density threshold
Output  : $C = \{c_1, \dots, c_{k<n}\}$, clusters to be formed

foreach $p_i \in P$ do
    $p_i.visited \longleftarrow true$;
    $NeighbourPts \longleftarrow$ FindNeighboursWithinEpsilon($p_i, eps$);
    if $NeighbourPts < minPts$ then
        $p_i.label \longleftarrow Noise$;
    end
    else
        intialize new cluster $c_m$;
        add $p_j$ to $c_m$;
        foreach $p_j \in NeighbourPts$ do
            if $p_j.visisted = false$ then
                $p_j.visited \longleftarrow true$;
                $NeighbourPts' \longleftarrow$ FindNeighboursWithinEpsilon($p_j, eps$);
                if $NeighbourPts' \geq minPts$ then
                    $NeighbourPts \longleftarrow NeighbourPts \cup NeighbourPts'$;
                end
            end
            if $p_j \notin$ any other cluster $c$ then
                add $p_j$ to cluster $c_m$;
            end
        end
    end
end
```

than partitioning and hierarchical clustering methods, but are exceptional when the goal of the clustering process is to identify hidden trends in the data and filter out noisy data that would otherwise be expensive to manually extract and exclude.

A 2015 paper noted the slow computational process of the DBSCAN algorithm [24], particularly when applied to multiple dimensions and difficulty of choosing appropriate parameters. The researchers also expressed a need in the density based clustering field for new efficient algorithms to take its place.

A 2017 paper was published as a rebuttal [25], highlighting techniques for selecting good paramaters and demonstrating the permanency and industry relevance of the DBSCAN algorithm despite its age.

5.3 Hierarchy Based Clustering

Hierarchical clustering methods differ slightly from density and partition based methods as they're often implemented using some tree data structure. Using tree structures the clustering process will retain the order of cluster grouping, making it possible to evaluate trends at a micro-level closer to the leaf-nodes, while also at a macro level closer to the root node. An advantage of hierarchical clustering methods is their efficiency. By utilizing efficient tree-structures such as a B+-tree, it is possible to iterate efficiently through datasets and generate clusters while reducing large amounts unnecessary comparisons of data points.

A popular hierarchical clustering method is the BIRCH algorithm [26] developed in 1996, which incrementally constructs a hierarchical CF (Clustering Feature)-tree to cluster data points in multiple phases. The CF-tree is typically a B+-tree where each division in the tree will contain the CF attribute to describe the underlying data points. The clustering feature consists of three values, the number of points in each subset, the linear sum of each dimension in the subset and the square sum of each dimension in the subset. These attributes are used as statistics to efficiently evaluate the points within each subset of the tree without having to manually iterate through each of the points contained.

The algorithm will calculate for each leaf node the centroid, radius and diameter of the cluster. The centroid is the center point of all the data points in the cluster while the radius is the average distance from any point in the cluster to the centroid. The diameter is the average point-to-point distance in the cluster, and this measure is used to determine when a leaf node should be split up.

The biggest strength of the BIRCH algorithm is its multi-layer clustering strategy. The higher levels of the hierarchy allows flexibility in terms of clustering algorithm used, while the lower levels of the hierarchy reduces the complexity and increases scalability of the algorithm. One weakness of the BIRCH algorithm is that its clusters have a tendency towards spherical shapes due to the radius and diameter measuring of the points in each subcluster. And while a desired amount of k-clusters aren't defined like in partitioning based clustering approaches, fixed leaf node sizes might lead to unexpected pairings of data points forming clusters.

Algorithm 5.3 demonstrates the rough pseudocode outline of the BIRCH algorithm. Important to note is the simplicity and flexibility of implementation, allowing it to easily be combined with other clustering methods, or altered to fit unique application needs.

Algorithm 5.3: BIRCH algorithm

```
Input  : Data points $P = \{p1, \dots, p_n\}$
Input  : Branching factor $B$ for maximum amount of children per node
Input  : Maximum diameter $D$ of sub-clusters in leaf nodes
Output : Set of clusters $C = \{c_1, \dots, c_m\}$

1  foreach $p_i \in P$ do
2      Determine correct leaf node for $p_i$ insertion
3      if Diameter constraint not violated by $p_i$ insertion then
4          Add $p_i$ to cluster and update CF triplets
5      else
6          if room to insert $p_i$ then
7              Insert $p_i$ as single cluster and update CF triplets
8          else
9              Split leaf node, and possibly parents, and redistribute CF triplets
10         end
11     end
12 end
```

5.4 Trajectory Clustering

A trajectory T can be defined as a spatial collection of n points p with a temporal ordering of timestamps t so that a trajectory $T = \{p_1^{t_k}, \ldots, p_n^{t_m}\}$, where $k < m$. The goal of clustering trajectories is to find common attributes between different trajectories and extracting commonalities.

An important distinction between trajectory and single point-data clustering is the temporal dependency between the points in the trajectory. There is always a single direction between any two points in the trajectory due to the temporal hierarchy. Depending on the desired results of a trajectory clustering algorithm the vector direction of a line segment should be taken into account. As an example, if a trajectory clustering algorithm attempts to predict hurricane movement patterns, it is crucial that the vector direction of trajectories is taken into account in order to avoid producing misleading results. Figure 5.2 illustrates how algorithms might exclude or include seemingly similar trajectories in clusters based on direction angle. In this illustration the green trajectories are clustered together, while the red trajectory is excluded from the cluster.

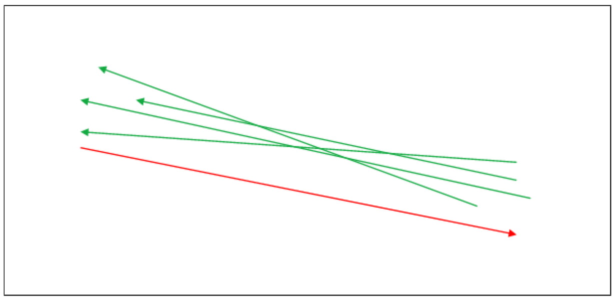


Figure 5.2: Exclusion of trajectories

Another important distinction, as pointed out in the 1999 paper on using linear regression to cluster trajectories [12], is that different trajectories in a dataset will often have different lengths. The paper states that "one cannot simply convert the trajectories to fixed-length vectors and apply a clustering technique such as the k-means algorithm in a fixed-dimensional space".

Despite this, Yang and Hu [27] published in 2006 a paper suggesting a trajectory clustering algorithm called TrajPattern. The algorithm is a partition-based clustering method of pattern-matching trajectories and was intended to be used to mine trajectory clusters based on mobile gps data signals. Due to limitations in geolocation technology at the time, the algorithm assumes location by inference rather than exact location tracking from the mobile devices. The paper explains that "[the] energy in a mobile device is very limited, so it is impossible for a mobile object to continuously send out its location information".

By assuming mobile devices will announce their locations at set intervals, the algorithm affords a partition based clustering approach. The algorithm requires the user to specify a set amount of desired clusters k, much like the k-means algorithm. The algorithm will then mine k patterns with the most "normalized match", where the normalized match refers to how the algorithm groups subtrajectories into similar-length trajectories before comparing them. Yang and Hu also admits in the paper that the algorithm responds poorly to noise, as is typical for partition based clustering approaches.

Whole trajectory clustering has remained a difficult problem for researchers, especially as such automated methods has a tendency to hide apparent common trends at a micro-level in the trajectories due to a focus of evaluating the trajectories at a macro-level.

5.4.1 Subtrajectory Clustering

Sub-trajectory clustering as a field in data mining emerged as a response to existing trajectory clustering algorithms that either ignored or performed poorly at identifying interesting sub-trends. The 2007 paper *TraClus: A Partition-and-Group Framework* [8], suggested a sub-trajectory clustering algorithm based on existing density clustering principles. The paper states that at the time, the most similar algorithm to the TraClus algorithm was Gaffney's trajectory regression clustering algorithm [12].

The TraClus algorithm utilizes three phases to extract and cluster sub-trajectories. First the trajectories are partitioned into sub-trajectories. Then the algorithm uses a distance based clustering algorithm to group the subtrajectories based on their distance in terms of the perpendicular and parallell distance of trajectories as well as their angle. The original TraClus adopts the the DBSCAN algorithm for clustering sub-trajectories. Lastly, it groups the previously partitioned sub-trajectories based on similarity. In addition to clustering sub-trajectories the TraClus algorithm also produces representative trajectories for each cluster. In this section the pseudocode for the TraClus algorithm will be presented as it was in the original TraClus paper [8] Algorithm 5.4 will describe the execution of the algorithm at a higher level, while algorithms 5.5, 5.6 and 5.7 will describe how each phase is performed at a more detailed level.

Algorithm 5.4: TraClus algorithm

```
Input  : A set of trajectories $T = \{TR_1, \dots, TR_n\}$
Output: A set of clusters $C = \{c_1, \dots, c_m\}$
Output: A set of representative trajectories $\mathcal{R}$

