RELEVANCE FEEDBACK AND SPARSITY HANDLING METHODS FOR TEMPORAL DATA

A DISSERTATION SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

By
Bahaeddin ERAVCI
July 2018

RELEVANCE FEEDBACK AND SPARSITY HANDLING METHODS FOR TEMPORAL DATA

By Bahaeddin ERAVCI

July 2018

We certify that we have read this dissertation and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Özgür Ulusoy(Advisor)

Hakan Ferhatosmanoğlu(Co-Adviser)

A. Enis Çetin

Uğur Güdükbay

Fatih Vehbi Çelebi

Yakup Sabri Özkazanç

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

RELEVANCE FEEDBACK AND SPARSITY HANDLING METHODS FOR TEMPORAL DATA

Bahaeddin ERAVCI
Ph.D. in Computer Engineering
Advisor: Özgür Ulusoy
Co-Advisor: Hakan Ferhatosmanoğlu
July 2018

Data with temporal ordering arises in many natural and digital processes with an increasing importance and immense number of applications. This study provides solutions to data mining problems in analyzing time series both in standalone and sparse networked cases. We initially develop a methodology for browsing time series repositories by forming new time series queries based on user annotations. The result set for each query is formed using diverse selection methods to increase the effectiveness of the relevance feedback (RF) mechanism. In addition to RF, a unique aspect of time series data is considered and representation feedback methods are proposed to converge to the outperforming representation type among various transformations based on user annotations as opposed to manual selection. These methods are based on partitioning of the result set according to representation performance and a weighting approach which amplifies different features from multiple representations. We subsequently propose the utilization of autoencoders to summarize the time series into a data-aware sparse representation to both decrease computation load and increase the accuracy. Experiments on a large variety of real data sets prove that the proposed methods improve the accuracy significantly and data-aware representations have recorded similar performances while reducing the data and computational load. As a more demanding case, the time series dataset may be incomplete needing interpolation approaches to apply data mining techniques. In this regard, we analyze a sparse time series data with an underlying time varying network. We develop a methodology to generate a road network time series dataset using noisy and sparse vehicle trajectories and evaluate the result using time varying shortest path solutions.

Keywords: Time Series, Relevance Feedback, Diversity, Autoencoder, Sparsity, Time-Varying Graphs.

ÖZET

ZAMANSAL VERİLER İÇİN İLGİLİLİK GERİ BİLDİRİMİ VE SEYREKLİK ELE ALMA METOTLARI

Bahaeddin ERAVCI
Bilgisayar Mühendisliği, Doktora
Tez Danışmanı: Özgür Ulusoy
İkinci Tez Danışmanı: Hakan Ferhatosmanoğlu
Temmuz 2018

Zamansal ilişkiye sahip veriler, artan önemi ve çok sayıda uygulaması ile birçok doğal ve dijital süreçte ortaya çıkar. Bu çalışma hem tek başına hem de seyrek ve ağ ilişkili durumlarda zaman serilerinin analizinde veri madenciliği problemlerine çözümler sunmaktadır. İlk olarak kullanıcı derecelendirmesine dayanan, yeni zaman serisi sorguları oluşturarak zaman serisi veri depolarını taramak için bir yöntem geliştirdik. Sonuç kümesi, ilgililik geri bildirim mekanizmasının etkinliğini arttırmak için çeşitlilik içeren seçim yöntemleri kullanılarak oluşturulmuştur. Geri bildirime ek olarak, zaman dizisi verilerinin benzersiz bir yönü göz önünde bulundurularak, el ile yapılabilecek bir seçimin aksine, kullanıcı açıklamalarına dayanan çeşitli dönüşümler arasında üstün performans gösteren temsil türüne yakınsamak için temsil geri bildirim yöntemleri önerilmektedir. Bu yöntemler, her bir temsil başarımına göre sonuç kümesinin bölümlendirilmesine ve birden çok temsilin farklı özelliklerini kuvvetlendiren bir ağırlık yaklaşımına dayanmaktadır. Daha sonra, hem işlem yükünü azaltmak hem de doğruluğunu arttırmak için, zaman serilerini veri bağımlı seyrek temsillere indirgemek için oto-kodlayıcıların kullanımını önermekteyiz. Çok çeşitli gerçek veri kümeleri üzerinde yapılan deneyler, önerilen yöntemlerin doğruluğu önemli ölçüde geliştirdiğini kanıtlamakta olup veriye duyarlı temsiller, verileri ve hesaplama yükünü azaltırken, benzer başarım kaydetmiştir. Daha zorlu bir durum olarak, zaman serisi veri kümesi noksan olup, veri madencilik tekniklerini uygulayabilmek için interpolasyon yaklaşımlarına ihtiyaç duyulabilir. Bu bağlamda, zamana bağlı değişen bir ağ ile ilişkili seyrek bir zaman dizisi verisini analiz ediyoruz. Gürültülü ve seyrek araç izlerini kullanarak bir yol ağı zaman serisi veri kümesi oluşturmak için bir metodoloji geliştirdik ve en kısa yol çözümlerini kullanarak değerlendirdik.
Anahtar sözcükler: Zaman Serileri, İlgililik Geri Bildirimi, Çeşitlilik, Otokodlayıcı, Seyreklik, Zamanla Değişen Çizgeler.

Acknowledgement

First and foremost, I gratefully thank my supervisors Özgür Ulusoy and Hakan Ferhatosmanoğlu for their suggestions, supervision, and guidance throughout the development of this thesis and my research career. I feel very fortunate for the opportunity to have them as my research advisors. I am indebted to my thesis monitoring committee members Enis Çetin and Uğur Güdükbay for their valuable comments and discussions enriching my studies. I also thank the jury members for their time and helpful remarks.

I also thank ASELSAN for the assistance throughout my studies as my employer and for giving permission to pursue my goals.

Last, but by no means the least, I would like to thank my family, especially my beloved wife, Sema, for her exhaustless encouragement and support throughout the ups and downs of the life journey. I also dedicate this Ph.D. thesis to her and my lovely boy, Mustafa Bilge who is the joy of our life. I appreciate all their patience during my studies.

Contents

List of Figures

List of Tables

List of Publications

This dissertation is based on the following publications:

P.1 Bahaeddin Eravci and Hakan Ferhatosmanoglu, "Diversity based relevance feedback for time series search", *Proceedings of the VLDB Endowment (PVLDB)*, v. 7(2), p.109-120, 2013 (also presented in VLDB 2014, 40th International Conference on Very Large Data Bases, Hangzhou) `https://doi.org/10.14778/2732228.2732230`

P.2 Bahaeddin Eravci and Hakan Ferhatosmanoglu, "Diverse Relevance Feedback for Time Series with Autoencoder Based Summarizations", *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2018, `https://doi.org/10.1109/TKDE.2018.2820119`

P.3 Elif Eser, Furkan Kocayusufolu, Bahaeddin Eravci, Hakan Ferhatosmanolu, Josep L Larriba-Pey, "Generating Time-Varying Road Network Data Using Sparse Trajectories", *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, p.1118-1124, 2016, `https://doi.org/10.1109/ICDMW.2016.0161`

Chapter 1

Introduction

Temporal data is common in many applications ranging from finance to healthcare and practically in any process that records data with respect to time. A time series can be defined as an array of numbers with temporal association within its elements. Generation and storage of time series data, in parallel with other data types, has reached a speed that was not possible to this day. Accumulation of such data is also gaining momentum with new technologies, such as the decline in the price and the miniaturization of different sensors (pressure, temperature, inertial, etc.) and is expected to further increase with Internet of Things (IoT) applications. In addition to one dimensional time series data, there is also a recent rise in temporal data with an underlying network structure. We can present examples of time varying graphs with time series data coupled with the related spatial information of the sensor or time varying properties of different social network users. This vast amount of data is expected to be translated into insight and knowledge by browsing, exploring and extracting application oriented information while taking temporal and network based relations into account.

Mining time series fall into several categories such as pattern recognition, classification, clustering, and forecasting, and generally starts with identifying a representation that harmonizes the end purpose of the application and the property of time series data. One of the important tasks in time series mining

is browsing, where users seek "similar" time series from a database via query by example, such as seeking online advertisements with similar view patterns for a period of time, stocks that are related in terms of price or return, cities with similar earthquake timing patterns. The volume of the time series data impedes users ability to observe and analyze the whole database and find the similar series manually.

Even though time series search and retrieval methods have been studied widely, there is limited work in utilizing relevance feedback (RF), which has enjoyed a great deal of attention in areas of text and information retrieval. In RF, a set of data items is retrieved as a result of an initial query by example, and presented to the user for evaluation. The informal goal is to present the right set of initial results that maximize the information gain from the user feedback.

To the best of our knowledge, no prior work considered improving time-series retrieval and relevance feedback (RF) by introducing diversity in the search results before our work [2]. This study enriches machine learning and information retrieval ideas of related nature by adapting it to time series using representations that capture the temporal relations within the data in conjunction with diverse retrieval. We study RF for retrieval in time series databases and adopt a query by example approach where the user submits a time series query reflecting the user intentions. Based on the initial query, a set of time series items according to a criterion is retrieved from the database and presented to the user for evaluation. The problem is to present a set of items to collect feedback in such a way that maximizes the information from the evaluation process, as opposed to the top matching set, correctly modeling and identifying the user intent for the subsequent retrieval iterations. The user evaluates and annotates the presented items to improve the results in the next round for finding the time series item of interest. We utilize different representations of time series, and use diversity between time series data in different rounds of retrieval process to further enhance the user satisfaction by increasing the number of relevant items in the resultant sets.

Given the large amount of research for identifying representation methods in time series literature, finding the right representation for one's intent is an important task. The already available user feedback in RF iterations can be used identify the appropriate representation for a query or an application in this context. We develop a representation feedback method in which the initial result set is populated with items from differing representations, and the system identifies the appropriate one by analyzing the annotated items. The user feedback is utilized to converge to the representation which satisfies the user most by increasing the total number of items from the best performing representation for the next rounds of retrieval. This can be useful in dynamic databases where the properties of the data are changing or with systems serving multiple user groups with different objectives each satisfied using a different representation. Experimental results show that besides the intended use as representation feedback, the method implicitly embodies item diversity as well [2].

We expand the framework by using autoencoder neural networks to extract sparse representations of the time series that both reduce the data size and extract more appropriate representations. Autoencoders can also be effective in combining multiple representations and selecting relevant features from best performing representations by analyzing the dataset. Our study with experiments spanning a diverse set of data aims to assess the potential of autoencoders use for the general time series data even with relatively simple network structures. The use of similar deep network techniques to learn data-aware representations which can suit time series data generally by analyzing huge diverse time series datasets can be an important tool in time series data mining. These general networks can also be trained to specific tasks, e.g. stock analysis where stock movement patterns are identified with years of human expertise [3].

Considerable amount of time series data mining methods accept complete and standalone time series data. As a different aspect of temporal data, there are different applications where graphs with time varying properties arise naturally. These types of time series datasets can also be incomplete due to deficiency of sensors/recorders in various nodes of the network. Management and analysis of

road networks is an example which is an important task for traffic management and location based services (LBSs).

Road networks are spatial graphs with vertices representing the geo-locational points and edges representing the roads between theses vertices. Most traditional graph algorithms assume that the edges have static attributes, such as the length of the road and the speed limit. A more enhanced and correct representation is a time-varying road network that models the changing traffic information via graph with each edge having a time-series attributed, rather than a single aggregate static value.

In this perspective, we analyze the trajectories found in the Floating Car Dataset of Telecom Italia [4] and generate a realistic time-varying graph dataset with different travel times for each time slot for its edges [5]. This real data set consists of sparse and noisy GPS trajectories collected from different vehicles. After mapping the trajectories to the road network, we use time series analysis and inference methods to estimate the missing values to generate a complete time-varying graph. The initial data from the mapping process consists of a very sparse time series data for each edge in the road network and to interpolate missing data points, the frequency content of nearly-complete edges is identified using spectrum analysis methods. We notice that the time series of interest are band-limited and most of the signal power is in the low-pass region, i.e., the time series are slow varying signals. This information is used as a model to complete the missing data using minimum travel time for the respective edge and random sample drawn from a normal distribution with parameters devise from the signal itself. These two parameters are selected according to Nyquist-Shannon sampling theorem to counter aliasing. After padding the signal, we use the fact that the signals are expected to be band-limited to smooth the padded time series such that the resultant signal is of the same model as the expected signals. The resultant complete time series dataset with the underlying road network is stored on Sparksee [6] graph database and a Java API is developed for easy manipulation of the graph.

1.1 Main Contributions

The contributions of this dissertation include the following:

- We analyze and utilize different time series representations useful for RF to capture a variety of global and local information. Spectrum based transforms (Fast Fourier Transform and Dual-tree complex wavelet transform) are considered due to its power to identify patterns localized both in time and frequency domain and SAX-based transform is used to identify time based patterns which can be essential for diversity based RF. We construct a RF technique for time series by tuning such systems without the need of explicitly defining features like amplitude shift, periodicity, etc.
- The performance of RF is enhanced by using diversity in the result set improving the effectiveness and utilization of the user annotation for additional improvements in precision.
- Two different on-the-fly representation selection mechanisms that choose best performing representation types by further exploiting the valuable feedback from the user are proposed.
- We study the use of autoencoders to decrease the complexity of the overall features and extract useful data aware features. A representation map is learned from the data and can be used in other time series tasks. The presented approach exploits the advantages of RF and diversification, and illustrates a potential use of autoencoder type networks in time series retrieval.
- A rich set of experiments on 85 real data sets provide insights on the feasibility of the general RF framework, performance of autoencoders and how diversity can improve time series retrieval. We discuss the performance of the developed methods under different data properties instead of specific applications and analyze advantages of the methods with respect to different cases.

- We apply time series analysis and interpolation methods to generate a time-varying graph for road networks using sparse and noisy time series. This process begins with analyzing the nearly-complete time series to create a model of the series. Using this model, we generate the incomplete data points to form a realistic full time series dataset for the network.

- To develop and validate algorithms/systems and for analysis of traffic for dynamic road networks, the research community needs publicly available time-varying graph datasets with edge weights varying over time based on realistic patterns. Our generated time varying graph dataset and an associated graph management tool is shared to support research on related topics.

Experimental results show 0.23 point increase in precision averaged over all four representations, with 0.48 point increase in specific cases in the third round of RF. Introducing diversity into RF increases average precision by 6.3% relative to RF with no transformations and 2.5% relative to the RF using the proposed representation. Results also show that the representation feedback method implicitly incorporates item diversity and converges to the better performing representation, confirming it to be an effective way to handle changing data properties and different user preferences.

Experiments with the autoencoders present runtime performance increases of around 6-9x due to reduced total data volume with a mild degradation in the average precision. We have also observed that in some challenging data cases, where the precision is low, the accuracy improves when using autoencoders, which is encouraging to further pursue this approach.

We performed experiments using the generated time varying graph dataset by employing the system in a use case, evaluating the time-varying shortest path solution. It is observed that the difference between the shortest paths using time-varying versus static weights increases as the number of vertices increases. The travel durations of these two types of shortest paths give differing results favoring the use of the time-varying road network.

1.2 Outline

The rest of the dissertation is organized as following: A literature review presenting the background information for the different components of the dissertation is given in Chapter 2. The general time series RF framework with its specific application and results is defined and presented in Chapter 3. We also provide intuitional and statistical justification for use of diversity in RF frameworks in this chapter. Chapter 4 details the extension of the proposed RF framework with representation feedback and autoencoders. We detail the method to generate a complete time varying graph from a sparse time series data with an application on road networks in Chapter 5. Finally, we conclude the dissertation in Chapter 6 with outlining possible future work.

Chapter 2

Related Work and Background

We present the current literature and its relations with the different aspects of the proposed methods in this chapter. We first give the general literature review of time series representations and different similarity measures used for various purposes which is related with the proposed time series RF setting. We follow on by noting the important research on relevance feedback presenting the different perspectives from information retrieval to machine learning. Subsequently, diversity related studies and its use in different contexts are presented which contributes as an important aspect of our proposed methods. Since we develop a retrieval framework ranking different time series items, we also discuss ranking algorithms each tackling the problem in a different aspect. We include a concise discussion of autoencoders focusing on its use in time series literature. Finally, we present the background for time-varying graphs with a brief introduction to route planning literature and discussion of different datasets already available in the literature.

Time series data mining research has immense literature on the representation of the time series, similarity measures, indexing methods, and pattern discovery ([7]). Generally, time series data mining tasks begin with identifying a representation that links the end purpose of the application and the properties of the time series. Various representations have been proposed to transform the

time series, each with a different perspective to meet the requirements of different applications, user intents, and data properties. A class of representations, such as piecewise aggregate approximation (PAA) and symbolic aggregate approximation (SAX) [8], are used to identify features in the time domain, while others, including discrete Fourier transform (DFT) and discrete wavelet transform (DWT), involve frequency domain properties dealing with the periodic components in the series.

After the time series is transformed, measures of similarity are used for comparing data items. A multitude of similarity measures (from L_p norms to dynamic time warping (DTW), etc.) has been proposed [9, 10, 11]. Indexing methods for similarity queries have also generated extensive interest in the community given the computational load of the algorithms [12]. Besides using geometric distance on coefficients [13], dynamic time warping (DTW) and other elastic measures are used to identify similarities between time series due to non-aligned data [9, 14, 11]. We also see approximate subsequence matching methods proposed for large time series datasets with DTW distance in [15]. We report recent work that proposes a relational database for time series by bi-clustering the tables to allow quick retrieval under heavy loads [16]. Additionally, methods to identify recorded sound excerpts by comparing various features of the signal stored in a music database has been described in [17].

There has been significant work in information retrieval community for relevance feedback (RF) since it was proposed in the 1970s ([18, 19, 20]) and to this day there still is research on variants of this method or mix of this method with different techniques ([21]). The first methods have concentrated on RF query movement in which the query point was moved toward the relevant items. Dimension weighting methods have been proposed for the same objective in [22]. There has also been use of RF in the image and multimedia retrieval applications [23, 24, 25]. Lately, researchers pose the RF problem as a classification problem and propose solutions in the context of machine learning [26, 27].

Combining relevance and diversity for ranking documents has been studied by [28] in the context of text retrieval and summarization. They define a Maximal Marginal Relevance (MMR) objective function to reduce redundancy while still

maintaining relevance to the query in re-ranking retrieved documents. There are recent studies which analyze MMR type diversification and provide efficient algorithms for finding the novel subset [29]. The problem of ambiguity in queries and redundancy in retrieved documents has been studied in [30]. They propose an evaluation framework and emphasize the importance of objective evaluation functions. Researchers study diverse results in web search application by posing the problem as an expectation maximization in [31]. A retrieval method for maximizing diversity, which also assigns negative feedback to the documents that are included in the result list is proposed by [32]. Given the considerable success of applying diversity in retrieval in different domains, [33] proposes scalable diversification methods for large datasets using MapReduce. Studies on using relevance, diversity and density measures to rank documents in information retrieval within an active learning setting have also found interest in the literature [34]. Diverse results have been reported to increase user satisfaction for answering ambiguous web queries [35] and for improving personalized web search accuracy [36]. Graph based diversity measures for multi-dimensional data have also been proposed in [37]. Methods to find the best representative of a data set based on clustering has been investigated in [38].

Top-k retrieval has been studied also as a machine learning problem to rank documents according to user behavior from analyzing implicit feedbacks like click logs. A Bayesian based method is proposed as an active exploration strategy (instead of naive selection methods) so that user interactions are more useful for training the ranking system [39]. A diverse ranking for documents is suggested to maximize the probability that new users will find at least one relevant document in [40]. There is recent interest to address the biases (e.g., presentation bias where initial ranking strongly influences the number of clicks the result receives) in implicit feedbacks using a weighted SVM approach [41]. We also note some studies concentrating on ways to balance diversity and relevance while learning ranking of documents [42] [43]. One can approach the result set selection problem using active learning where the main aim is to label parts of the dataset as efficiently as possible for classification of any data item in the dataset. There is a variety of techniques each with a different perspective such as minimization of

uncertainty concerning output values, model parameters and decision boundaries of the machine learning method [44].

Contrary to its popularity in the information retrieval, RF and diversity have not attracted much attention in time series community. Representation of time series with line segments along with weight associated to the related segments and explicit definition of global distortions have been used in time series relevance feedback [45, 46]. We are not aware of any studies using representation feedback for time series retrieval and diversification in such systems before our initial paper on the issue.

Autoencoder neural networks formulate an unsupervised learning that uses the input data as the output variable to be learned [47]. The network structure and the training objectives force the outcome to be a sparse representation of the input data. It has attracted a renewed interest lately with deep network approaches generally utilizing restricted Boltzmann machines [48]. We also note some recent work on time series visualization utilizing autoencoder structures [49]. Time series forecasting with neural networks is reported to be advantageous even with relatively small data cases in [50].

Incomplete time series is generated and has been studied under different contexts. Tormene et.al. proposed a variant of DTW to match truncated time series in [51] within a medical application. There is also interest in accurate calculation of spectral content with time series of missing content with applications in seismology [52]. Researchers have also focused on analyzing sparse time series data due to an increase in number of sensors/apps recording temporal data bursts. One study [53] focuses on prediction of power consumption at electric vehicle charging stations using nearest neighbor methods while another paper aggregates the sparse data in different time windows and uses weighing methods to estimate free parking spaces in an urban setting [54].

Road networks and route planning has generated a great deal of interest due to increasing mobility with increasing use of navigation systems and online route planning services [55]. Computation of shortest paths over time-dependent road

networks is shown to be polynomially solvable with Dijkstra based solutions adapted for these types of graphs [56, 57]. Route planning over time-dependent graphs has also appeared in the literature aiming at reducing traffic jams [58]. As an increasing trend, personalized route planning emerges by considering driver's preferences [59].

Major service providers such as Google, Yandex, and TomTom have the ability to observe real time traffic in certain regions and can update their underlying road network's edge weights accordingly. However, most users and researchers do not have access to such dynamic updates. The datasets used in the literature are usually combinations of real maps with synthetically generated travel time-series [56] or real data collected over a limited amount of time [58] [59]. This makes the comparison of algorithms difficult because they usually have data dependencies or these datasets are not publicly available via an API or web service.