1 foreach $TR_i \in T$ do
2     Execute Algorithm 5.5: Approximate Trajectory Partitioning as $\mathcal{L}$;
3     Accumulate $\mathcal{L}$ into a set $\mathcal{C}$;
4 end
5 Execute Algorithm 5.6: Line Segment Clustering
6 foreach $C \in O$ do
7     Execute Algorithm 5.7: Representative Trajectory Generation as $\mathcal{R}$
8 end
```

To identify common sub-trajectories the TraClus algorithm evaluates each trajectory as a series of line segments L_i between two points p_n with a timestamp t_k ordering such that any line segment can be expressed as $L_i = (p_{i1}, t_{i1}), (p_{i2}, t_{i2})$. Figure 5.3 illustrates how any line segment is composed of two spatio-temporal data points.

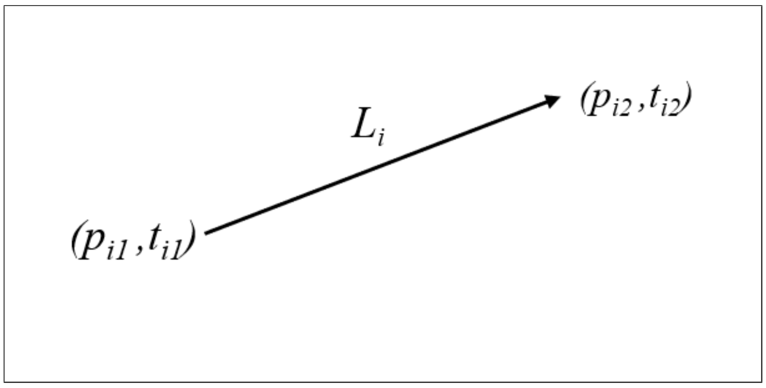


Figure 5.3: A line segment L_i with end points $(p_{i1}, t_{i1}), (p_{i2}, t_{i2})$

The TraClus algorithm measures the spatial distances between line segments in three components. Line segment perpendicular distance, parallel distance and angle distance. The TraClus paper suggests calculating the perpendicular distance $d_\perp$ between two line segments L_i and L_2 as demonstrated in equation 5. Figure 5.4 demonstrates how the TraClus paper suggests identifying the spatial coordinates used in calculating the perpendicular distance by using projection points p_s and p_e.

$$d_\perp(L_i, L_j) = \frac{l_{\perp 1} + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}} \tag{5}$$

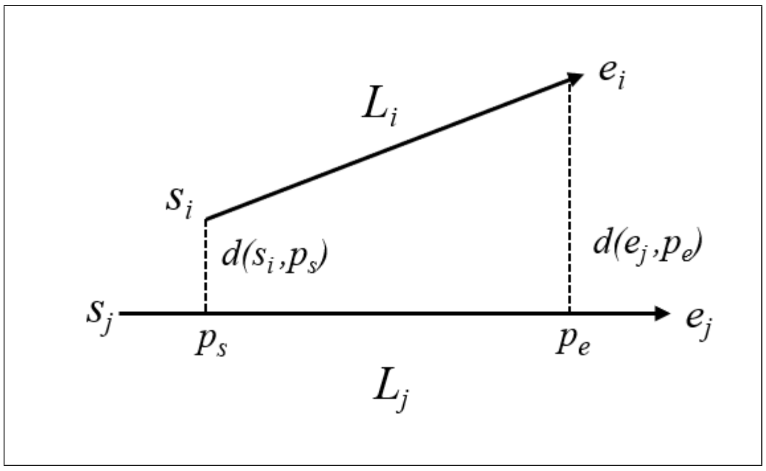


Figure 5.4: Perpendicular distance between line segments L_i and L_j in TraClus algorithm

Measuring the parallel distance between two line segments is done by utilising the same projection points p_s and p_e to generate two smaller line segments $l_{||1}$ and $l_{||2}$ and selecting the smaller of the two as demonstrated in equation 6.

$$d_{||}(L_i, L_j) = MIN(l_{||1}, l_{||2}) \tag{6}$$

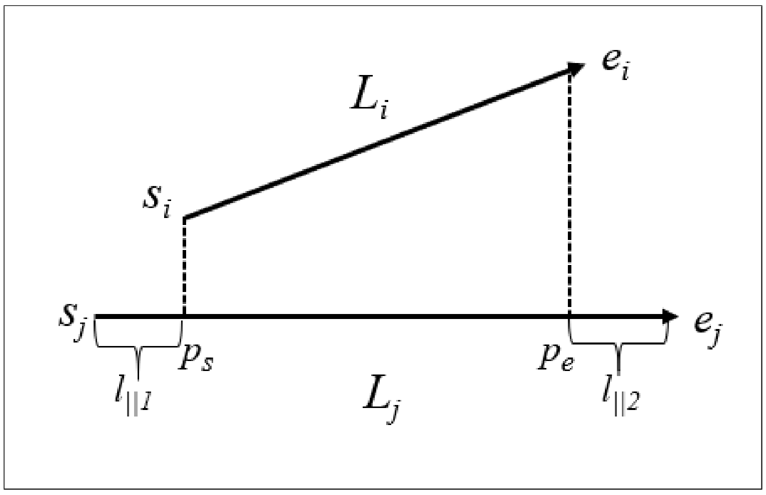


Figure 5.5: A line segment L_i with end points $(p_{i1}, t_{i1}), (p_{i2}, t_{i2})$

The angular distance between two line segments L_i and L_j is defined in equation 7. The TraClus paper defines the angle distance where " $||L_j||$ is the length of L_j and θ ($0° \leq \theta \leq 180°$) is the smaller intersecting angle between L_i and L_j.

$$d_\theta(L_i, L_j) = \begin{cases} ||L_j|| \cdot sin(\theta), & 0° \leq \theta \leq 90° \\ ||L_j||, & 90° \leq \theta \leq 180° \end{cases} \tag{7}$$

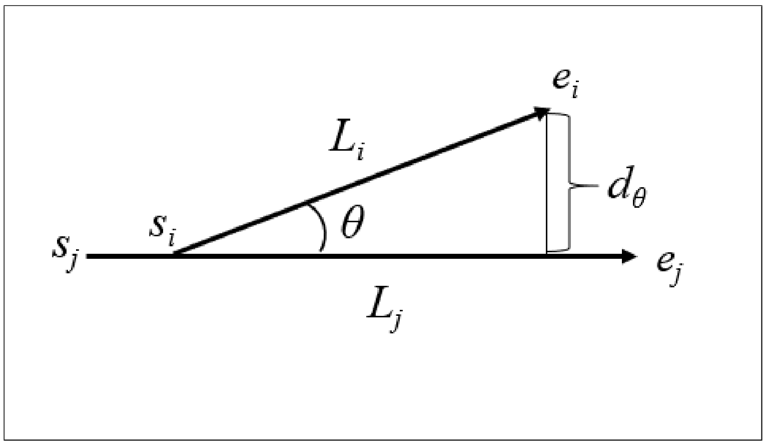


Figure 5.6: A line segment L_i with end points $(p_{i1}, t_{i1}), (p_{i2}, t_{i2})$

TraClus performs trajectory partitioning as described in Algorithm 5.5 by identifying characteristic points p_c in each trajectory T. The characteristic points indicate rapid changes on the edges within a specific trajectory. At these characteristic points, the trajectory is partitioned so that any line segments before and after a given characteristic point are split up into separate trajectory partitions. The paper states that "the optimal partitioning of a trajectory should possess two desirable properties: *preciseness* and *conciseness*". The paper goes on to explain that "preciseness means that the difference between a trajectory and a set of its partitions should be as small as possible, while conciseness means that the number of trajectory partitions should be as small as possible".

These two principles are conflicting, and finding a good balance between two is a challenging endeavour. The TraClus algorithm solves this using the minimum descriptive length (MDL) principle [28]. The algorithm defines the MDL cost as the sum of $L(H)$, which describes the sum of the length of all trajectory partitions, and $L(D|H)$, the sum of the difference between a trajectory and a set of its trajectory partitions. The goal is to iteratively partition trajectories by characteristic points to minimize $L(H) + L(D|H)$ for any given trajectory. This is a very computationally costly process, as every line segment subset must be considered for each trajectory.

TraClus implements this through an approximation by incrementally evaluating the MDL cost of adding more line segments to a candidate partition. If the MDL cost increases by adding another line segment, the previous point is selected as a characteristic point and all line segments preceding that point are selected as a trajectory partition.

Algorithm 5.5: Approximate Trajectory Partitioning

```
Input  : A trajectory $TR_i = p_1, .., p_{len_i}$
Output: A set $CP_i$ of characteristic points

1  Add starting point $p_i$ to the set $CP_i$
2  $startIndex \longleftarrow 1$;
3  $length \longleftarrow 1$;
4  repeat
5      $currentIndex \longleftarrow startIndex + length$;
6      $cost_{par} \longleftarrow MDL_{par}(p_{startIndex}, p_{currentIndex})$;
7      $cost_{nopar} \longleftarrow MDL_{nopar}(p_{startIndex}, p_{currentIndex})$;
8      if $cost_{par} > cost_{nopar}$ then
9          Add $p_{currentIndex-1}$ into $CP_i$;
10         $startIndex \longleftarrow currentIndex - 1$;
11         $length \longleftarrow 1$;
12     else
13         $length \longleftarrow length + 1$;
14     end
15 until $startIndex + length > len_i$;
16 Add $p_{len_i}$ into $CP_i$;
```

Line segment clustering in the TraClus algorithm draws inspiration from the DBSCAN algorithm by identifying dense clusters of trajectory partitions. The algorithm begins by classifying all line segments as "unclassified". Then, the algorithm measures from a line segment whether another line segment is reachable within a user defined distance of ε. Just as in the DBSCAN algorithm, if a user defined number of *minLines* linesegments are reachable, a cluster is formed. This process continues until all line segments are assigned to a cluster or classified as noise.

Algorithm 5.6: Line Segment Clustering

```
Input  : A set of line segments $\mathcal{D} = \{L_1, \ldots, L_{num}\}$
Input  : Two parameters $\varepsilon$ and $MinLns$
Output: A set of clusters $\mathcal{O} = \{C_1, \ldots, C_{num}\}$

Set $clusterId = 0$; // initialization
Mark all line segments in $\mathcal{D}$ as $unclassified$
foreach $L \in \mathcal{D}$ do
    if $L = unclassified$ then
        Compute $N_\varepsilon(L)$;
        if $|N_\varepsilon(L)| \geq MinLns$ then
            Assign $clusterId$ to $\forall X \in N_\varepsilon(L)$
            Insert $N_\varepsilon(L) - \{L\}$ into the queue $\mathcal{Q}$;
            ExpandCluster($\mathcal{Q}, clusterId, \varepsilon, MinLns$);
            Increase $clusterId$ by 1;
        else
            Mark $L$ as $noise$
        end
    end
end
Allocate $\forall L \in \mathcal{D}$ to its cluster $C_{clusterId}$;
foreach $C \in O$ do
    if $|PTR(C)| < MinLns$ then
        Remove $C$ from the set $O$ of clusters;
    end
end

ExpandCluster ($\mathcal{Q}, clusterId, \varepsilon, MinLns$){
    while $\mathcal{Q} \neq \varnothing$ do
        Let $M$ be the first line segment in $\mathcal{Q}$;
        Compute $N_\varepsilon(M)$
        if $|N_\varepsilon(M)| \geq MinLns$ then
            foreach $X \in N_\varepsilon(M)$ do
                if $X = unclassified \lor noise$ then
                    Assign $clusterId$ to $X$;
                end
                if $X = unclassified$ then
                    Insert $X$ into the queue $\mathcal{Q}$;
                end
            end
        end
        Remove $M$ from the queue $\mathcal{Q}$;
    end
}
```

The final step of the TraClus algorithm is representative trajectory generation. This generates a trajectory that attempts to describe the overall movement in each cluster. The representative trajectory is constructed as a sequence of points $RT_i = \{p_1, \ldots, p_{i_{len}}\}$ for $1 \leq i \leq clustersize$. The coordinates of the points in the representative trajectory is generated by sweeping a vertical line across the longest diameter of the cluster. Each starting or ending point met along the sweep is counted. If the final count is equal to or greater than the *minLines* variable, the average coordinates of those line segment points in respect to the major axis will be inserted into the representative trajectory. A smoothing parameter is used to ensure that points are not too close in proximity.

Algorithm 5.7: Representative Trajectory Generation

```
Input  : A cluster $C_i$ of line segments
Input  : MinLns
Input  : A smoothing parameter $\gamma$
Output: A representative trajectory $RT_j$ for $C_i$