Employing GPS traces to build or exploit road networks has also been studied in the literature [60], [61]. In [60], GPS data are utilized to generate a road network without any prior information about the network topology. In [61], GPS data are employed with a road network for traffic and travel time estimation of the paths by using probabilistic model based approaches.

Chapter 3

Diverse Relevance Feedback for Time Series

This chapter presents the proposed framework for relevance feedback (RF) using diversity amongst its result sets increasing the accuracy of retrieving similar items from a time series database given a user specified query. Application of this framework from different perspectives include the following:

- finding products with similar sale patterns in online commerce with respect to a selected product
- identifying electrocardiography signals from past patients correlated with a specific patient
- finding network nodes with similar communication loads extracted from network logs

In each of these applications, the user queries the database and seeks relevant time series items according to the specific application and intent.

3.1 Problem Definition

We consider a database, $TSDB$, of N time series: $TSDB = \{TS_1, TS_2, ..., TS_N\}$. Each item of $TSDB$: TS_i, is a vector of real numbers which can be of different size, i.e. $TS_i = [TS_i(1), TS_i(2), ..., TS_i(L_i)]$ where L_i is the length of a particular TS_i. Given a query, TS_q (not necessarily in $TSDB$), the problem is to find a result set (a subset of $TSDB$) of k time series that will satisfy the expectation of the user. Since we formulate the solution in an RF setting, the user is directed for a binary feedback by annotating the items in the result set as relevant or irrelevant.

3.2 Relevance Feedback Framework

RF is an important tool in information retrieval to increase user satisfaction where the user is given a set of relevant items in each iteration and is expect to evaluate and annotate the relevance of each item presented by the system. A feedback mechanism is established where items that are more relevant are presented in the successive rounds. The basic model is given in Figure 3.1. Each component of the system shall be explained in the successive sections.

The framework proposed dictates the transformation of the time series, as a preprocessing step, into the preferred representation (CWT, FFT, SAX, PAA etc. according to properties of the time series in the database and the application requirements) such that the relevant features are captured. The preprocessing may also involve a normalization procedure (unit-norm, zero-mean, etc.) as necessary. Given a user provided initial time series query (TS_q), the relevant transformation is applied and a transformed query vector, q, is calculated which will be used in the similarity calculations and top-k retrieval process. T_i denotes the transformed TS_i according to the transformations explained in the previous sections, i.e., $T_i = \mathcal{F}(TS_i)$.

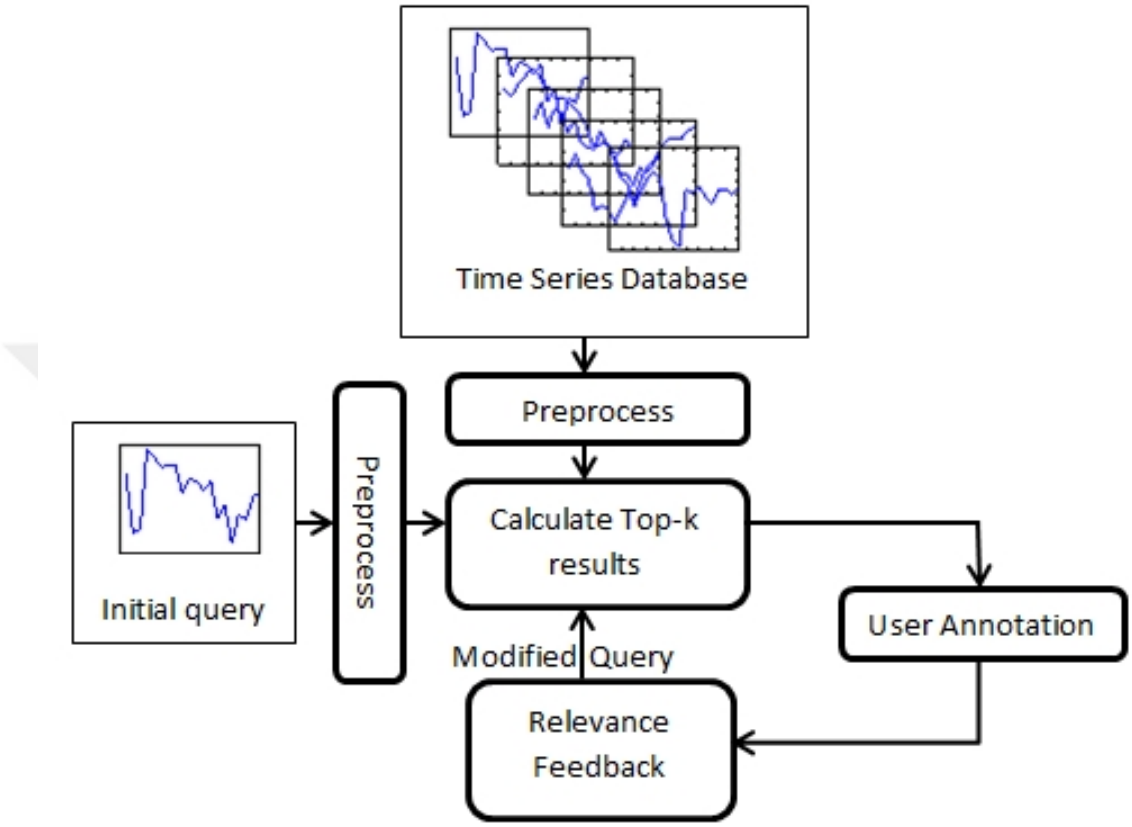


Figure 3.1: Relevance feedback system

3.3 Time Series Representation

The effectiveness of representation to decode the user intention is essential for the performance of time series similarity match. The appropriate representation depends on the application, time series properties and user intent. For example, if the user intention is to figure out all the time series with specific periods like weekly patterns, frequency domain approaches like DFT (Discrete Fourier Transform) can serve the purpose. An important general property to consider is the shift-invariance of the transform. This will allow correct retrieval even if time series pairs have offset in time. Accurate and easy handling of time series with varying length is another important criteria for the representation choice. Transforms that help compare local and global properties of time series items would be expected to be functional for diversity based browsing. Based on these

observations, we focus on different representation methods and approaches: based on Wavelet Transform and based on SAX (Symbolic Aggregate approXimation in [62]), in addition to Fast Fourier Transform (FFT), and the raw time series as a baseline in our experiments. FFT is a computationally optimized version of DFT with the same numerical outputs. Experiments with four different representations provide insights on how different types of representation work with RF and how they affect the precision for different datasets.

On one side, we will illustrate our approaches using these methods; and on another we will evaluate the appropriateness of these successful representations and provide insights on their use in our RF and diversity context.

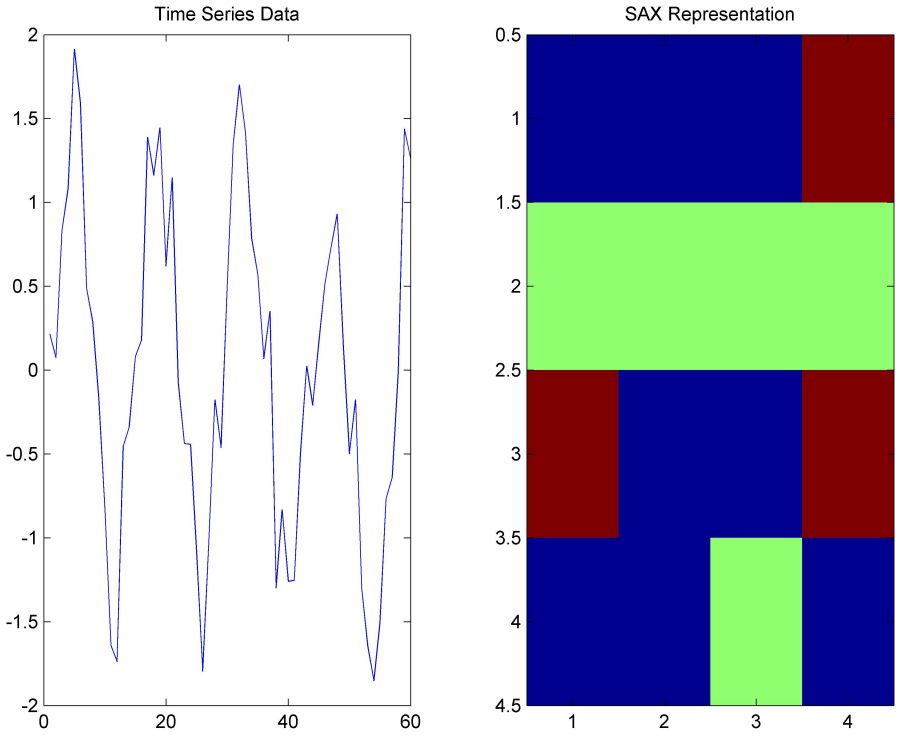


Figure 3.2: An example SAX bitmap representation

SAX has gained a prominent place in the time series research community due to its success in representation. SAX transforms the time series into a string of elements from a fixed alphabet which gives the ability to exploit different techniques already found and used in string manipulation. After the SAX

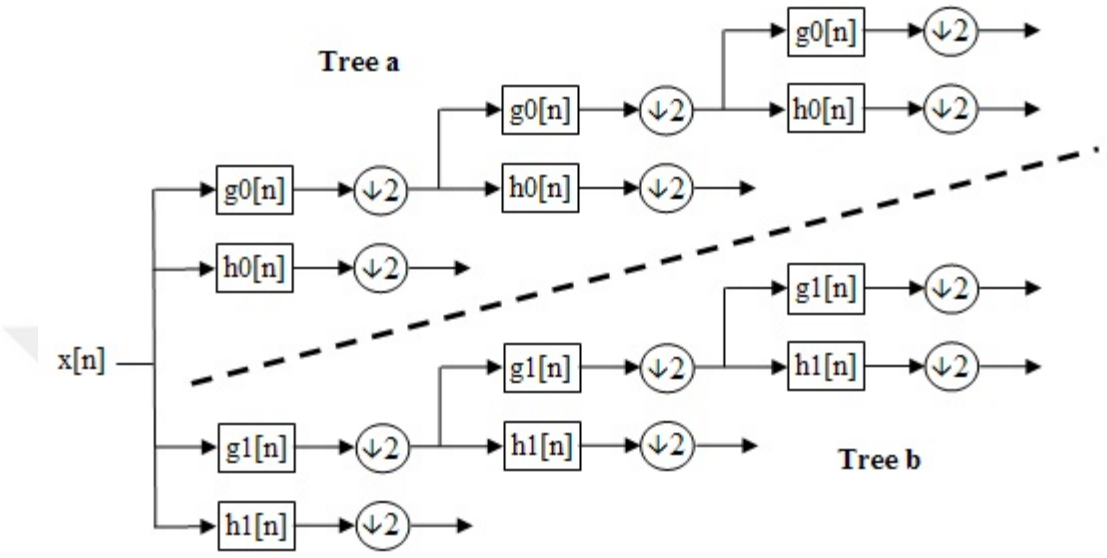


Figure 3.3: Three level Dual-Tree Complex Wavelet Transform [1]

transformation of the time series into a string, a post-processing method called SAX-bitmap is utilized which turns the string into a matrix (SAX-bitmap image in the visualization context) by counting the different substrings included in the whole string. SAX-bitmap is reported to be intuitive, useful in extracting important sub-patterns in the time series and is a perceptually appropriate representation for humans in visualizing and interpreting time series ([62]). In our context, we use it as a transformation of the time series to a vector which is then used with different distance measures for similarity retrieval. The method effectively counts the number of different local signatures after transforming the original time series to SAX representation. The level of the representation (L) corresponds to the length of the local patterns in the SAX representation. The length of the output of the SAX-Bitmap transform is M^L where M is the number of symbols used in the SAX transformation, which is independent of the time series length (L_i).

SAX inherently divides the time series into blocks and normalizes the block within itself which extracts local features of the time series useful for diverse retrieval methods. The total number of occurrences in the whole time series gives information about the global features as well.

Wavelet Transform (and its variants Discrete WT, Continuous WT, Complex WT etc.) is a time-frequency representation used extensively in time-series domain. The transformed data (scaleogram) provides a frequency and time localization. The level of the representation (L) in CWT corresponds to the height of low pass components of CWT which in turn corresponds to different details of the low pass and high pass components. The upper part of the tree is the real part of the transform and the lower part of the tree is the complex part given in Figure 3.3. Dual-tree complex wavelet transform (named due to two parallel filter banks in the process) is relatively shift-invariant with respect to other flavors of the algorithm which is a reason behind its selection for this study [63]. The magnitude of the complex and real part is used in this paper. The length of the transformed data is independent of the number of levels and is given by $2^{\lceil \log_2 L_i \rceil}$.

CWT has a similar approach with SAX but with a different perspective. CWT extracts some low-pass features, i.e., components which are in the lower frequency band and are relatively slowly varying giving an averaged version of the overall series and high pass features, i.e., components which are in the higher frequency band and are relatively fast varying, related to detail and differential information of the series. Down-sampling of the series along the branches allows the transform to extract information from different zooms of the data. As a summary, CWT decomposes the time series into local patterns in both time and frequency with different scales and can help for diversity as different subsets of the information given by the transformation provide different perspectives of the data.

3.4 RF with Diverse Top-k Retrieval

RF techniques inherently model the distribution around the query point with a limited number of user annotated data items to increase the accuracy subsequently. After each iteration of RF, the user is given the opportunity to evaluate the resultant items presented by the system. A variety of different techniques can be utilized for the feedback mechanism, such as Rocchio's algorithm ([18]) which

moves the query point in space closer to the relevant items. We have selected this algorithm since it is one of the foremost algorithms used in information retrieval for RF and is still considered with different variants as an important method in recent studies [21, 64, 65]. We have adapted a modified version of Rocchio's algorithm. Rocchio method forms an additional query using the relevant and irrelevant items for successive rounds of RF. Equation 3.1 details the procedure where Rel is the set of items classified relevant, $Irrel$ is the set of items classified as irrelevant by the user.

$$q_{new} = \frac{1}{|Rel|} \sum_{i=1}^{|Rel|} Rel_i - \frac{1}{|Irrel|} \sum_{i=1}^{|Irrel|} Irrel_i \tag{3.1}$$

Newly formed query vector is not dependent on the original query but the original query affects the results via Equation 3.2 since the system uses the original query with the newly formed query vectors in the previous RF stages to calculate the distances. We also experimented with a Rocchio algorithm which directly replaces original query at each iteration and found that the modified version explained above performs better. For the successive iterations, a distance is calculated with respect to all the query points of the previous iterations which is outlined in Equation 3.2. The high level algorithm for RF system is shown in Algorithm 1.

$$Dist(q_1, q_2, ..., q_N, T_{test}) = \frac{1}{N} \sum_{i=1}^{N} Dist(q_i, T_{test}) \tag{3.2}$$

where N is the RF iteration number

The RF method at its essence forms new queries (or modifies the initial query depending on the various implementations of the idea) and uses this new query in the next iterations for retrieving the similar time series items in the database. The main objective is to move the query vector closer to where more relevant objects are expected.

Top-k retrieval part of the RF system identifies k time series to be presented to the user who is seeking information relevant to the query, q. The general

Algorithm 1 High-level algorithm for diverse relevance feedback

```
Initialize k : number of items in result set
Initialize RFRounds : number of RF iterations
Initialize λ = [λ_1, λ_2, ..., λ_RFRounds]: MMR parameters
Initialize α = [α_1, α_2, ..., α_RFRounds]: CBD parameters
Input q_1 : initial query (transformed if needed)
Input TSDB : time series database (transformed if needed)
for i = 1 → RFRounds do
    // Find Top-k results
    if Nearest Neighbor then
        R = Top-K(q_1,...,q_i,k,TSDB)
    else if MMR then
        R = Top-K_MMR(q_1,...,q_i,k,λ_i,TSDB)
    else if CBD then
        R = Top-K_CBD(q_1,...,q_i,k,α_i,TSDB)
    end if
    // User annotation of the result set
    (Rel,Irrel) = User Grade(R)
    // Expand query points via relevance feedback
    q_{i+1} = Relevance_Feedback(Rel,Irrel)
end for
```

method often used is to find the k-nearest neighbors which is a list of time series ranked according to a defined distance function with respect to q. The traditional assumption in this similarity model is that the data points closest to the query, irrespective of direction and non-circular user intent, is related to the user preference. However, there can be data points close to the query in theoretical sense yet not related to the interest of the user. Moreover, the intent of user can be already ambiguous itself or the query point may also be on the boundary of the user intent.

In the above explained case, as the name nearest neighbor (NN) implies, only the data points in the vicinity of the query point are retrieved with rankings proportional to closeness of the data to the query. But the database may include numerous time series items very similar to each other which can degenerate the top-k list to an item list with very little variation. This degenerate top-k list will give very limited novel information about the user intentions since q is already known making RF less useful and wasting the time and annotation effort of the

user.

As a solution to the above issue, the user needs to be presented a top-k list with diversity among items which are still close to the query point. With a balanced diverse set of choices provided, the successive iterations of RF is expected to better meet the user intentions.

We present an example to illustrate the mechanics of the query relocation in Figure 3.4 and to discuss the potential advantages of utilizing diverse results. We have plotted three different classes of data (using three normal distributions with different means, each representing a user's or a group of users' interest) and queries of two extreme cases: Q_1 query on the boundary in terms of user interests and a Q_2 query near to the main relevant set. These queries are moved to revised points (or the effect of using new queries translates to this effect via Equation 3.2) Q_1' and Q_2' given that Data2 and Data3 are considered relevant by the user respectively. If nearest neighbor (NN) retrieval is used, the result can be a degenerate list with too little variation and limited information about the user intentions since Q_1 and Q_2 are already known. If one provides a larger radius around the query, which samples the region around the query, the displacement of the vectors from their original location will be higher. On the other hand, over-diversification of the results, causing very few relevant items finding a spot in the result set, will hinder and lower the accuracy of the next phase of user annotation. A probabilistic explanation to this intuition is also provided in Section 3.5.

We consider two different methods to diversify the top-k results in this study: maximum marginal relevance (MMR) and cluster based diversity. MMR (Algorithm 1 Line 12) merges the distance of the tested data item to both the query and to the other items already in the relevant set. The diversity achieving distance used is given in Equation 3.3 and a greedy algorithm is used until a specific number of items is selected from the whole dataset. $Dist$ function is general and can be any distance function of choice. When λ is chosen as 1, the $DivDist$ collapses to an NN distance and the result turns to a mere top-k nearest neighbor result. When λ decreases, the importance of the distance to the initial query decreases which gives an end result of diverse set of items within itself but

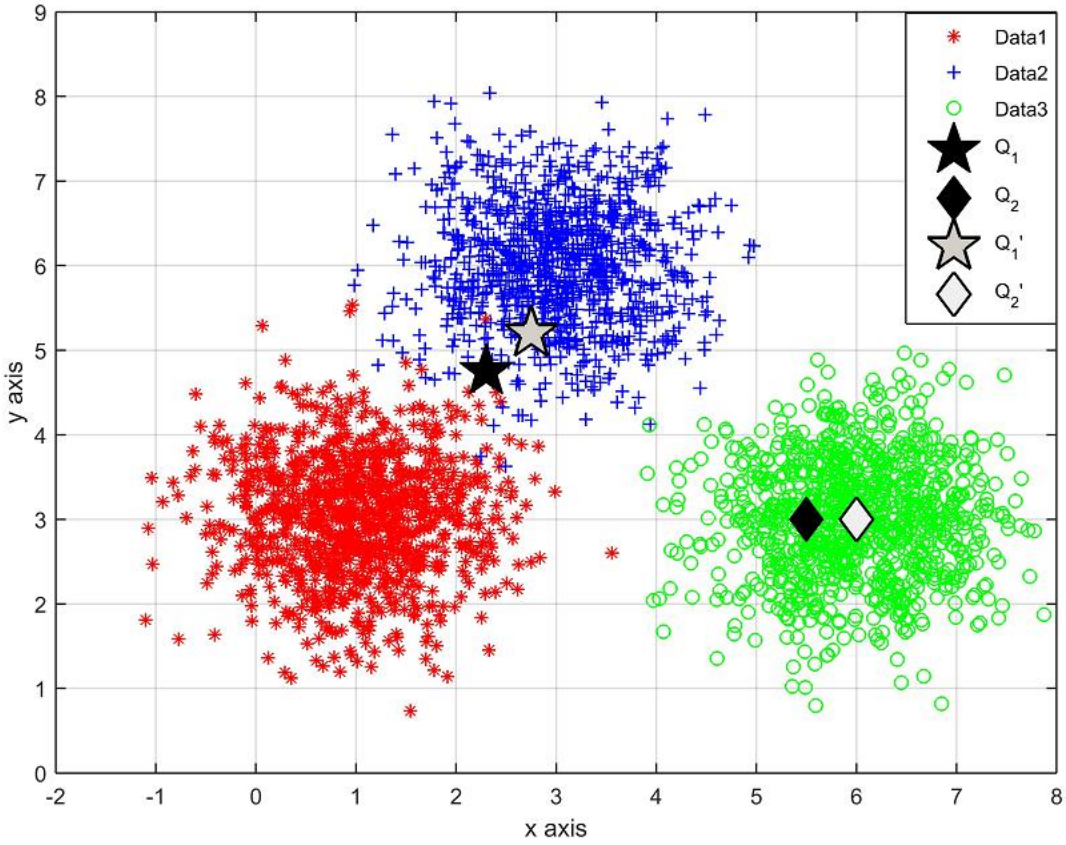


Figure 3.4: An example case of data and query movement with Rocchio based algorithm

are also related to the query.

$$DivDist(T_q, T_i, R) = \lambda Dist(T_q, T_i) - \frac{1}{|R|}(1-\lambda)\sum_{j=1}^{|R|} Dist(T_i, T_j) \tag{3.3}$$

The second term of the *DivDist* involves pairwise comparisons of data points in the database which is independent of the query and is performed repetitively for each query. To decrease the running time of the algorithm, we use a look-up table that stores all the possible pairwise distances calculated offline once at the beginning for the particular database.

Cluster Based Diversity (CBD) method uses a different approach than the optimization criteria as given in Equation 3.3. The method is inspired by [38]

which proposes method for finding best representatives of a data set. This method (Algorithm 1 Line 14) retrieves Top-αk elements ($\alpha \geq 1$) with a NN approach and then clusters the αk elements into k clusters. The parameter α controls the diversity desired, increasing α increases the diversity of the result set. If α is chosen as 1 then the results are the same as the NN case. We implement a k-means algorithm for the clustering phase in this study. The data points nearest to the cluster centers are chosen as the representative retrieved points which are presented to the user. An advantage of CBD is that the tuning parameter α is intuitive and the results are predictable.