1  Compute the average direction vector $\overrightarrow{\mathcal{V}}$;
2  Let $\mathcal{P}$ be the set of the starting and ending points of the line segments in $C_i$;
3  //$X'$-value denotes coordinate of the $X'$ axis Sort the points in the set $\mathcal{P}$ by their
   $X'$-values;
4  foreach $p \in \mathcal{P}$ do
5      //Count $num_p$ using a sweep line;
6      Let $num_p$ be the number of line segments that contain the $X'$-value of the
        point $p$;
7      if $num_p \geq MinLns$ then
8          $diff$ := the difference in $X'$-values between $p$ and its immediately previous
            point;
9          if $diff \geq \gamma$ then
10             Compute the average coordinate $avg'_p$;
11             Undo the rotation and get the point $avg_p$;
12             Append $avg_p$ to the end of $RT_i$
13         end
14     end
15 end
```

6 Methodology

This section will cover the methods used to develop an algorithm to process TRACLUS-clustering on spatio-textual data. To begin with the system environment used during development and evaluation will be described. Afterwards the thesis will go in depth to describe the dataset used during evaluation and offer optimalizations to make processing more efficient both processing- and memorywise. Finally the chapter will explain how the TraClus algorithm was modified to handle clustering spatio-textual data for both textual and spatial dimensions using longitude and latitude.

6.1 Environment

The environment for evaluating the use of a modified TraClus algorithm on internet based trajectories was a commercial desktop computer. Evaluation of the modified TraClus algorithm required two steps: pre-processing and execution of the modified TraClus algorithm itself. For the pre-processing the memory was the limiting factor, while for the algorithm itself the processor was the limiting factor. Due to the limitations in hardware, both the yelp-dataset and the twitter-dataset had to be pre-processed to bring run-times down.

Capacity	Speed
16 GB	2133 MHz

Table 6.1: Memory specifications

Model	Clock speed	Cores	Threads
Intel Core i5-6500	3.2 GHz	4	4

Table 6.2: Processor specifications

6.2 Datasets

In this this thesis we will develop and evaluate a spatio-textual trajectory clustering algorithm based on the TraClus algorithm using the Jaccard Similarity Coefficient. Two datasets will be analyzed and their results compared. The first dataset is based on status update replies from the Twitter-application and the second dataset is is based on restaurant reviews from the Yelp-application.

6.2.1 Twitter Dataset

The Twitter dataset used to evaluate the algorithm was a snapshot of spatio-textual twitter status update replies in NoSQL JSON format as presented in Appendix A. To understand the dataset, one must not only first understand the Twitter API, but also how the Twitter application has evolved over its existance.

The Twitter service was originally designed to utilize the SMS protocol [29] which the official Twitter documentation describes as having, at the time, a 160 character limit. As such, any Twitter status update could contain no more than that amount of characters.

With 20 characters reserved for commands and username mentions, the actual length of the text portion of a message was 140 characters. This was later extended to 280 characters as Twitter evolved from relying on the SMS protocol into a web service. Modern Twitter messages are not bound to containing text alone and have the option to display multimedia content such as images or videos. The method of using Twitter also evolved as smart-phones emerged. By utilizing the GPS trackers found in modern laptop computers and smart phones, Twitter messages could be geo-tagged at the time of creation. This could be done either as a precise geographical location in the form of a single point represented in longitude and latitude or as having been created within a geospatial area represented as a polygonal bounding box with arbitrarily numbered vertices.

Twitter can be viewed as a social-network, where users interact with each other through publically available messages. These messages will appear on the user's profile as what the official Twitter documentation refers to a "status update" [30]. A user may block specific logged-in users from viewing their status updates or allow their status updates to be available only to a list of approved users. It is possible to create a status update that is a response to another status update as well as possible for a twitter user to re-broadcast another status update onto their user profile without adding textual or multimedia content of their own. The latter action is known as a "re-tweet" while the former is officially referred to as a reply. A third Twitter status reply type is the "quoted status", which is a form of reply which contains the content of the original message the quote is replying to as well as the reply.

There are several ways to exctract spatio-textual trajectories from a dataset based on twitter status updates. The most intuitive way would be to construct trajectories based on status updates per user. However, most status update locations are represented by a bounding box that span large geographical areas. If trajectories were to be constructed this way, most trajectories would appear spatially stationary as most status updates per user appear within one geospatial bounding box. To construct interesting trajectories in the context of clustering we could utilize the geo-tagging of retweets, where the original status update's text could be represented as travelling from location to location based on the timestamp of each re-tweet. However, in this case the text would remain static and have little value in a spatio-textual clustering context. Status update replies generally contain textual data and are a better candidate for constructing spatio-textual trajectories. Using replies, we can construct trajectories by considering each reply a spatio-textual data point ordered temporally by timestamps and group them by the original status update.

The Twitter data object model has two ways of representing location. The "geo" data field and the "place" data field. The "geo" data field is represented by a single geospatial point with one latitude coordinate and one longitude coordinate. The "place" data field consists of a list of geospatial points that form a bounding box, and states that the twitter message was published from a device somewhere inside the bounding box. A status update can contain both of these data fields, either one exclusively or neither. For status updates containing only the "place" data field the location will be derived from calculating the average latitude and longitude values of all the points in the bounding box as demonstrated in equation 8.

$$loc(lat, lon) = \frac{\sum_{n=1}^{loc\_num} loc(lat_n, lon_n)}{loc\_num} \tag{8}$$

The Twitter dataset used for evaluating spatio-textual TraClus is an arbitrarily selected subset of the Twitter live enterprise database. A consequence of this is that there is no guarantee that a status update and all its replies are necessarily contained within. The original status update for a set of replies can't be guaranteed to be present in the dataset. Likewise, there is no guarantee that every reply among all replies to any single status update is contained either. A fair compromise is to infer trajectories by grouping any reply by the status update they respond to. For this thesis, spatio-textual trajectories will be inferred by ordering status update replies chronologically by the "timestamp" field and grouping them by the data field "in_reply_to_status_id" from the Twitter data object API as demonstrated in Appendix A.

6.2.2 Yelp Dataset

The Yelp based dataset used to evaluate the algorithm was a 6 GB snapshot of Yelp review data in NOSQL json format. Yelp is a service that allows users to submit reviews of restaurants users have visited, where users are to submit a rating from 1 to 5 and a written textual review. Yelp allows users to download a subset of its data in NoSQL JSON-format for academic purposes.[2] The datasets of interest for this thesis are the two datasets containing business-data as described in Appendix C, which specifies the geospatial locations of each business, and the dataset containing review-data from each user as described in Appendix B. The business dataset contains geospatial locations, while the review-dataset contains the textual reviews themselves, a user-id and a timestamp. Both datasets contain the business id, allowing for joining of the two datasets so that spatio-textual trajectories can be formed based on the user-ids and timestamps.

Constructing spatio-textual trajectories from a Yelp dataset is a much easier task than in the Twitter case. A trajectory can be constructed for each user where the location of each point in the trajectory is based on the geolocation of the business each review corresponds to. The ordering of the points can be based on the temporal timestamps that denote the time of each review was published to the database. It is important to note that each review is not necessarily submitted to the yelp database in the same order that the business was visited. A user might visit a number of businesses in a given time period and submit the reviews in batch at a later date. Thus, the trajectories of the users might not necessarily accurately represent the movement of the user.

In stark contrast to the Twitter-data set where 280 characters are the absolute maximum character count per textual status update, the yelp dataset allows a much larger character limit for each review. This is likely to encourage users to write as detailed and accurate reviews as possible. The higher character count in text strings per review requires much larger memory capacity from the computer, and will potentially become a bottleneck during both pre-processing and during text similarity comparisons in the spatio-textual TraClus algorithm, depending on the size of the subset used.

[2] Publically available Yelp datasets: `https://www.yelp.com/dataset` (Accessed: 01 December 2020)

6.2.3 Pre-processing

Due to the sheer size of the datasets some processing was helpful before the algorithm could be applied.

The original geospatial twitter dataset amounts to a total of 162 MB while the yelp review and business datasets totalled 6.1 GB and 150 MB respectively. Appendix A displays all possible data fields possible in the Twitter dataset, while appendix C and B displays all possible data fields present in the Yelp datasets. To prioritize evaluating the spatio-textual TraClus algorithm on interesting data, only trajectories consisting of four data points or more were considered. While not all fields are present in every Twitter data row, each row still consists of large amounts of unnecessary data.

Useful trajectories do not inherently exist in the Twitter data object. Like previously mentioned, most consecutive status updates from unique twitter users never move spatially as a consequence of a majority of status updates originating from large geospatial bounding boxes, and users rarely, if ever, publish status updates outside a given bounding box. The trajectories exctracted from the Twitter dataset had to be constructed by status update replies grouped by the original status update. This was done utilizing SQL queries to create a temporary table containing any "status_id" with four or more replies. Then, this table was joined with the original dataset to extract all status updates replies that were replies to the temporary table of status updates and ordered by timestamp. The pseudocode in figure 6.1 demonstrates how this process was executed.

To reduce memory load when pre-processing the Yelp dataset, the datasets had to be sorted by timestamps ascendingly per user so that the dataset could be read as a stream without requiring intermediate storage in memory.

```
SELECT
    t.in_reply_to_status_id as t_id,
    t.id as re_id,
    t.geo as geo,
    t.place as place,
    t.text as text,
    t.timestamp_ms as timestamp_ms,
FROM TwitterDB as t
INNER JOIN
(
    SELECT in_reply_to_status_id as t_id
    FROM twitterDB
    WHERE lang = 'en'
    GROUP BY t_id
    HAVING count(t_id) > 3
) as j
    ON t.t_id = j.t_id
WHERE t.lang = 'en'
ORDER BY t.t_id, t.timestamp_ms ASC
```

Figure 6.1: Pseudocode creating join table of twitter reply trajectories with 4 or more points in English language

A problem with the Twitter dataset is that the status updates in the snapshot aren't limited to only one language. As this spatio-textual TraClus algorithm utilizes textual similarity to cluster points, filtering on language was necessary. According to a 2011 study [31] the language most frequently used in the service is English. Thus, for this thesis only status updates containing English text are evaluated.

The Twitter application uses a language detection algorithm [32] using n-grams to identify which language was used in each status update. This information is available in the Twitter data object and could be used to filter the dataset.

In the Yelp dataset, the vast majority of the textual reviews were written in English as the businesses reviewed were only located in the North American region. Figure 6.2 illustrates the geospatial range of locations for the businesses in the Yelp data set.[3] We can then assume that the overwhelming majority of reviews contained in the dataset were written in English. Thus, no filtering based on language was deemed necessary for the Yelp dataset.

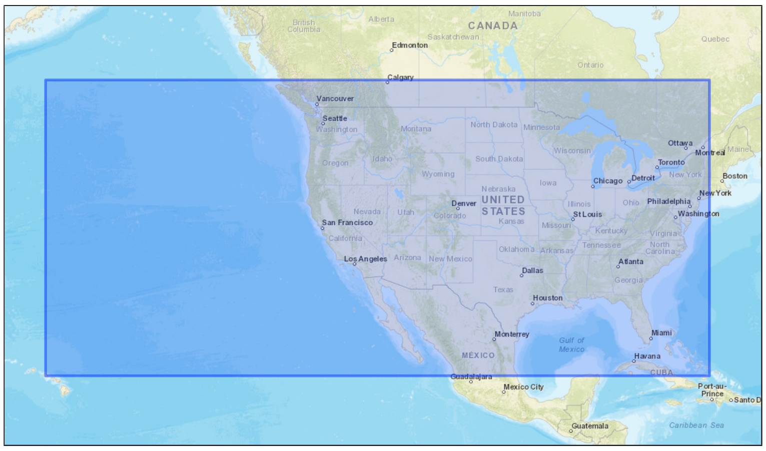


Figure 6.2: Maximum and minimum latitude and longitude coordinate bounding box in Yelp Business data set

During pre-processing, all terms in were converted to lower-case to remove false negatives on identical words in different cases. The Twitter dataset contained many instances of terms that were capitalized in unexpected places, such as in cases where the entire text message was capitalized to convey anger or where certain words were unusually capitalized as a pop-culture reference or to convey irony. This also occurred in the Yelp dataset, but not to the same degree mostly because of the more formal nature of the medium. As the Twitter dataset contained far more use of slang, special characters to dress up the text and even hyperlinks, a deeper lexical analysis was necessary to make text more comparable. The most prominent example of this is the inclusion of "hash-tagged" terms. These are terms that feature a pound sign (#) pre-fix that users include in their text messages to allow other users to find similar status updates more easily or to amplify the own users' status update in search rankings.

In order to gather meaningful keywords from the textual comparisons, stop words should be removed from the text. This can be done statistically by utilizing a dictionary and stripping from each text string words that occurred too rarely or too often, as described

[3] Generated with National Geographic MapMaker tool: `https://mapmaker.nationalgeographic.org` (Accessed: 16th Feb 2021)

in chapter 4.2. For this thesis, each dataset utilized its own respective dictionary to filter out stop words built from words existing in each dataset. Each dictionary counted the frequency of each word and the occurrence rate of each word. The occurrence rate was calculated by dividing the frequency of the word by the total amount of words in the dataset as shown in equation 9. Words with very low frequencies and words with very high occurrence rates were eliminated during pre-processing.

$$occurrence\_rate_n = \frac{word_{n_{frequency}}}{dataset\_wordcount} \tag{9}$$

Tables 6.3 and 6.4 describe the byte-sizes of the datasets before and after pre-processing. Table 6.4 also describes the final trajectory and point counts of the processed datasets. Table 6.5 displays the difference in textual data character count distribution between the two data sets.

Dataset	Size (bytes)
Twitter	162.7 MB
Yelp Review	6.1 GB
Yelp Business	149.3 MB

Table 6.3: Original dataset sizes

Dataset	Size (bytes)	% of original	Unique points	Unique Trajectories	Bytes per point
Twitter	49 MB	30.1%	215,350	33,988	227 bytes
Yelp	4.2 GB	≈ 69.0%*	5,848,605	441,586	718 bytes

*size of business dataset not accounted for in calculation

Table 6.4: Pre-processed dataset sizes and reduction

Dataset	Average word count	Average word count after processing	% of original
Yelp	113.43	34.65	30.54%
Twitter	14.098	3.60	25.53%

Table 6.5: Average text character count in datasets

As displayed in Table 6.4, the Yelp dataset had a much higher average bytes per data point. This is due to the larger textual contents per data point. The final processed data set was also much larger byte-wise than the Twitter dataset. To make evaluation of the two datasets when run by the spatio-textual TraClus algorithm directly comparable, the Yelp dataset had to be shrunk to a comparable size. Both the Yelp and the Twitter datasets were further shrunk to various sizes to measure differences in runtime based on dataset size. The shrunk sizes of the datasets will be presented in chapter 7.

After textual pre-processing, the textual contents of the spatio-textual databases shrunk tremendously. The Yelp dataset contained 113.43 words per review and shrunk down to 34.65 words per review. The Twitter dataset contained a much smaller amount of textual content initially than the Yelp dataset at approximately 14 words per entry, shrinking down to 3.6 words per entry after processing. As figure 6.5 demonstrates, both data sets shrunk in textual contents at a very similar rate after textual pre-processing. The textual portion of the Twitter dataset ended up at 25.5% of its original size, while the textual portion of the Yelp dataset ended up at 30.5% of its original size.

6.3 Implementation

In 2017 the TraClus algorithm was implemented in Java and published as a repository by github user loburliu.[4] This implementation was adapted for this thesis with modifications to the TraClus distance function and representative line generation.

The implementation of a spatio-textual TraClus algorithm was inspired by the 2017 spatio-textual k-Means [7] utilizing the Jaccard Similarity Coefficient in tandem with spatial euclidian distances to cluster in both spatial and textual dimensions.

6.3.1 Distance Functions

The original TraClus paper suggests using euclidian distances for spatial clustering. This was extended to the spatio-textual implementation and calculated a distance d as presented in equation 10.

$$d(long, lat) = \sqrt{(lat_2 - lat_1)^2 + (long_2 - long_1)^2} \tag{10}$$

However, a spatio-textual implementation of the TraClus algorithm requires modification of the distance functions between data points such that textual similarity can be expressed as a distance and compared with the spatial euclidian distance between two points. The latitude and longitude points for each point had to be scaled within a range $[0, 1]$ to be compatible with the Jaccard Similarity Coefficient which returns a similarity measure within a range of $[0, 1]$. This was done by finding the maximum and minimum latitude and longitude points present in each respective dataset and scaling each location point p as shown in equation 11.

$$p_{scaled} = \frac{p - p_{min}}{p_{max} - p_{min}} \tag{11}$$

To make the spatial trajectories more human readable for visualization and manual evaluation, this domain space was increased from $[0, 1]$ to $[0, 1000]$ in both spatial dimensions as well as the textual dimension.

The spatio-textual k-Means implementation utilized a function to inverse proportionally weigh the importance of the spatial and the textual dimension. This technique is extended to the spatio-textual implementation of the TraClus algorithm as shown in equation 12.

$$d = (\alpha \cdot d_{spatial}) + (1 - \alpha) \cdot d_{textual} \tag{12}$$

How these two distance functions were combined to create a inversely proportional distance function is demonstrated in Algorithm 6.1. The method of measuring distances

[4] TraClusAlgorithm in Java by loburliu: `https://github.com/luborliu/TraClusAlgorithm`
(Accessed: 13th March 2020)

from one spatial point to another point is utilized in two of the three distance functions utilized in the TraClus algorithm. Both perpendicular and parallell distances utilize the following distance function. The only distance function textual distance is not evaluated in is the TraClus angular distance function described in equation 7.

Algorithm 6.1: Spatio-textual TraClus distance function

```
Input  : Two spatio-textual data points $p(x_1, y_1, t_1)$ and $p(x_2, y_2, t_2)$
Input  : A weight = $\alpha$
Output : A distance = $d$

Spatial distance $d_s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
Textual distance $d_t = JaccardSim(t_1, t_2)$
return $(\alpha * d_t) + (1 - (\alpha * d_s))$
```

6.3.2 Representative terms generation

When the TraClus algorithm clusters line segments, a representative trajectory will be generated to represent each resulting cluster. This method of describing the trends inherent in the datasets should be extended to the textual dimension as well to offer the user feedback in the textual dimension.

During TraClus spatial clustering, original point coordinates are preserved as is without manipulation. In the same vein, all textual keywords for each point clustered should be included in the line segment clusters to preserve the integrity of the spatio-textual data foundation. It is only during representative trajectory generation that the keywords should be filtered or transformed as it is also the only place in the algorithm where spatial data points are generated to generate a model for the clusters. In this thesis, each representative line will include a selection of the most frequent keywords occurring in the cluster. Algorithm 6.2 demonstrates how top-N keywords were selected for any given cluster C_k.

Algorithm 6.2: spatio-textual TraClus Representative text generation

```
Input  : Clusters $C_k = \{p_{xyt_1}, p_{xyt_2}, \dots, p_{xyt_n}\}$
Output : Top $N$ ocurring terms in cluster