We also notice that precision increase due to use of diversity depends significantly on the data distribution. Since the diversification algorithms are based on a distance directly or indirectly, the particular meaning of the features is important for the overall performance. This is especially important in the case of time series, as we have an autocorrelation within the elements. This observation stresses that the representation of the time series which is chosen according to its power in decoding general user intentions and its suitability for varying properties of time series with respect to different applications, should also have the power of decomposing time series into meaningful parts which have novel information. These properties have been considered in choosing the suitable representation.

The method for diversification can be tailored with respect to the methods used for searching and learning the user feedback. As an example, one can also utilize a support vector machine (SVM) based binary classifier to learn the relevant/irrelevant sets instead of the distance based ranking method used in this study. In this case providing the cluster centers as in the proposed CBD method will perform poorly since SVM classifier tries to learn the boundaries of the classes and requires the instances around the border regions for optimal performance. Hence, It will be better to sample the boundaries and form those type of queries in the SVM case. Whereas, NN-like distance based models, similar to the one considered in this paper, functions better if we learn the centroids of the relevant data with low uncertainty. We can expect to have better results from the CBD method which fits nicely with the distance based retrieval methods. This relation needs to be taken into account when diversity achieving methods are used and

the ranking system should be compatible with the diversification technique for satisfactory results.

3.4.1 Algorithmic Complexity

We present the algorithmic complexity of the retrieval methods in terms of N (the number of time series in database), L (the length of time series or representation), and k (the number of requested items). The NN based retrieval first calculates distances to all items in dataset ($O(NL)$) and finds k nearest items ($O(kN)$) which corresponds to a total complexity of $O(N(L+k)) = O(N)$.

For the MMR case we have two possibilities with respect to Equation 3.3:

- Without memoization: Distance calculations to all items in the dataset ($O(NL)$), distance calculations for relevant set items ($O(N \cdot L \cdot (k-1) \cdot (k-2)/2) = O(NLk^2)$, finding the minimum distance element k times ($O(kN)$) with an overall complexity of $O(NL(1+k^2)) = O(N)$.
- With memoization: Distance calculations to all items in the dataset ($O(NL)$), distance calculations from lookup table for relevant set items ($O((k-1) \cdot (k-2)/2) = O(k^2)$, finding the minimum distance element k times ($O(kN)$) with an overall complexity of $O(N \cdot (L+k)) = O(N)$ (where $NL \gg k^2$).

For the CBD case, we first find αk nearest neighbors ($O(N(L+\alpha k))$ and cluster the results. K-means clustering (based on Lloyd's which has a limit i for the number of iterations) is considered $O(NkLi) = O(NkL)$ algorithm. The total complexity for CBD case is $O(N(L+\alpha k + Lki)) = O(N)$.

3.5 Illustrative Analysis of Diverse Retrieval

We now present an illustration of the intuition behind using diversity in the RF context. Given a query, q, we retrieve a top-k list using NN with the last element d distance away from the query. Figure 3.5 illustrates the relevant set, $R \sim \mathcal{N}(0, \sigma^2)$ and the irrelevant set, $IR \sim \mathcal{N}(\mu, \sigma^2)$ assuming Gaussian distributions for both.

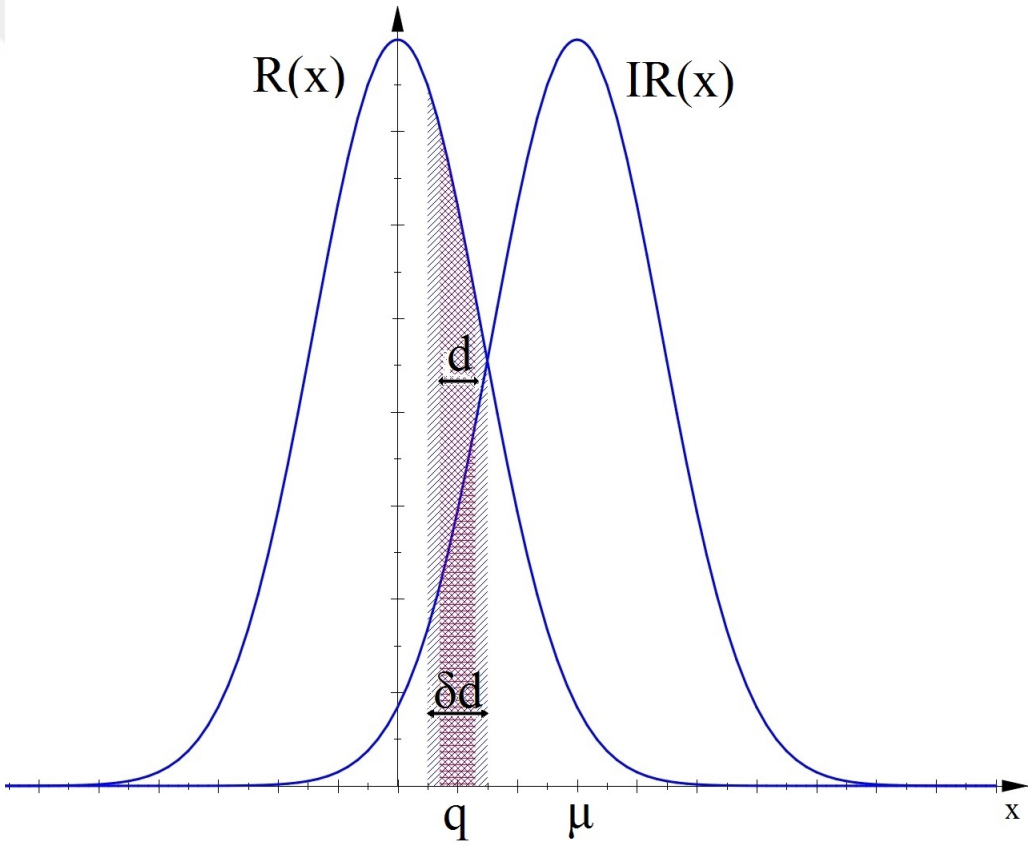


Figure 3.5: Data distributions used in analysis

If there are N relevant and M irrelevant items, we can find the number of

relevant ($k1$) and irrelevant items ($k2$) in the top-k list with approximations as:

$$\begin{aligned} k_1 &= N \cdot \int_{q-d}^{q+d} R(x)\ \mathrm{d}x \approx N \cdot R(q) \cdot 2d \quad if \quad k_1 \ll N \\ k_2 &= M \cdot \int_{q-d}^{q+d} IR(x)\ \mathrm{d}x \approx M \cdot IR(q) \cdot 2d \quad if \quad k_2 \ll M \\ k &= k_1 + k_2 \end{aligned} \tag{3.4}$$

We can then define and calculate the precision for the query as:

$$Prec(q) = \frac{k_1}{k_1 + k_2} = \frac{N \cdot R(q)}{N \cdot R(q) + M \cdot IR(q)} \tag{3.5}$$

This formula follows the general fact that if R and IR are separable (μ is very large) or if the query point is near the mean of R precision will be high. We also observe that the performance is dependent on the accuracy of the known model (i.e. the R and IR distributions) itself. We learn the model of the relevant set in the RF setting by modifying the query according to the feedback from the user. Consider a simplified RF model that forms the query for the next iteration (q_2) as the average of all the relevant items, i.e.:

$$\begin{aligned} q_2 &= \sum_{i=1}^{k_1} R_i = \int_{q-d}^{q+d} x \cdot R(x)\ \mathrm{d}x \\ &= \sqrt{\frac{\sigma^2}{2 \cdot \pi}}[\mathrm{e}^{q-d} - \mathrm{e}^{q+d}] \end{aligned} \tag{3.6}$$

A diverse set of points around q would span a larger distance (δd) around the query, which is also shown in Figure 3.5. In this case we get a modified q_2' from the relevance feedback as:

$$\begin{aligned} q_2' &= \sum_{i=1}^{k_1} R_i = \int_{q-\delta d}^{q+\delta d} x \cdot R(x)\mathrm{d}x \\ &= \sqrt{\frac{\sigma^2}{2 \cdot \pi}}[\mathrm{e}^{q-\delta d} - \mathrm{e}^{q+\delta d}] \quad \delta > 1 \end{aligned} \tag{3.7}$$

Diversity ensures $q_2' < q_2$ which increases our understanding of the relevant data distribution and consequently the query precision via Equation 3.5. If the precision is already high (i.e., if R and IR are well separated or the query is not near the R and IR boundary), then the precision increase due to diversity will not be significant.

3.6 Evaluation

We evaluate the performance of the methods with experiments on all the data (85 real data sets) currently available in the UCR time series repository [66]. The data sets used with their respective properties are provided in Table A.1.

Since we have an unsupervised application, the training and test datasets are combined to increase the size of the data sets. The numbering of the datasets in this paper is according to the numbering given in the table. Some aggregate properties of the datasets are as follows:

- Number of classes within separate data sets vary from 2 to 60
- Lengths of the time series (L) in the data sets vary from 24 to 2709
- Sizes (number of time series N) of the data sets vary from 40 to 16,637

3.6.1 Experimental Setting

We first transform all the time series data to CWT, SAX and FFT. SAX parameters are $N = L_i$, $n = \lceil \frac{N}{5} \rceil$ (meaning blocks of length 5), an alphabet of four with SAX-Bitmap level of 4. The values for L and n can be optimized for different data sets to further increase accuracy. We have experimented with several different values around the vicinity of the given values ($n = \lceil \frac{N}{6} \rceil$, $L \in [3, 4]$) for randomly selected datasets and have seen that the improvement in precision is still evident on similar scales. For the Complex Wavelet Transformation we utilized the Dual-Tree CWT implementation given in [67] with detail level $L = 5$. We used both the complex and real parts by taking the absolute value of the CWT coefficients. We performed the same experiments also on the raw time series without any modification (TS) to compare the effectiveness of the representations.

Since the objective of this study is not to find solutions for specific cases and we aim to enhance RF via diverse results for general cases, we did not fine

tune parameters, we used the same set of parameters for all the data sets for an impartial treatment.

In the experiments, we explored 5 different methods of top-k retrieval:

1. nearest neighbor (NN)
2. MMR with $\lambda = [0.5, 1, 1]$ (MMR_1)
3. MMR with $\lambda = [0.5, 0.75, 1]$ (MMR_2)
4. CBD with $\alpha = [3, 1, 1]$ (CBD_1)
5. CBD with $\alpha = [3, 2, 1]$ (CBD_2)

In the stated configuration, we explore the effects of diversification on the accuracy by varying the level of diversification in different iterations. We note that MMR_2 and CBD_2 cases decrease the diversity in a more graceful way whereas MMR_1 and CBD_1 go directly to NN case after the initial iteration. We did not try to optimize the parameters (λ and α) of the diversification schemes and the values present themselves as mere intuitive estimates. We implemented a unit normalization method for each dataset and used cosine distance for all the experiments.

We also implemented the method given in [46] to compare the performance of our algorithms. This method uses a piecewise linear approximation (PLA-RF) for time series and associates a weight for each part of the series when calculating the distances to query. These weights are modified in each iteration of feedback according to the user feedback.