HashMap(String: word, Integer: count) $M$;
foreach $p_{xyt_n} \in C_k$ do
    foreach $t_k \in c_{xyt_n}$ do
        if $t_k \notin M$ then
            $M(t_k) :\longleftarrow 1$;
        else
            $M(t_k) : M(t_k) + 1$;
        end
    end
end
sortDesc($M$);
return top($N$) $\in M$
```

7 Experimental Results

This section will begin with a brief presentation of the data derived from the Twitter and Yelp datasets and then present the results of the spatio-textual TraClus algorithm both by measuring runtime and by visualization.

7.1 Twitter Dataset

For evaluating the Twitter dataset, four tests were ran at progressively reduced sizes of the dataset. Table 7.1 shows the four different datasets and their respective sizes. When reducing the datasets to different sizes to measure differences in run time, care was taken in making sure whole trajectories were preserved and without manipulating their contents. Thus, the reduction had to be based on the trajectories themselves. However, the largest factor impacting run time was the point count rather than the trajectory count. Differences between points in word count after textual pre-processing was also a large factor.

The processed Twitter dataset contained 33,988 unique trajectories. was then reduced 25% three times by using the SQL "LIMIT *N*" option using a join table 7.1. The table used was not sorted to reduce bias, so the order of trajectories selected for each data partition was coincidental.

Unique trajectories	% of original count	Unique points	% of original count	Byte size
33,988	100.0%	215,350	100.00%	49.0 MB
25,491	75.0%	160,744	74.64%	36.6 MB
16,994	50.0%	107,729	50.02%	24.4 MB
8,497	25.0%	53,840	25.00%	12.1 MB

Table 7.1: Twitter data subsets used in spatio-textual TraClus algorithm evaluation

Figure 7.1 visualizes the trajectories in the largest Twitter dataset at 215,350 data points contained in all 33,988 trajectories.

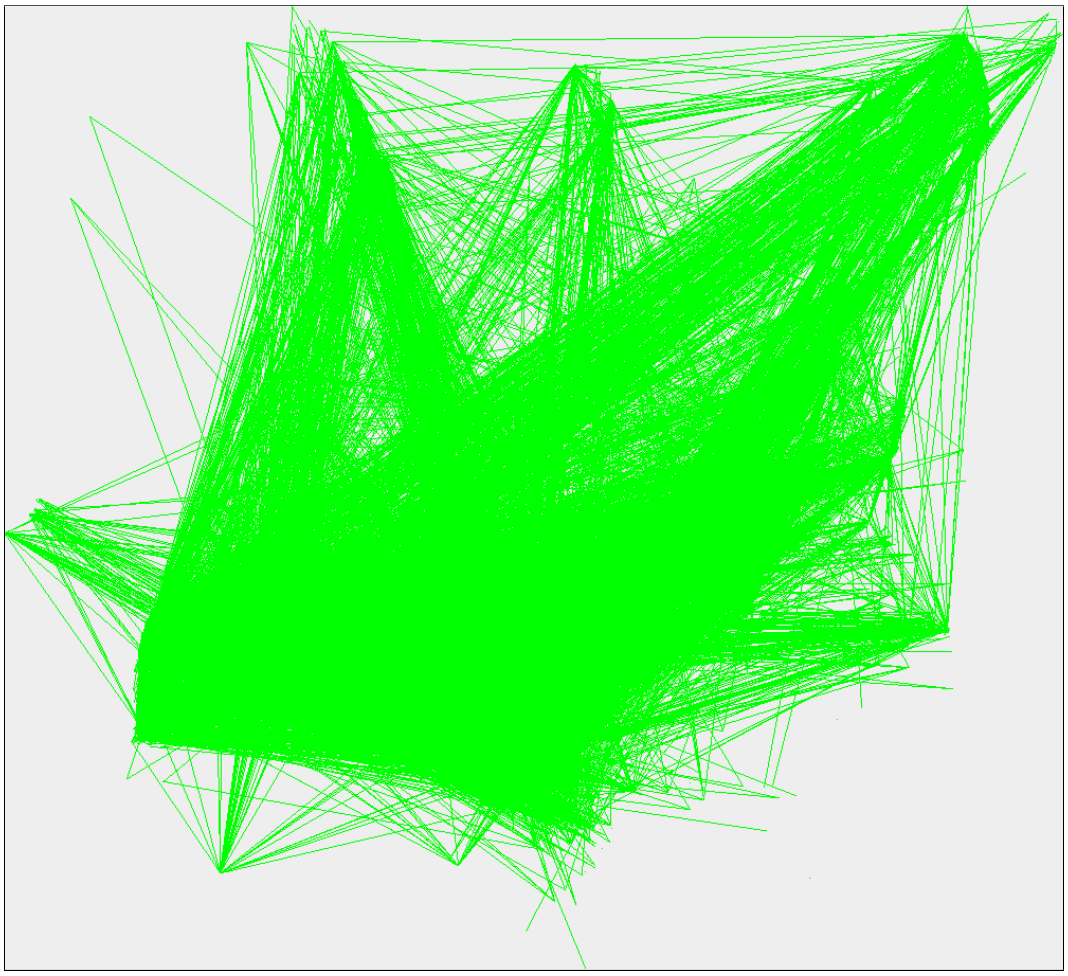


Figure 7.1: Twitter dataset trajectories visualized

7.2 Yelp Dataset

The processed Yelp dataset was much larger than the processed Twitter dataset as shown in Table 6.4. To more accurately measure and compare the differences between them in run time and cluster quality, the Yelp dataset had to be shrunk to a comparable size without compromising the continuity of the trajectories. As with the Twitter dataset, the number of points contained was the largest impacting factor in the run time of the spatio-textual TraClus algorithm. Thus the Yelp dataset had to be reduced to by its trajectories to thresholds so that they roughly matched the point counts of the evaluated Twitter datasets. For the selection of the final yelp data subsets, the unique point count of the twitter subsets were used as references. When reducing these datasets, care was taken not to compromise the continuity of any trajectory. At 16,000 unique trajectories the dataset contained 214,571 unique points, which is comparable to the total amount of unique points in the entire Twitter dataset. This was used as the baseline for benchmarking the Yelp subsets.

As more data was available in the Yelp dataset than in the Twitter dataset, three larger data partitions at 150%, 250% and 400% of the baseline were evaluated. These larger data partitions were only used to measure runtime, as the algorithm would take exponentially more time to process as the data set increased.

Unique trajectories	% of reference count	Unique points	% of reference count	Byte size
64,000	400.0%	844,729	393.68%	612.2 MB
40,000	250.0%	531,503	247.70%	384.7 MB
24,000	150.0%	313,523	146.11%	225.7 MB
16,000	**100.0%**	**214,571**	**100.00%**	**154.0 MB**
11,700	73.12%	160,777	74.92%	115.9 MB
7,600	47.50%	108,080	50.37%	78.4 MB
3,700	23.12%	54,126	25.22%	39.4 MB

Table 7.2: Yelp data subsets used in spatio-textual TraClus algorithm evaluation

Figure 7.1 visualizes the baseline Yelp dataset with 214,571 data points contained in 16,000 trajectories.

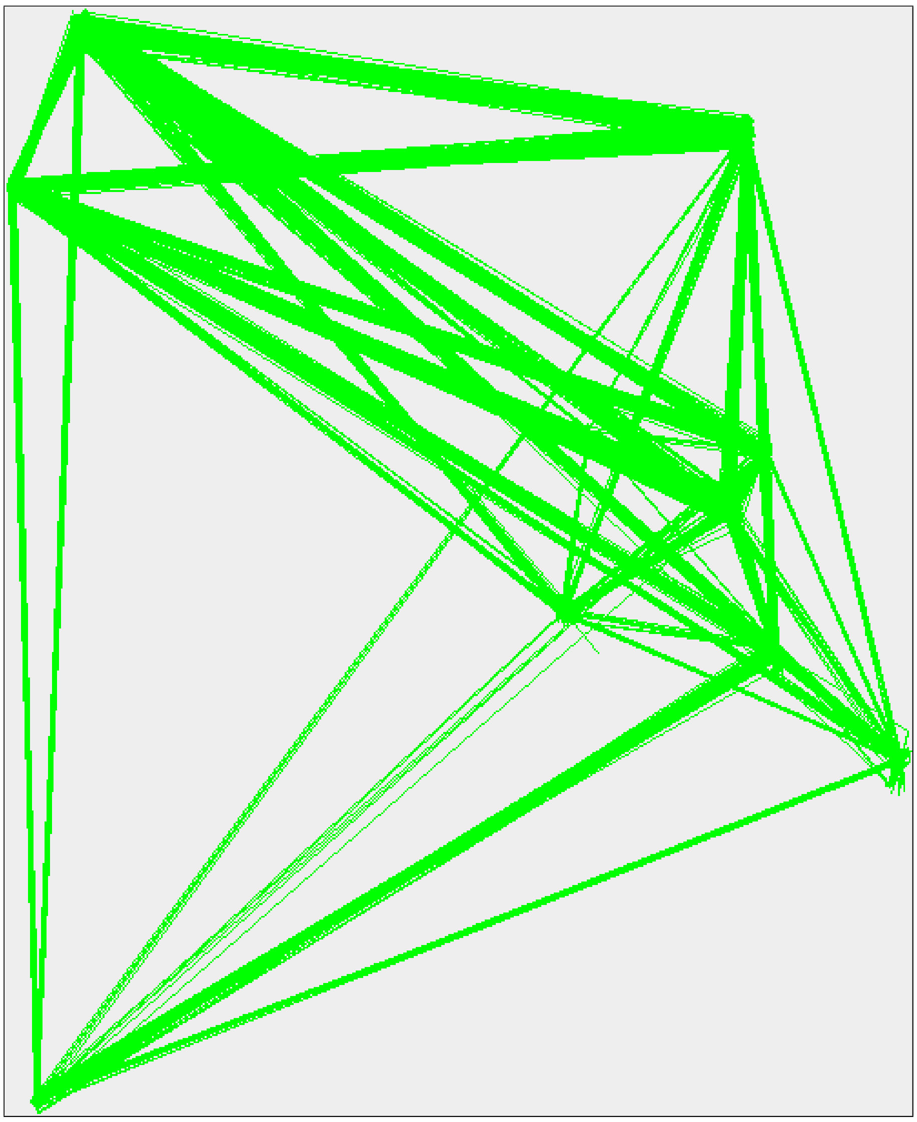


Figure 7.2: Yelp dataset trajectories visualized

7.3 Experimental Results

The spatio-textual TraClus algorithm was evaluated at different data set partitions to evaluate run time as it relates to dataset size. Then, the algorithm was evaluated on one dataset partition from each dataset with different selections in parameters. Selecting optimal parameters for clustering algorithms can be a difficult task in lack of specialized domain knowledge. For the run time evaluation, the selection of parameters was automated through the use of entropy, while the silhouette coefficient was utilized to measure the quality of the parameters selected for clustering quality evaluation.

7.3.1 Parameter Estimation

The original TraClus paper [8] utilizes the entropy theory [33] for selecting optimal parameters for ε and $minLns$. The paper states that "In information theory, the entropy relates to the amount of uncertainty about an event associated with a given probability distribution. If all the outcomes are equally likely, then the entropy should be maximal".

This method of automating the prediction of parameter selection was utilized on the smallest partitions of both the Twitter dataset and the parameters selected were used to evaluate both the Twitter dataset and the Yelp dataset at identical parameter values. For the smallest Twitter partition containing 8,497 trajectories with 53,840 unique points, the entropy function suggested $\varepsilon = 10$ and $minLns = 51$. For the run time evaluation, the quality of the clustering of lower relevancy.

To evaluate the quality of the clustering algorithm the silhouette-coefficient was used at different parameterizations to identify parameters that lead to high quality clustering for the two datasets respectively.

7.3.2 Runtime

The runtime of the spatio-textual TraClus algorithm was evaluated at four different partitions of the Twitter dataset, and seven different partitions of the Yelp dataset at similar sizes with identical parameterization. Three of the seven yelp data set partitions were larger than the largest available Twitter data set partition.

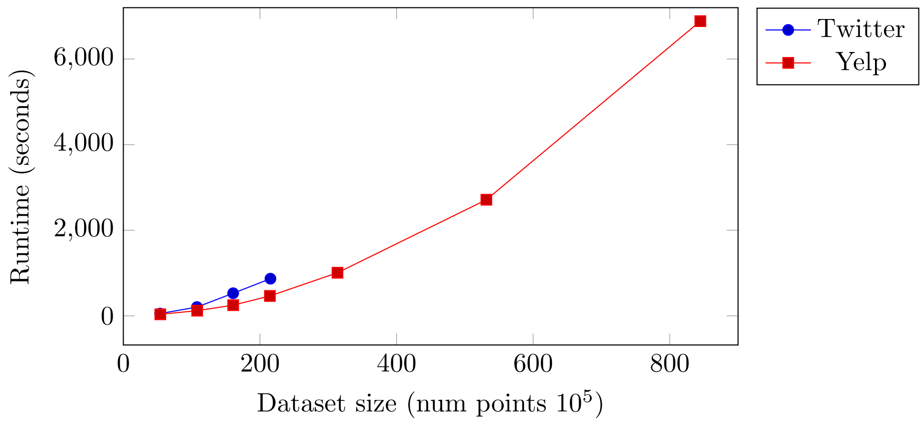


Figure 7.3: Runtime for Yelp and Twitter datasets at varying data set sizes

Unique trajectories	Unique points	Line segments	Run time (seconds)	Clustering factor
64,000	844,729	17,230	6885.79 s	0.156
40,000	531,503	11,029	2713.32 s	0.156
24,000	313,523	6,695	1007.18 s	0.155
16,000	**214,571**	**4,602**	**465.88** s	**0.155**
11,700	160,777	3,465	250.01 s	0.155
7,600	108,080	2,308	121.59 s	0.153
3,700	54,126	1,194	36.07 s	0.149

Table 7.3: Yelp spatio-textual TraClus run-time for $\varepsilon = 10, minLines = 51, \alpha = 0.3$.

Unique trajectories	Unique points	Line segments	Run time (seconds)	Clustering factor
33,988	215,350	17,280	1145.28 s	0.328
25,491	160,744	12,889	591.13 s	0.317
16,994	107,729	8,659	283.83 s	0.300
8,497	53,840	4,317	67.48 s	0.221

Table 7.4: Twitter spatio-textual TraClus run-time for $\varepsilon = 10, minLines = 51, \alpha = 0.3$.

7.3.3 Clustering results

The spatio-textual TraClus algorithm was first ran at several permutations of both the ε and $minLines$ variables with a static α-weight at 0.5. Then, acceptable measures of ε and $minLines$ were identified for the Twitter and the Yelp dataset separately through the usage of the silhouette coefficient (displayed in figure 7.4 and 7.5 respectively), before being evaluated again at permutations of the α-weight. The silhouette coefficient was again measured before clustering of the spatio-textual trajectory datasets were visualized.

$$sc = 1 - (\frac{c_{avg\_intracluster\_dist}}{c_{avg\_intercluster\_dist}}) \tag{13}$$

The silhouette coefficient measures the dissimilarity of clusters by evaluating the average intracluster distance to the highest average intercluster distance. Intracluster distance measures the average distance between points within a cluster, while the intercluster distance measures the average distance from a point in a cluster to all the points in all the other clusters, respectively. The lowest intercluster distance between the evaluated cluster and all the other clusters is then evaluated. The silhouette coefficient sc for a cluster c_n is then given by equation 13.

The silhouette coefficient is calculated for every cluster in the cluster set and will result in a number in the range of $[-1, 1]$. High values indicate that the selected item is well matched to its own cluster and is a poor match to the neighbouring clusters. For the analysis in this thesis, the largest silhouette coefficient returned is selected.

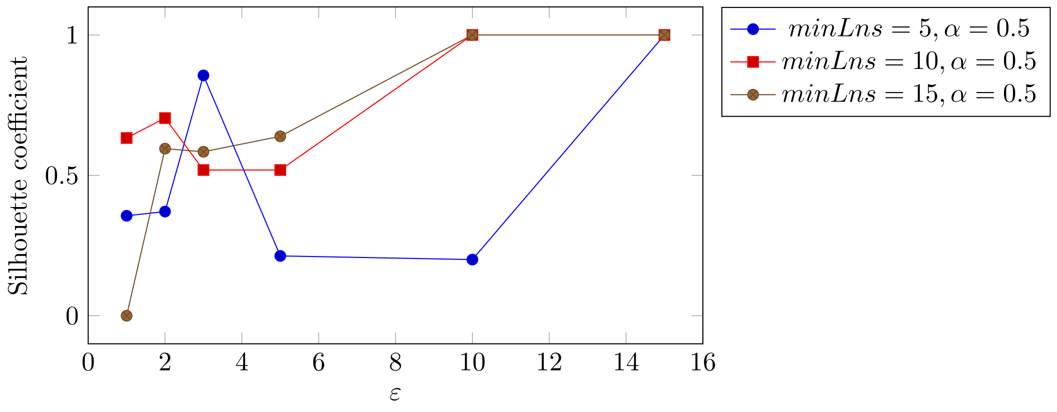


Figure 7.4: Silhouette coefficient for spatio-textual TraClus on a Twitter dataset at permutations of ε and $minLns$

$minLns$ value	ε value	Silhouette coefficient
5	1	0.356
5	2	0.371
5	3	0.856
5	5	0.213
5	10	0.200
5	15	1.000
10	1	0.633
10	2	0.704
10	3	0.519
10	5	0.519
10	10	1.000
10	15	1.000
15	1	≈0.000
15	2	0.595
15	3	0.584
15	5	0.639
15	10	1.000
15	15	1.000

Table 7.5: Silhouette coefficient for permutations of ε and $minLines$ for $\alpha = 0.5$ on Twitter dataset

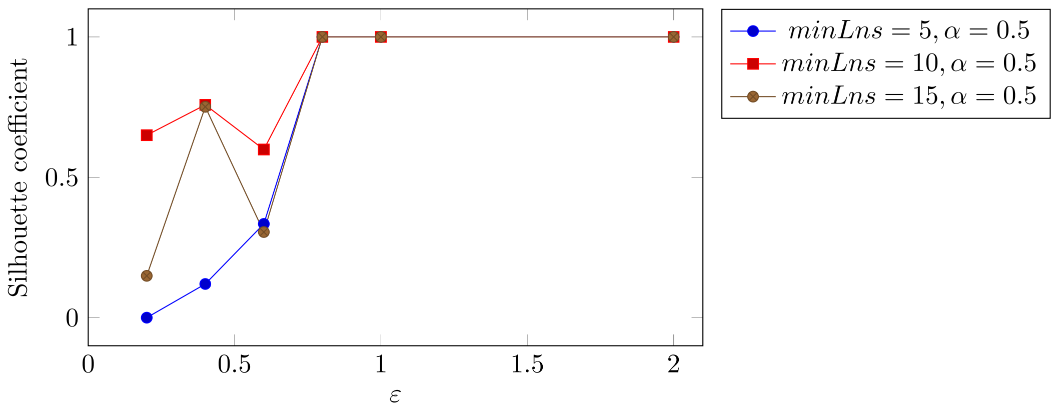


Figure 7.5: Silhouette coefficient for spatio-textual TraClus on a Yelp dataset at permutations of ε and $minLns$

$minLns$ value	ε value	Silhouette coefficient
5	0.2	≈0.000
5	0.4	0.120
5	0.6	0.334
5	0.8	1.000
5	1.0	1.000
5	2.0	1.000
10	0.2	0.650
10	0.4	0.758
10	0.6	0.599
10	0.8	1.000
10	1.0	1.000
10	2.0	1.000
15	0.2	0.149
15	0.4	0.751
15	0.6	0.305
15	0.8	1.000
15	1.0	1.000
15	2.0	1.000

Table 7.6: Silhouette coefficient for permutations of ε and $minLines$ for $\alpha = 0.5$ on Yelp dataset

Interpreting the results for the silhouette coefficent measurements for permutations of ε and $minLns$ in the Twitter dataset, the highest rated silhouette coefficient occurred at $\varepsilon = 3$ with $minLns = 5$. All top silhouette coefficient rankings on the right side of the graph are misleading because these evaluations only returned a single spatio-textual sub-trajectory cluster and should not be used as good metrics for clustering. Due to the sparse nature of the yelp dataset, and the smaller dense regions, smaller values of ε were selected.

For silhouette coefficient evaluations of both the Yelp and Twitter datasets, values equal to 1 were discarded, as they were all results of clusterings that constructed only one singular cluster.

For the twitter dataset the parameter values $\varepsilon = 3$ and $minLns = 5$ were selected for measuring the silhouette coefficient permutations of the α-weight. For the Yelp dataset the parameter values $\varepsilon = 0.4$ and $minLns = 10$ were selected.

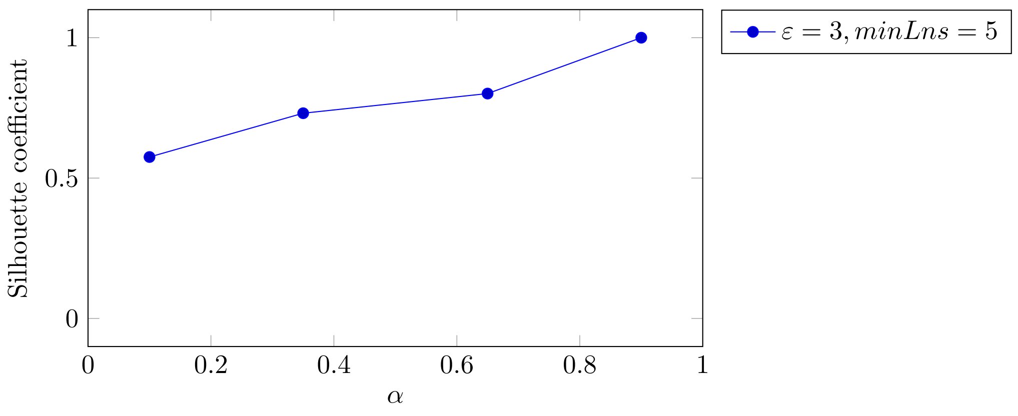


Figure 7.6: Silhouette coefficient for spatio-textual TraClus on a Twitter dataset at permutations of α

α value	Silhouette coefficient
0.1	0.575
0.35	0.731
0.65	0.801
0.9	1

Table 7.7: Silhouette coefficient for permutations of α values for $\varepsilon = 3$, $minLines = 5$ on Twitter dataset

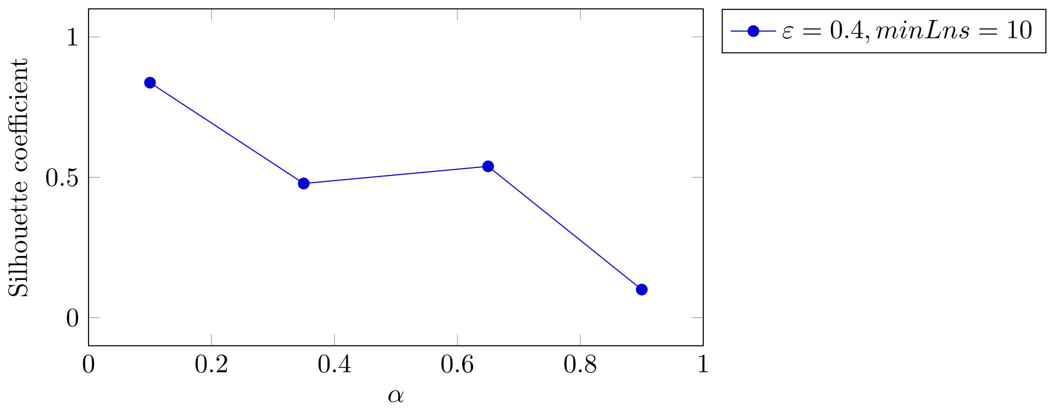


Figure 7.7: Silhouette coefficient for spatio-textual TraClus on a Yelp dataset at permutations of α

α value	Silhouette coefficient
0.1	0.837
0.35	0.478
0.65	0.539
0.9	0.042

Table 7.8: Silhouette coefficient for permutations of α values for $\varepsilon = 3$, $minLines = 5$ on Yelp dataset

In the end the parameters $\varepsilon = 3$, $minLns = 5$ and $\alpha = 0.35$ were selected for the twitter dataset, and the parameters $\varepsilon = 0.4$, $minLns = 10$ and $\alpha = 0.65$ were selected for the yelp dataset. Figure 7.8 and 7.9 visualize the results of spatio-textual TraClus for the Twitter dataset and Yelp dataset, respectively.

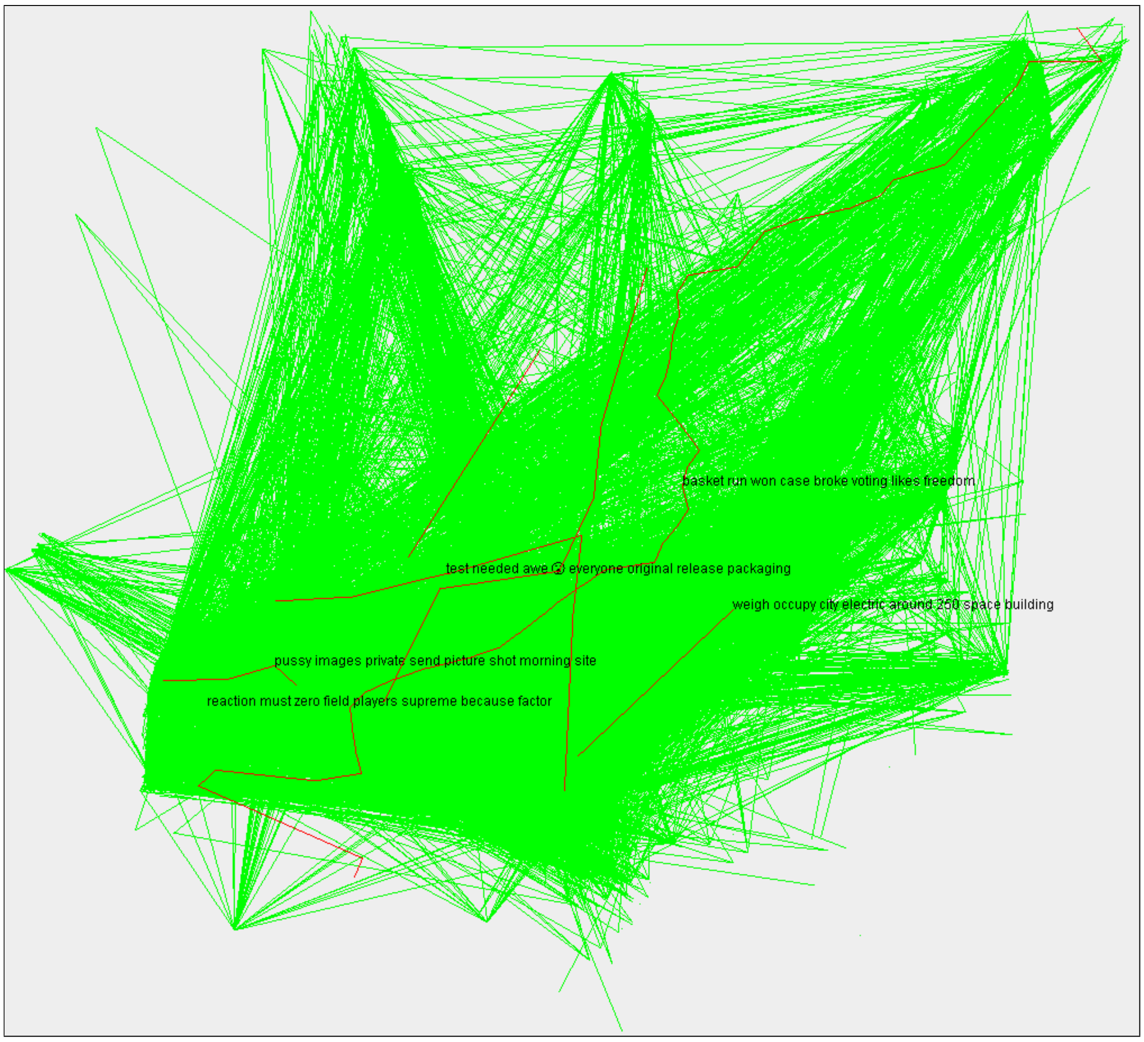


Figure 7.8: Spatio-textual TraClus on a Twitter dataset using $\varepsilon = 3$, $minLns = 5$ and $\alpha = 0.35$

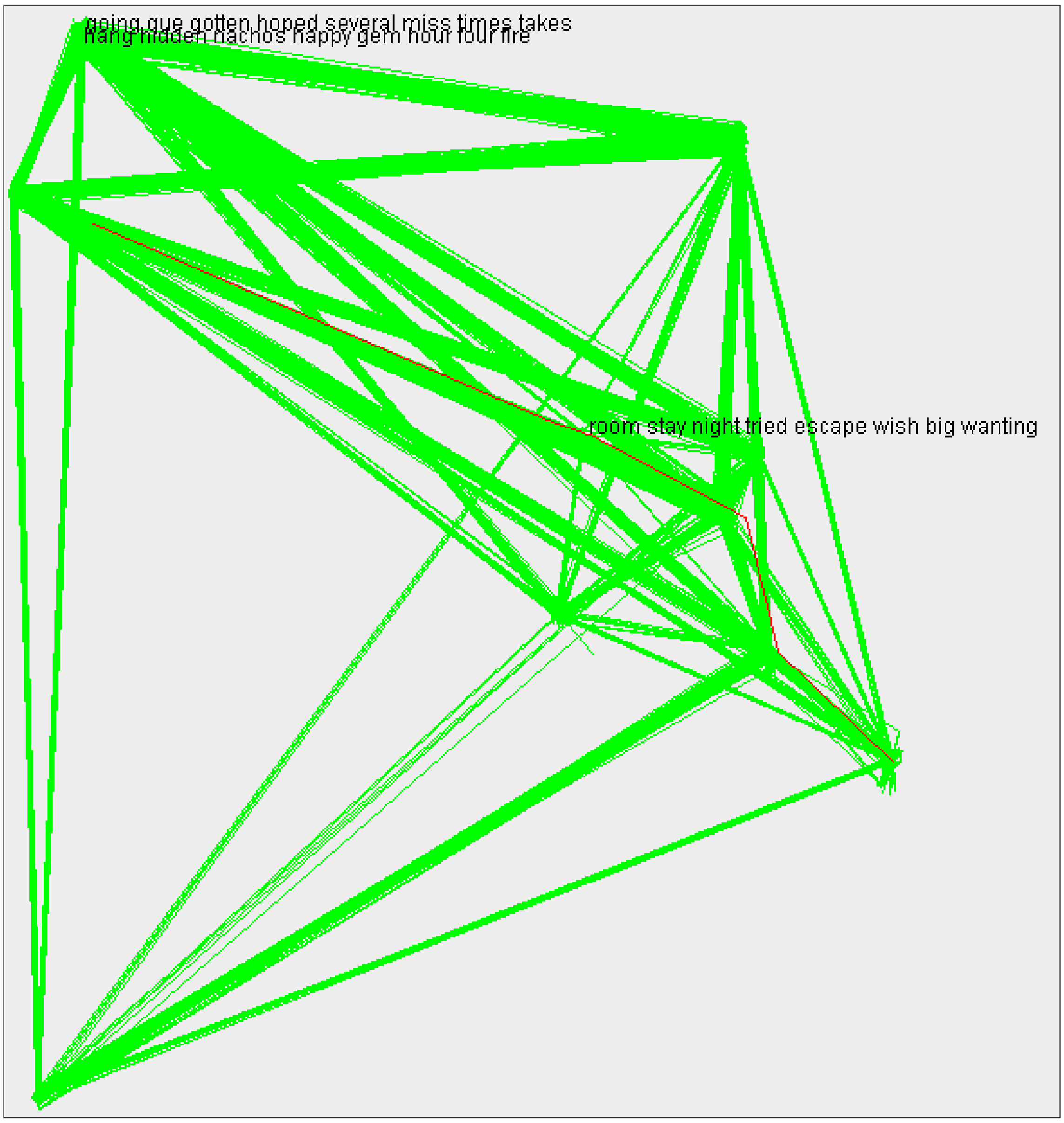


Figure 7.9: Spatio-textual TraClus on a Yelp dataset using $\varepsilon = 0.4$, $minLns = 10$ and $\alpha = 0.65$

8 Discussion

This section will discuss the impact and quality of the spatio-textual TraClus algorithm on the Yelp and Twitter datasets, methods and algorithms used in evaluating the use of the TraClus-algorithm to cluster spatio-textual trajectories in context of scalability and eligibility.

8.1 Datasets

The stop word filtering of the Twitter and Yelp datasets was executed through dictionary filtering. For each dataset a dictionary was generated consisting of word count and total word occurrence for the entire dataset. Filtering was executed by removing words that occurred extremely rarely or too frequently. As the data sets contain a large amount of text terms, manually selecting stop words is unrealistic and thresholds were used instead. The Yelp dataset primarily consisted of reviews of restaurants and bars, and thus its dictionary was dominated by entries describing food, drinks and the service industry.

A consequence of building a dictionary that is domain dominated to a large degree is that certain words with high level of interestingness were filtered out due to occurring too frequently. Table 8.1 demonstrates how some content relevant terms occurred more frequently than certain non-relevant terms. The less descriptive preposition "before" occurrs fewer times than the word "chicken" in the dataset. And although common prepositions should be removed, there is no trivial way to automate this statistically without also excluding the highly domain relevant word "chicken". This effect did not impact the Twitter dataset's dictionary, as the conversations held varied widely in subject matter.

Term	Count	Relative Term Occurrence
than	914465	0.001149
		...
chicken	910830	0.001145
		...
before	648413	0.000815

Table 8.1: Sample of Yelp dictionary

However, the Twitter dataset suffered greatly from sparsity in its textual data. As mentioned in Chapter 6.2.1, the Twitter service has an absolute limit of 240 characters per textual message. When stop word elimination is executed on the Twitter dataset, an immense amount of information is removed, as demonstrated in table 6.5. While similar rates of textual content was removed from the Yelp dataset, the textual content remaining in Some points in the trajectories contained no textual content at all after stop word elimination. This lead to much poorer clustering for higher α weighting the textual dimension in the spatio-textual distance function as demonstrated by equation 12.

The Yelp dataset suffered from containing sparse spatial data. Figure 7.2 demonstrates large empty areas in the data model. Without domain knowledge, it is not possible to confirm the reason for these patterns. One could speculate that these patterns are a result of vacationing users travelling very long distances by plane and reviewing local businesses during their stay before travelling elsewhere to review other businesses. The sparsity of the data set and the long line segments makes identifying good parameters very challenging. If low ε-values are selected local popular routes will be prioritized, but macro-level travelling routes will be ignored. A compromise would be to utilize select data slices in the set based on sub-regions, rather than evaluating the dataset as a whole.

Trajectory clustering methods with a focus on semantic regions would be better suited for clustering a data set of this nature. One such method is the SemTraClus algorithm [34] suggested in 2019, where the researchers adopted the TraClus algorithm to prioritize semantic regions. Clustering geospatial data can be done in multiple ways. In this thesis, individual latitude and longitude coordinates were rescaled such that any coordinate value $p \in [0, 1]$ based on the local minimum and maximum latitude and longitude values present in the original datasets. This was done to allow geospatial distances to be comparable to textual distances produced by the Jaccard Similarity Coefficient.

A trend that can be observed in the visualization of the resulting clusters is that the outline of the visualized trajectories in figure 7.1 and 7.2 tend towards circular shapes. Projections such as the Mercator Projection could be utilized during pre-processing to transform coordinates so that the outlier would trend towards rectangular shapes. However, this would not solve an inherent problem in the data sets. Both data sets are based on mobile social network data where trajectories are inferred based on timestamp orderings. The relative movement between two data consecutive entries can potentially be very skewed. In the case of the Yelp dataset, a user might travel very long distances between each entry, or publish reviews in a different order than the establishments were visited.

In the case of the Twitter dataset, a user can reply to a status message instantly from the other side of the globe. This can cause issues in how the shortest distance between two points is calculated if an interaction happens at the extremes of two spherical coordinate limits, such as $latitude \in [-180, 180]$. If a status update from $longitude = -175$ is interacted with at $longitude = 175$ in a naive implementation, the spatial distance between the two will be calculated to be 150 units, when in reality the real distance between the two should be 10 units.

The problem of the shortest path between two geospatial locations travelling over the limits of the spherical coordinate system can potentially be solved by transforming the geospatial domain into three dimensions as demonstrated by equation 14

$$\begin{aligned}
\phi &= \frac{latitude \cdot \pi}{180}, \\
\lambda &= \frac{longitude \cdot \pi}{180} \\
\\
X &= cos(\phi) \cdot cos(\lambda), \\
Y &= cos(\phi) \cdot sin(\lambda), \\
Z &= sin(\phi)
\end{aligned} \tag{14}$$

However, this will incur greater computational costs as the DBSCAN algorithm is documented to perform considerably worse in multi-dimensional spaces [24].

8.2 Evaluation of TraClus for spatio-textual data

Evaluation of a clustering algorithm is not a trivial task. There exists multiple methods of evaluating the quality of a clustering algorithm, from runtime efficiency to graphic visualization. In this thesis, the efficiency of the spatio-textual TraClus algorithm was measured in run-time for different data sub set sizes. The quality of the clustering was measured using the silhouette coefficient to identify parameters that lead to better clustering results.

8.2.1 Runtime

Figure 7.3 describes the differences in runtime for the data foundations at subsets of varying sizes. The spatio-textual TraClus algorithm was evaluated with identical parameterization across all partitions of the two datasets. Interestingly, the Yelp dataset performed better than the twitter datasets at similar measures of data points. A large reason to this is the spatially sparse nature of the Yelp dataset. The Twitter dataset is much more densely populated and evaluates a much higher amount of line segments in contrast to the Yelp data set. For 215,350 data points, the Twitter data set took 19 minutes to process compared to the Yelp data set which took less than 8 minutes to process at 214,571 data points. At these two runtime evaluations the Yelp dataset evaluated 4,602 line segments in contrast to Twitter dataset which evaluated 17,280 line segments.

The line segments alone do not tell the full story. At 844,729 unique data points, the yelp data set evaluated 17,230 line segments. This is a comparable number of line segments to the largest twitter evaluation. However, the Yelp dataset took 114 minutes to process at this data set partition. A data partition at 168,000 unique trajectories consisting of 2,232,257 unique points was scheduled for evaluation, but was stopped manually after running continuously for over 24 hours.

8.2.2 Cluster evaluation

The original TraClus paper utilized visualization as a method of evaluation, and the same principles will be extended to the spatio-textual TraClus algorithm. In addition to presenting the representative trajectories visually, a top N occuring selection of keywords from each representative trajectory will be presented in the visualization.

Additionally, the silhouette coefficient [35] was utilized to evaluate the quality of the clusters generated.

The results of the spatio textual TraClus-algorithm varied heavily based on the selection of input parameters. Understanding which parameters Methods such as the silhouette coefficient [35] was utilized can be extended to the TraClus evaluate the quality of clustering performed. The silhouette coefficient is calculated by measuring the

8.3 Future Work

The efficiency of the spatio-textual TraClus algorithm is heavily impacted by textual comparisons. The poor scalability negatively impacts the usefulness of the implementation, as the slow computation will deter enterprises and research projects from using the algorithm in larger scale systems. This can be compensated for by combining research from a 2017 paper exploring possibilities of parallelizing the DBSCAN algorithm through the Apache Spark system [36].

Other methods, such as the 2018 Hierarchical adaption of the TraClus algorithm [17] can potentially be combined with the findings of this thesis to utilize the increased computational efficiency afforded by hierarchical methods to further improve the run time of the density based clustering methods of a spatio-textual TraClus algorithm.

9 Conclusion

This thesis has sought out to research methods in the data mining field and apply them on the problem of clustering spatio-textual trajectories on large datasets.

This thesis demonstrates the viability of performing spatio-textual clustering using Jaccard Coefficiency Similarity in tandem with the TraClus sub-trajectory clustering method. This thesis extends concepts implemented in other research [7] to bridge the gap between textual and spatial similarity.

The validity of the resulting clusters have been confirmed using the Silhouette Coefficient to identify good parameters for the clustering technique. Methods of visualization have been applied to the resulting clusters to visually confirm that the algorithm is capable of identifying good clusters.

The algorithm has been measured in run time efficiency and although the inclusion of a textual dimension to a spatial clustering technique significantly impacts the efficiency of the original algorithm, the research shows [36] [17] that other interesting work can be combined with this thesis' work to increase its efficiency.

References

[1] J. Naisbitt, *Megatrends: Ten New Directions Transforming Our Lives.* Manhattan, NY: Warner Books, 1984. pp. 17.

[2] U. M. Feyyad, "Data mining and knowledge discovery: making sense out of data," *IEEE Expert*, vol. 11, no. 5, pp. 20–25, 1996.

[3] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, vol. 1. Cambridge, MA (US): Morgan Kaufmann, 4 ed., 2016. pp. 6, 33-35.

[4] S. S. Anand, D. A. Bell, and J. G. Hughes, "The role of domain knowledge in data mining," in *Proceedings of the fourth international conference on Information and knowledge management*, pp. 37–43, 1995.

[5] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487–499, Citeseer, 1994.

[6] S. Hosseinimotlagh and E. E. Papalexakis, "Unsupervised content-based identification of fake news articles with tensor decomposition ensembles," in *Proceedings of the Workshop on Misinformation and Misbehavior Mining on the Web (MIS2)*, 2018.

[7] D.-W. Choi and C.-W. Chung, "A k-partitioning algorithm for clustering large-scale spatio-textual data," *Information Systems*, vol. 64, pp. 1–11, 2017.

[8] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, (New York, NY), pp. 593–604, ACM, 2007.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, p. 226–231, AAAI Press, 1996.

[10] C. Renso, M. Baglioni, J. A. F. de Macedo, R. Trasarti, and M. Wachowicz, "How you move reveals who you are: understanding human behavior by analyzing trajectory data," *Knowledge and information systems*, vol. 37, no. 2, pp. 331–362, 2013.

[11] J. B. Oates, *Researching Information Systems and Computing.* City Road, London: SAGE Publications Ltd, 1st ed., 2006. pp. 71-92.

[12] S. Gaffney and P. Smyth, "Trajectory clustering with mixtures of regression models," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 63–72, 1999.

[13] Z. Li, M. Ji, J.-G. Lee, L.-A. Tang, Y. Yu, J. Han, and R. Kays, "Movemine: Mining moving object databases," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, (New York, NY, USA), pp. 1203–1206, ACM, 2010.

[14] F. Wu, T. K. H. Lei, Z. Li, and J. Han, "Movemine 2.0: Mining object relationships from movement data," *Proc. VLDB Endow.*, vol. 7, p. 1613–1616, Aug. 2014.

[15] P. Jaccard, “The distribution of the flora in the alpine zone. 1,” *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[16] M. D. Nguyen and W.-Y. Shin, “Dbstexc: Density-based spatio-textual clustering on twitter,” in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM ’17, (New York, NY, USA), p. 23–26, Association for Computing Machinery, 2017.

[17] D. Zhang, K. Lee, and I. Lee, “Hierarchical trajectory clustering for spatio-temporal periodic pattern mining,” *Expert Systems with Applications*, vol. 92, pp. 1–11, 2018.

[18] R. J. G. B. Campello, D. Moulavi, and J. Sander, “Density-based clustering based on hierarchical density estimates,” in *Advances in Knowledge Discovery and Data Mining* (J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, eds.), (Berlin, Heidelberg), pp. 160–172, Springer Berlin Heidelberg, 2013.

[19] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[20] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: the concepts and technology behind search.* Harlow, Essex (UK): Pearson Education Ltd., 2 ed., 2011. pp. 68-76.

[21] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques.* Burlington (MA): Elsevier, 3rd ed., 2011. pp. 74-76, 361-403.

[22] S. Lloyd, “Least squares quantization in pcm,” *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[23] M. Yedla, S. R. Pathakota, and T. Srinivasa, “Enhancing k-means clustering algorithm with improved initial center,” *International Journal of computer science and information technologies*, vol. 1, no. 2, pp. 121–125, 2010.

[24] J. Gan and Y. Tao, “Dbscan revisited: Mis-claim, un-fixability, and approximation,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’15, (New York, NY, USA), p. 519–530, Association for Computing Machinery, 2015.

[25] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “Dbscan revisited, revisited: Why and how you should (still) use dbscan,” *ACM Trans. Database Syst.*, vol. 42, July 2017.

[26] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: An efficient data clustering method for very large databases,” *SIGMOD Rec.*, vol. 25, p. 103–114, June 1996.

[27] J. Yang and M. Hu, “Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects,” in *Advances in Database Technology - EDBT 2006* (Y. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. Boehm, A. Kemper, T. Grust, and C. Boehm, eds.), (Berlin, Heidelberg), pp. 664–681, Springer Publishing Company, Incorporated, 2006.

[28] P. D. Grünwald, I. J. Myung, and M. A. Pitt, *Advances in minimum description length: Theory and applications.* MIT press, 2005.

[29] “Counting characters: Counting characters when composing tweets,” May 2018. `https://developer.twitter.com/en/docs/counting-characters`. [Accessed: 09. October 2020].

[30] “Data dictionary: Tweet object,” Oct 2018. `https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview/tweet-object`. [Accessed: 09. October 2020].

[31] L. Hong, G. Convertino, and E. H. Chi, “Language matters in twitter: A large scale study,” in *Fifth international AAAI conference on weblogs and social media*, p. 518, Citeseer, 2011.

[32] @tm, “Evaluating language identification performance,” Nov 2015. `https://blog.twitter.com/engineering/en_us/a/2015/evaluating-language-identification-performance.html`. [Accessed: 11. February 2021].

[33] C. E. Shannon, “A mathematical theory of communication,” *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[34] A. Nishad and S. Abraham, “Semtraclus: an algorithm for clustering and prioritizing semantic regions of spatio-temporal trajectories,” *International Journal of Computers and Applications*, pp. 1–10, 2019.

[35] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[36] F. Huang, Q. Zhu, J. Zhou, J. Tao, X. Zhou, D. Jin, X. Tan, and L. Wang, “Research on the parallelization of the dbscan clustering algorithm for spatial data mining based on the spark platform,” *Remote Sensing*, vol. 9, no. 12, p. 1301, 2017.

Appendices

A Twitter Object Data Format