In the experiments, we model the user as a person seeking similar time series from the same class in the dataset. Under this model, the class of the series is used to generate relevant/irrelevant user feedback after each RF iteration. Items in the result set which are of the same class as the query are considered relevant and vice versa. The experiments were performed on a leave-one-out basis such that we use each and every time series in the database as a query and RF is

executed with the related parameters using the database excluding the query itself. Accuracy is defined by precision value based on the classes of the retrieved top-k set. Precision for the query is calculated using the resultant top-k list and the averaged precision over all the time series in the database is considered as the final performance criteria which are defined below:

$$\text{Query Precision}(T_q) = \frac{1}{10} \sum_{i=1}^{10} \delta(i)$$

$$\text{Average Precision} = \frac{1}{N} \sum_{\forall T_q \in TSDB} \text{Query Precision}(T_q)$$

$$\text{where } \delta(i) = \begin{cases} 1 & \text{if class of } T_q \text{ is equal to class of } R_i \\ 0 & \text{otherwise} \end{cases}$$

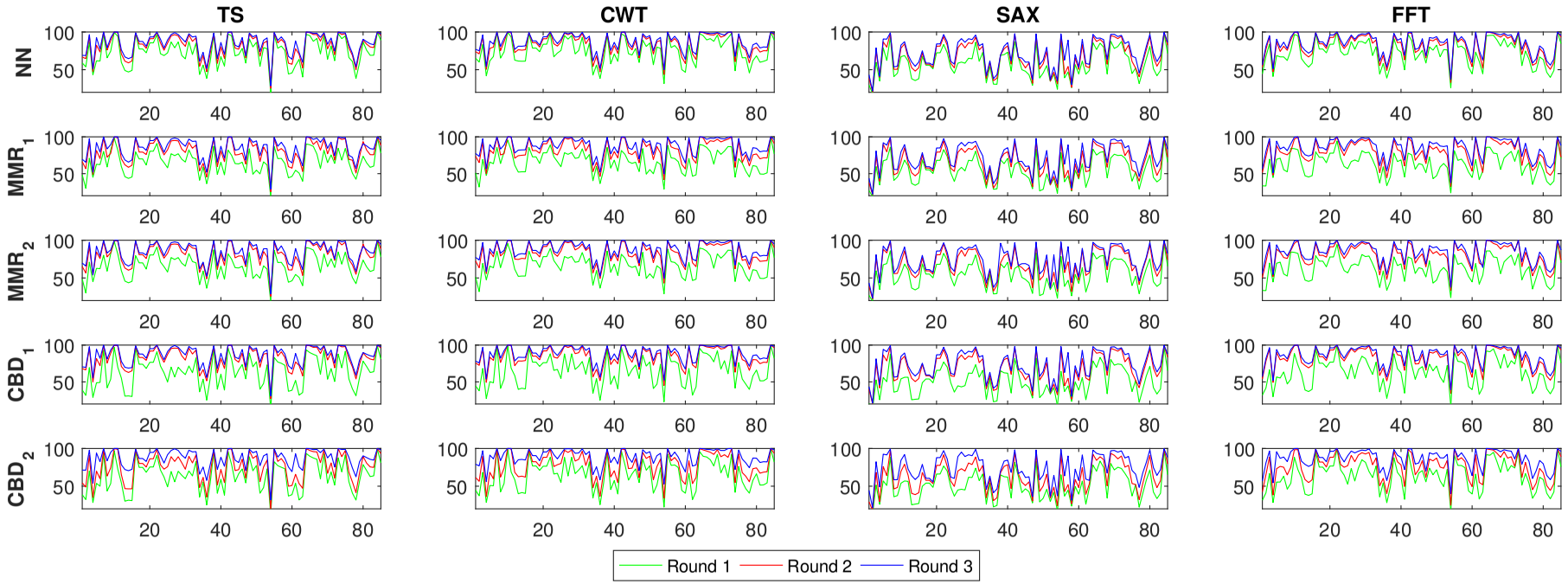


Figure 3.6: Performance for three rounds of RF for all the datasets (precision scaled to 100 in y-axis versus dataset number in x-axis)

3.6.2 Experimental Results

The experimental results for diversity in the result set are given in Figure 3.6 for all the data sets. Each row in the figure corresponds to one of five retrieval methods and each column corresponds to the representation (TS, CWT, SAX, and FFT) used. In each individual graph, the average precision in different RF iterations is plotted with the data set number given in x-axis. We present an aggregate result here to summarize the results.

Table 3.1: Average Increase (absolute) in Precision

RF Round	2	3
NN	9.08	12.73
$MMR(\lambda_1)$	14.19	19.98
$MMR(\lambda_2)$	15.75	20.01
$CBD(\alpha_1)$	18.88	22.98
$CBD(\alpha_2)$	12.60	23.44
PLA-RF	3.7	4.4

We calculated the precision (scaled to 100) difference between different rounds and the first round of RF for a particular representation, method and data set (4 representations x 5 methods x 85 data sets = total 5100 cases). Histogram of the resulting improvements is provided in Figure 3.7. Differences in precision (averaged over all cases) are provided in Table 3.1 to quantify the performance increase with the use of diverse RF. We also performed a t-test between the average values given in the table and a zero mean distribution to verify the statistical significance of the improvement. The p-values, in the range of 10^{-110}, are notably smaller than 0.05 which is considered as a threshold for significance. RF with the configurations given in this study improves accuracy in all cases without any dependence of data type or data representation and it provides significant benefits with 0.50 point precision increases in some cases. We also note that the proposed methods outperform the state of the art.

Since the experiments produced large amount of results (given the number of time series data types, representations, top-k retrieval methods), for illustrative purposes, we consider a reference case where the time series without any

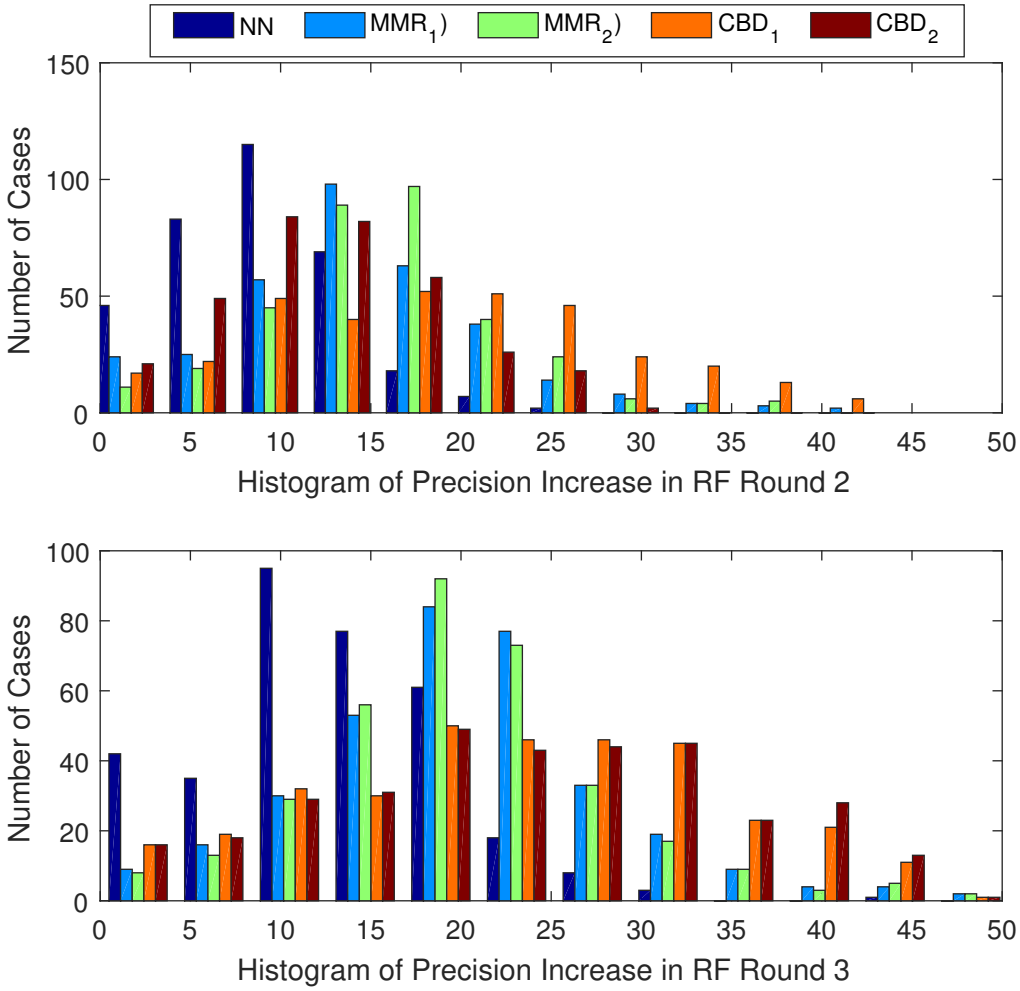


Figure 3.7: Histogram of increase in precision with different RF settings feedback

transformation and NN only method is used. Accordingly, for each RF round and each data set, the accuracy results are normalized to a total 100 with respect to the base case for that particular data set and RF round. Figure 3.8 shows the normalized results averaged over all the experimental cases. CWT based representation outperformed FFT, SAX and the time series without any transformation (TS) in nearly all cases. We note that representation parameters are not optimized and different results may be achieved by further optimizing transformation parameters. We did not perform such rigorous testing since it would divert us from the main focus of the study. However, CWT performed better consistently with no need of parameter optimization.

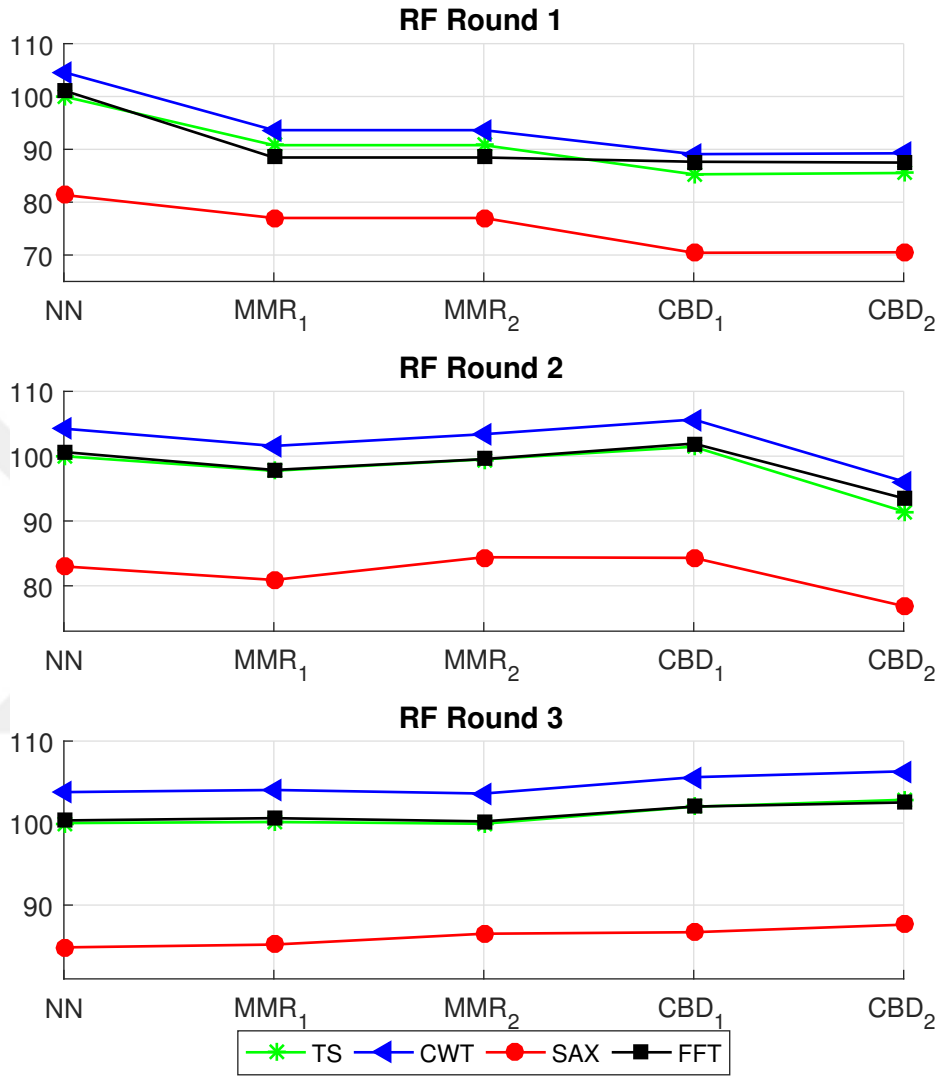


Figure 3.8: Normalized performances of different methods and representations

Although NN achieves the best performance in the first iterations of RF as expected, introducing diversity in the first iteration leads to a jump in RF performance exceeding NN in nearly all the cases. In RF round 3, CBD_2, the best performing method, adds 6.3% (p-value < 0.05) improvement over the reference case and 2.5% (p-value < 0.05) over the case which uses NN method with CWT. Diversity increases its effect further in the third rounds where NN is outperformed even in more cases with similar performance advancements. We also note that CBD_1 and CBD_2 perform best in second and third iterations respectively. This also underlines the enhancement in performance due to

increased diversity if the number of iterations increases.

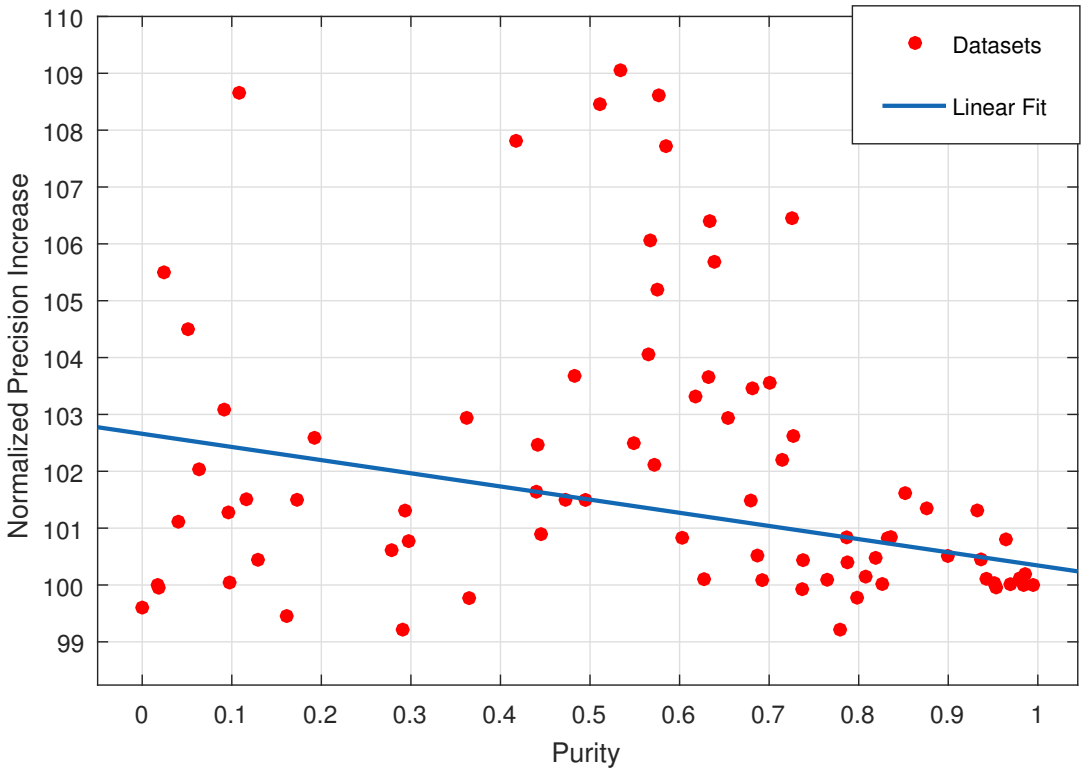


Figure 3.9: Normalized performances of different datasets versus purity of dataset

We also investigated the relation between cluster separability within the dataset and the improvements due to diversity. For this purpose, we calculated a separability score for each data set by using a k-means classifier and the average accuracy of the classifier is considered as the separability of the data set. This score, which is in the range $0 - 1$, essentially quantifies the separability of the classes where a score closer to 1 means easily classifiable datasets. We plot the normalized precision described in the previous paragraphs against the purity of the related dataset with the corresponding linear fit in Figure 3.9. Effect of diversity is not significant where classes are already separable (datasets with purity in the range $[0.75, 1]$) which is inline with our expectations. Positive effect of diverse retrieval increases when the classes are more interleaved which is the harder case in terms of system performance.

3.7 Conclusion

Even though combinations of diversity and RF have been explored in the fields of information retrieval and text mining, this successful concept did not attract much attention for time series analytics. This chapter explores the use of diversity enhanced RF for improving system accuracy and increasing user satisfaction in time series retrieval applications. We also illustrated the statistical background of the potential of diversity for increasing RF precision.

Experimental results on a wide variety of real data demonstrates that regardless of the selected representation, even with a relatively simple RF model, user feedback increases the retrieval accuracy. Results show precision improvements even in just one iteration of user feedback confirming the suitability and potential of RF in the similarity based time series search scenarios. Additionally results endorse our intuition and shows even higher accuracy improvements when diversity within the result set is used in the first iteration of RF in many of the cases.

The intuitive cluster based diversity method, without any rigorous parameter optimization, performed higher in terms of overall precision. Fine tuning of the diversity balance according to the dataset properties and user objectives can extend the improvements. We also note that methods with higher level of diversity (MMR_2 and CBD_2) perform superior with respect to their counterparts in the third iteration of RF. This also underlines the enhancement in performance due to increased diversity as the number of iterations increases.

The analysis of the results in terms of cluster separability withing the dataset provides evidence in favor of result diversification performing better in non-pure and non-separable data cases which are the challenging cases in the performance of machine learning and retrieval systems.

The presented results and analysis can serve as a basis for new approaches for diversification of time series data. During our exploration of the topic, we experimented different potential approaches for diversification of time series. We adapted matching based similarity (e.g., k-n match [68]) and STFT (short time fourier transform [69]) for time series diversification. However, we did not include their discussions in this study as the proposed approaches produced better results than these possible alternatives.

Algorithm 2 High-level algorithm for representation feedback system

r is given as the number of representations
Initialize parameter $NumberOfIterations$
q_1 is given as the initial query in time domain
$TSDB_r$ is given as the time series database with representation r
if Representation Feedback via Weighting **then**
  $TSDB$ = Concatenation the representations ($TSDB_r$)
  Initialize β weights
  Initialize parameter k // number of items to retrieve
else if Representation Feedback via Partitioning **then**
  Initialize parameter k_i for $i : 1 \ldots r$
end if
for $i = 1 \rightarrow NumberOfIterations$ **do**
  // Find Top-k results using any alternative method
  if Representation Feedback via Weighting **then**
    R = Top-K(q_1,...,q_i,$TSDB$,β)
  else if Representation Feedback via Partitioning **then**
    $R = \emptyset$
    for $j = 1 \rightarrow r$ **do**
      $R = R \cup$ Top-K(q_1^j,...,q_i^j,$TSDB_j$,k_j)
    end for
  end if
  // Let user grade the retrieval results
  (Rel,$Irrel$) = UserGrade(R)
  // Expand query points via relevance feedback
  if Representation Feedback via Weighting **then**
    q_{i+1} = Relevance_Feedback(Rel,$Irrel$)
  else if Representation Feedback via Partitioning **then**
    $R = \emptyset$
    for $j = 1 \rightarrow r$ **do**
      q_{i+1}^j = Relevance_Feedback(Rel,$Irrel$)
    end for
  end if
  // Update representation feedback parameters
  if Representation Feedback via Weighting **then**
    β = UpdateWeights(β,Rel,$Irrel$)
  else if Representation Feedback via Partitioning **then**
    k_i = UpdateK(k_i,Rel,$Irrel$)
  end if
end for

Chapter 4

Variations of Time Series Relevance Feedback

This chapter further examines the different variations of the RF system defined in Chapter 3. We present two modifications to the proposed algorithms to satisfy different application requirements and data properties.

Firstly, we propose representation feedback which is a distinctive aspect for time series data. The proposed two methods tackle the problem of selecting the satisfactory representation automatically without a human intervention using the already available user annotations. This can be useful in cases where different representations are relevant for different users (groups) or in dynamic cases where optimal representation can vary because of the incoming data properties.

Secondly, we propose the use of autoencoders for learning data-aware representations from the time series data to be used in our RF framework. Successful application of this concept leads to decrease in both computational and memory requirements of the retrieval engine. We also report that learning data-aware representation leads to increase in accuracy in some cases which indicates the potential of this method for other time series analytics tasks.

4.1 Representation Feedback

Choice of representation has significant effects on the accuracy of the system as expected and observed from the experimental results with result diversity. This can be caused by the properties of the time series such that meaningful and useful clusters are not separated as good as a expected with particular representation when compared with others. This issue can partially be addressed by offline experimenting with different representations against some performance parameter and choosing the representation for the system accordingly. But this approach can fail when the data content and properties change with dynamic modifications to the database. Secondly, a fixed and particular representation may not be able to represent a user's (or a user group's) intention as well as some other representation. Moreover, different users' objectives can vary even when using the same time series database. As an example, a group of users might be interested in time domain features while other groups searching for a frequency domain feature. A generic and universal time series representation appropriate for all the possible application areas and user intentions is simply not possible, proved by the vast amount of research effort on the problem. Hence, we naturally see different time series representation proposed in the research community for different cases and applications.

For alleviation of the representation choice problem, one can exploit the user feedback for deciding on both related items and related representation(s). We investigate two methods for representation feedback. The first method partitions the top-k list proportionally to the different representations available and attempts to find best performing representation or the best combination of representations. Secondly, we concatenate different representation vectors and use a feature learning method to select the best performing parts of a variety of representations. The high level flow of the representation feedback algorithm is given in Algorithm 2. This method is used in conjunction with query modification in each iteration as presented in Chapter 3. The benefit of fusing different time series representations is to reach an aggregate expressive power from each representation, with implicit diversification to improve the RF performance.