```
{
    "contributors" : String,
    "coordinates" : {
        [ Double ] ,
        "type" : String
    },
    "created_at" : String,
    "entities": {
        "hashtags" : [
            {
                "indices" : [Long] ,
                "text" : String
            }
        ],
        "media" : [
            {
                "display_url" : String,
                "extended_url : String,
                "id" : Long ,
                "id_str" : String,
                "indices" : [Long] ,
                "media_url" : String,
                "sizes" : {
                    "large" : {
                        "h" : Long ,
                        "resize" : String,
                        "w" : Long ,
                    },
                    "medium" : {
                        "h" : Long ,
                        "resize" : String,
                        "w" : Long ,
                    },
                    "thumb" : {
                        "h" : Long ,
                        "resize" : String,
                        "w" : Long ,
                    }
                },
                "source_status_id" : Long ,
                "source_status_id_str" : String,
                "source_user_id" : Long ,
                "source_user_id_str" : String,
                "type" : String,
                "url" : String,
            }
        ],
        "symbols" : [
            {
                "indices" : [Long],
                "text" : String
            }
        ],
        "urls" : [
            {
                "display_url" : String,
                "expanded_url" : String,
                "indices" : [Long],
                "url" : String
            }
        ],
        "user_mentions" : [
            {
                "id" : Long,
                "id_str" : String,
                "indices" : [Long],
                "name" : String,
                "screen_name" : String
            }
        ]

    },
    "extended_entities" : {
        "media" : [
            "display_url" : String,
            "expanded_url" : String,
            "id" : Long,
            "id_str" : String,
            "indices" : [Long],
            "media_url" : String,
            "media_url_https" : String,
            "sizes" : {
                "large" : {
                    "h" : Long,
                    "resize" : String,
                    "w" : Long,
                },
                "medium" : {
                    "h" : Long,
                    "resize" : String,
                    "w" : Long,
                },
                "small" : {
                    "h" : Long ,
                    "resize" : String,
                    "w" : Long ,
                },
                "thumb" : {
                    "h" : Long ,
                    "resize" : String,
                    "w" : Long ,
                }
            }
        ],
        "source_status_id" : Long ,
        "source_status_id_str" : String,
        "source_user_id" : Long ,
        "source_user_id_str" : String,
        "url" : String,
        "video_info" : {
            "aspect_ratio" : [Long],
            "duration_millies" : Long ,
            "variants" : [
                {
                    "bitrate" : Long ,
                    "content_type" : String,
                    "url" : String
                }
            ]
        }
    },
    "favorite_count" : Long ,
    "favorited" : Boolean ,
    "filter_level" : String,
    "geo" : {
        "coordinates" : [Double],
        "Type" : String
    },
    "id" : Long ,
    "id_str" : String,
    "in_reply_to_screen_name" : String,
    "in_reply_to_status_id" : Long ,
    "in_reply_to_status_id_str" : String,
    "in_reply_to_user_id" : Long ,
    "in_reply_to_user_id_str" : String,
    "is_quote_status" : Boolean ,
    "lang" : String,
    "place" : {
        "attributes" : {
            "street_address" : String
        },
        "bounding_box" : {
            "coordinates" : [
                [
                    [ Double ]
                ]
            ],
            "type" : String
        },
        "country" : String,
        "country_code" : String,
        "full_name" : String,
        "id" : String,
        "name" : String,
        "place_type" : String,
        "url" : String,
    },
    "possibly_sensitive" : Boolean ,
    "quoted_status" : {
        "contributors" : String,
        "coordinates" : {
            "coordinates" : [ Double ] ,
            "type" : String
        },
        "created_at" : String,
        "entities" : {
            "hashtags" : [
                {
                    "indices" : [ Long ] ,
                    "text" : String
                }
            ],
            "media" : [
                {
                    "display_url" : String,
                    "expanded_url" : String,
                    "id" : Long ,
                    "id_str" : String,
                    "indices" : [ Long ] ,
                    "media_url" : String,
                    "media_url_https" : String,
                    "sizes" : {
                        "large" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        },
                        "medium" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        },
                        "small" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        },
                        "thumb" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        }
                    },
                    "source_status_id" : Long ,
                    "source_status_id_str" : String,
                    "source_user_id" : Long ,
                    "source_user_id_str" : String,
                    "url" : String
                }
            ],
            "symbols" : [
                {
                    "indices" : [Long],
                    "text" : String
                }
            ],
            "urls" : [
                {
                    "display_url" : String,
                    "expanded_url" : String
                    "indices" : [Long],
                    "url" : String
                }
            ],
            "user_mentions" : [
                {
                    "id" : Long ,
                    "id_str" : String,
                    "indices" : [Long],
                    "name" : String,
                    "screen_name" : String
                }
            ]
        },
        "extended_entities" : {
            "media" : [
                {
                    "display_url" : String,
                    "expanded_url" : String,
                    "id" : Long ,
                    "id_str" : String,
                    "indices" : [Long],
                    "media_url" : String,
                    "media_url_https" : String,
                    "sizes" : {
                        "large" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        },
                        "medium" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        },
                        "small" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        },
                        "thumb" : {
                            "h" : Long ,
                            "resize" : String,
                            "w" : Long ,
                        }
                    },
                    "source_status_id" : Long ,
                    "source_status_id_str" : String,
                    "source_user_id" : Long ,
                    "source_user_id_str" : String,
                    "url" : String,
                    "video_info" : {
                        "aspect_ratio" : [Long]
                        "duration_millis" : Long ,
                        "variants" : [
                            {
                                "bitrate" : Long ,
                                "content_type" : String,
                                "url" : String
                            }
                        ]
                    }
                }
```

```
            ]
        },
        "favorite_count" : Long ,
        "favorited" :  Boolean ,
        "filter_level" : String,
        "geo" : {
            "coordinates" : [ Double ] ,
            "type" : String
        },
        "id" : Long ,
        "id_str" : String,
        "in_reply_to_screen_name" : String,
        "in_reply_to_status_id" : Long ,
        "in_reply_to_status_id_str" : String,
        "in_reply_to_user_id" : Long ,
        "in_reply_to_user_id_str" : String,
        "is_quote_status" :  Boolean ,
        "lang" : String,
        "place" : String,
        "possibly_sensitive" :  Boolean ,
        "quoted_status_id" : Long ,
        "quoted_status_id_str" : String,
        "retweet_count" : Long ,
        "retweeted" :  Boolean ,
        "scopes" : {
            "followers" :  Boolean ,
            "place_ids" [String]
        }
        "source" : String,
        "text" : String,
        "truncated" :  Boolean ,
        "user" : {
            "contributors_enabled" :  Boolean ,
            "created_at" : String,
            "default_profile" :  Boolean ,
            "default_profile_image" : String,
            "description" : String,
            "favorites_count" : Long ,
            "follow_request_sent" : String,
            "followers_count" : Long ,
            "following" : String,
            "friends_count" : Long ,
            "geo_enabled" :  Boolean ,
            "id" : Long ,
            "id_str" : String,
            "is_translator" :  Boolean ,
            "lang" : String,
            "listed_count" : Long ,
            "location" : String,
            "name" : String,
            "notifications" : String,
            "profile_background_color" : String,
            "profile_background_image_url" : String,
            "profile_background_image_url_https" : String,
            "profile_background_tile" : Boolean ,
            "profile_banner_url" : String,
            "profile_image_url" : String,
            "profile_image_url_https" : String,
            "profile_link_color" : String,
            "profile_sidebar_border_color" : String,
            "profile_sidebar_fill_color" : String,
            "profile_text_color" : String,
            "profile_use_background_image" : Boolean ,
            "protected" : Boolean ,
            "screen_name" : String,
            "statuses_count" : Long ,
            "time_zone" : String,
            "url" : String,
            "utc_offset" : Long ,
            "verified" : Boolean ,
        }
    },
    "quoted_status_id" : Long ,
    "quoted_status_id_str" : String,
    "retweet_count" : Long ,
    "retweeted" : Boolean ,
    "scopes" : {
        "place_ids" : [Long]
    },
    "text" : String,
    "timestamp_ms" : String,
    "truncated" : Boolean
    "user" : {
        "contributors_enabled" : Boolean ,
        "created_at" : String,
        "default_profile" : Boolean ,
        "default_profile_image" : String,
        "description" : String,
        "favorites_count" : Long ,
        "follow_request_sent" : String,
        "followers_count" : Long ,
        "following" : String,
        "friends_count" : Long ,
        "geo_enabled" : Boolean ,
        "id" : Long ,
        "id_str" : String,
        "is_translator" :  Boolean ,
        "lang" : String,
        "listed_count" :  Long ,
        "location" : String,
        "name" : String,
        "notifications" : String,
        "profile_background_color" : String,
        "profile_background_image_url" : String,
        "profile_background_image_url_https" : String,
        "profile_background_tile" :  Boolean ,
        "profile_banner_url" : String,
        "profile_image_url" : String,
        "profile_image_url_https" : String,
        "profile_link_color" : String,
        "profile_sidebar_border_color" : String,
        "profile_sidebar_fill_color" : String,
        "profile_text_color" : String,
        "profile_use_background_image" :  Boolean ,
        "protected" :  Boolean ,
        "screen_name" : String,
        "statuses_count" :  Long ,
        "time_zone" : String,
        "url" : String,
        "utc_offset" :  Long ,
        "verified" :  Boolean ,
    }
}
```

B Yelp Review Object Data Format

```
{
    "business_id" : String,
    "cool" : Long ,
    "date" : String,
    "funny" : Long ,
    "review_id" : String
    "stars" : Double ,
    "text" : String,
    "useful" : Long ,
    "user_id" : String
}
```

C Yelp Business Object Data Format

```
{
    "address" : String,
    "attributes" : {
        "AcceptsInsurance" : String,
        "AgesAllowed" : String,
        "Alcohol" : String,
        "Ambience" : String,
        "BYOB" : String,
        "BYOBCorkage" : String,
        "BestNights" : String,
        "BikeParking" : String,
        "BusinessAcceptsBitcoin" : String,
        "BusinessAcceptsCreditCards" : String,
        "BusinessParking" : String,
        "ByAppointmentOnly" : String,
        "Caters" : String,
        "CoatCheck" : String,
        "Corkage" : String,
        "DietaryRestrictions" : String,
        "DogsAllowed" : String,
        "DriveThru" : String,
        "GoodForDancing" : String,
        "GoodForKids" : String,
        "GoodForMeal" : String,
        "HairSpecializesIn" : String,
        "HappyHour" : String,
        "HasTV" : String,
        "Music" : String,
        "NoiseLevel" : String,
        "Open24Hours" : String,
        "OutdoorSeating" : String,
        "RestaurantsAttire" : String,
        "RestaurantsCounterService" : String,
        "RestaurantsDelivery" : String,
        "RestaurantsGoodForGroups" : String,
        "RestaurantsPriceRange2" : String,
        "RestaurantsReservations" : String,
        "RestaurantsTableService" : String,
        "RestaurantsTakeOut" : String,
        "Smoking" : String,
        "WheelchairAccessible" : String,
        "WiFi" : String,
    },
    "business_id" : String,
    "categories" : String,
    "city" : String,
    "hours" : {
        "Friday" : String,
        "Monday" : String,
        "Saturday" : String,
        "Sunday" : String,
        "Thursday" : String,
        "Tuesday" : String,
        "Wednesday" : String,
    },
    "is_open" : Long ,
    "latitude" : Double ,
    "longitude" : Double ,
    "name" : String,
    "postal_code" : String,
    "review_count" : Long ,
    "stars" : Double ,
    "state" : String
}
```

D Processed Spatiotextual Trajectory Data Object

```
{
    "id" : Int,
    "numPoints" : Int,
    "points" : [
        {
            "id" : Int,
            "timestamp" : Long ,
            "keywords" : [String] ,
            "latitude" : Double ,
            "longitude" : Double ,
        }
    ]
}
```