4.1.1 Representation Feedback via Top-k List Partitioning

In this method, we populate the top-k list using retrieved items from numerous representations and interpret the user annotation to converge to the representation which is expected to maximize user satisfaction. This method is used in conjunction with the traditional query modification in the following RF iterations. The important benefit of fusing different time series representations is the expressive power of each representation which depicts a contrasting angle of the data which may not be captured otherwise. This ensures a comprehensive result set with an implicit diversity generated by the retrieval process further improving RF performance. The modifications to the RF system is presented in Figure 4.1.

Proposed technique partitions the k value of the system into different k_i values (where the sum of k_is add up to k), each value regulating the share of different representations in final result set. Equal distribution can be selected in the first round of RF. Any priori knowledge about the performance of the representations with respect to user intention or data properties can be used to calculate better initial estimate. Feedbacks from all the user base of the RF system can be used to set the starting k_i values more accurately.

Top-k set from each of the representations available are found by using any of the NN, MMR or CBD retrieval methods. k_i items from each of the respective top-k list are chosen, compiled into a single top-k list and presented to the user. After evaluation of the user, k_i values are updated according to the accuracy of the related representation. Starting value and update of the k_i partition are given in Equation 4.1.

$$\begin{aligned} &\textbf{Initialization:} \\ &k_i = \left[\frac{k}{r}\right] \text{ where } r \text{ is number of representations} \\ &\textbf{Update:} \\ &k_i = \frac{\text{Number of relevant items from representation } i}{\text{Number of relevant items}} \;\; \forall i \leq r \end{aligned} \tag{4.1}$$

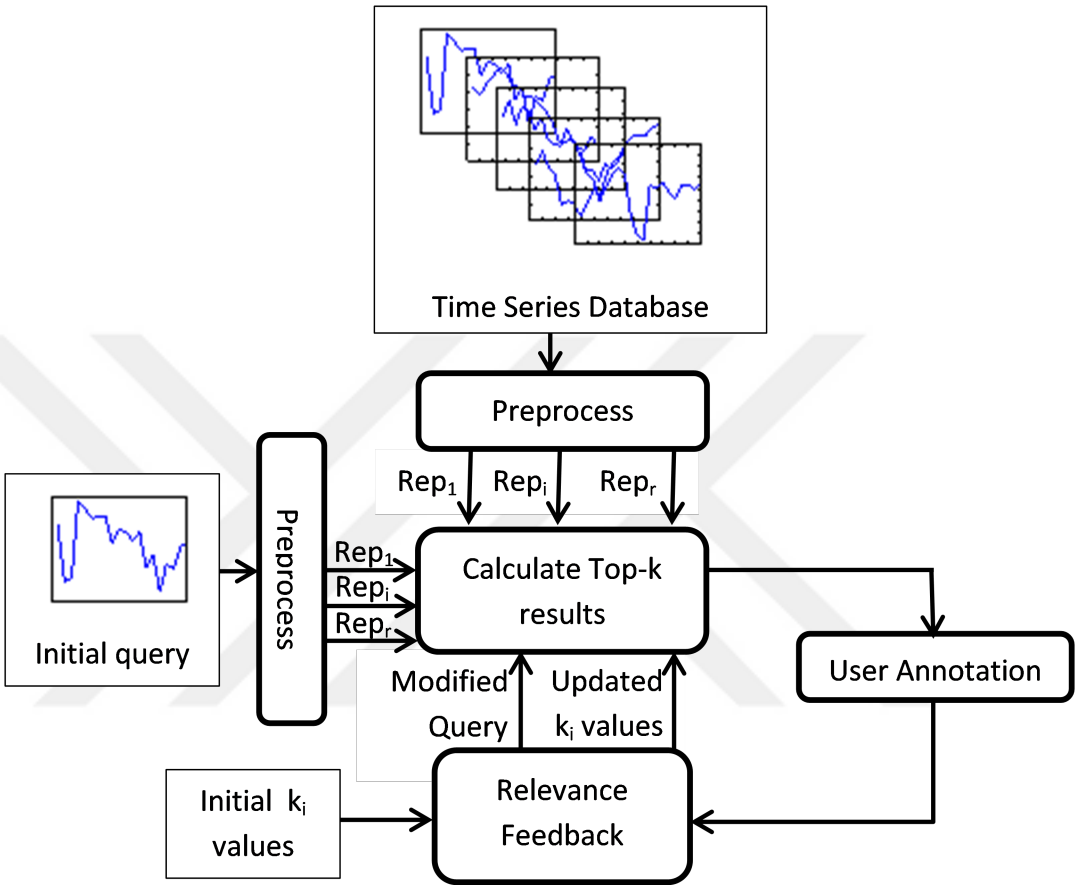


Figure 4.1: Representation feedback with top-k list partitioning

4.1.2 Representation Feedback via Weighting

As our main aim is to understand the user intent and we lack any prior indication about the intent, it is important to identify generic local and global features which are beneficial in expressing different user targets. By building a framework built on general principles, each user intention will be associated with a subset of these generic features. For this purpose this technique begins with concatenation of each representation forming a longer vector including different properties of the time series, depending on the perspective and expressive power of each representations used. This aggregate vector can be considered a new hyper-vector which can map the user objectives more thoroughly, due to a composite and holistic view of time series. Representations should be chosen such that

this combination includes possible user intents as much as possible by selecting representations with different perspectives, each providing novel information. The flow of the method is provided in Figure 4.2.

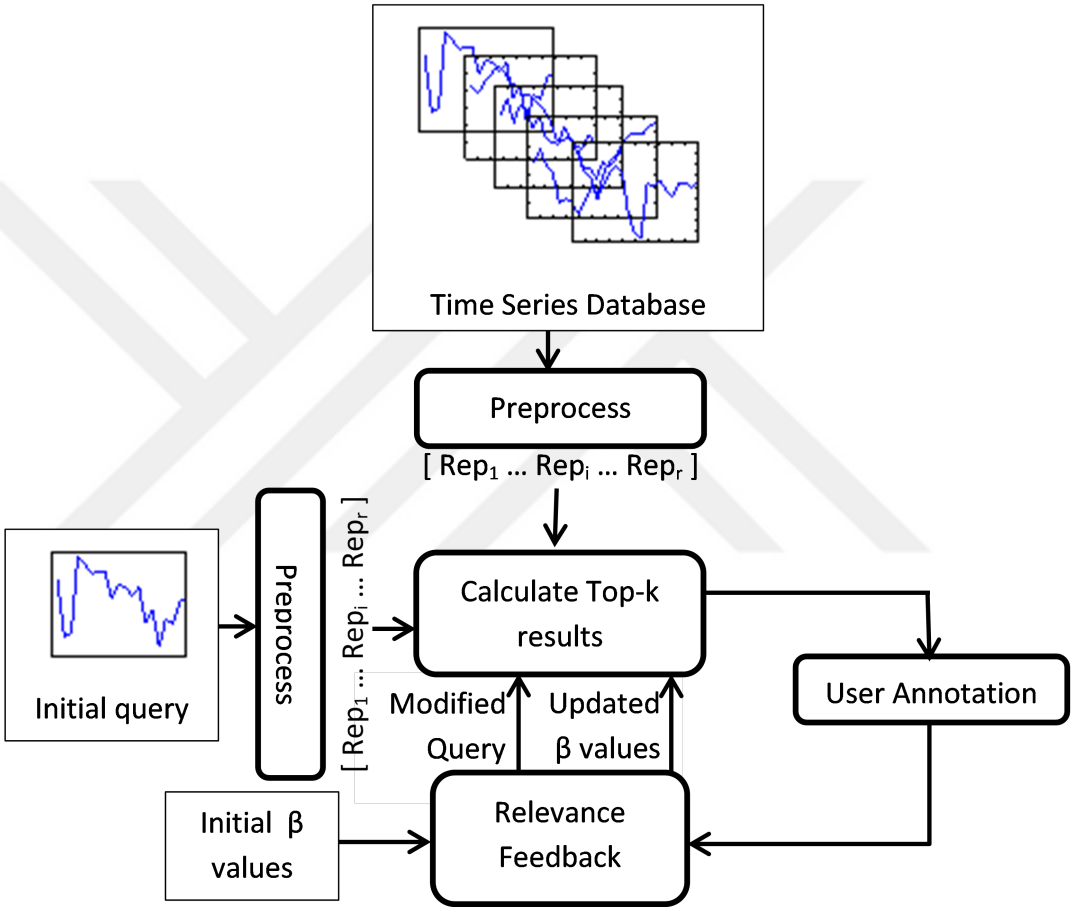


Figure 4.2: Representation feedback via weighting approach

Following the aggregation step, specific features important to the user should be identified based on the interactions. For this purpose, a simple learning step is implemented which is based on a linear model of the generic properties by modifying similar approaches from information retrieval [22]. The user objective is modeled as a normalized linear combination of the properties of the time series where each weight corresponds to the relevance of the respective property for user objective:

$$\text{User Intention} = \sum_{j=1}^{N^F} \beta^j . T^j \tag{4.2}$$

where N^F : Number of features and $\sum_{j=1}^{N^F} \beta^j = 1$.

To estimate the β weight parameters, we analyze the variation of the particular feature within the relevant items. If a particular feature is favored by the user, some consistent values in that particular field is expected and vice versa. We determine the importance of each feature by comparing the decrease of the standard deviation of the particular feature within the relevant items and the variation of that specific feature in the whole dataset. We use this ratio as an estimate for β parameters. Algorithm 3 details the estimation and iterative update of β parameters. β importance weights are used in the similarity calculations of top-k ranking to blend user preferences in the present ranking. Importance weighting can be used in all of the previously explained diverse retrieval methods by introducing the related process in the feedback updates and the distance calculations.

Algorithm 3 Estimation and update of β parameters

INITIALIZATION
T_i^j : jth feature of time series i (or aggregated vector)
N^F: number of features
σ^j = Standard deviation over $T_1^j, T_2^j, \ldots, T_N^j \quad \forall j$
$\beta^j = \frac{1}{N^F} \quad \forall j : 1, \ldots, N^F$
UPDATE
for each RF round **do**
  if $|\mathbf{Rel}| > 3$ **then**
    Analyze standard deviation differences
    for $j = 1 \rightarrow N^F$ **do**
      $\hat{\sigma}^j$ = standard deviation over T_i^j where $T_i \in (\mathbf{Rel})$
      $\Delta\sigma^j = \frac{\sigma^j}{\hat{\sigma}^j}$
    end for
    Normalize $\Delta\sigma^j = \frac{\Delta\sigma^j}{\sum_j \Delta\sigma^j} \quad \forall j : 1, \ldots, N^F$
    Update $\beta^j = \frac{\Delta\sigma^j + \beta^j}{2} \quad \forall j : 1, \ldots, N^F$
  end if
end for

4.1.3 Evaluation

We have experimented with the proposed representation feedback method and we summarize the results for its use in conjunction with item diversity. Results of normalized performance with respect to the baseline (NN method in the first round of RF) averaged over all the data sets are given in Figure 4.3. We note that, in contrary to our item-only diverse RF method results provided in the previous section, pure NN retrieval achieves similar performance when top-k partitioning representation feedback is used. We associate this difference in results with the observation that data items retrieved from different representations implicitly provide a diverse result set which improves the performance of RF without the need for further item diversity.

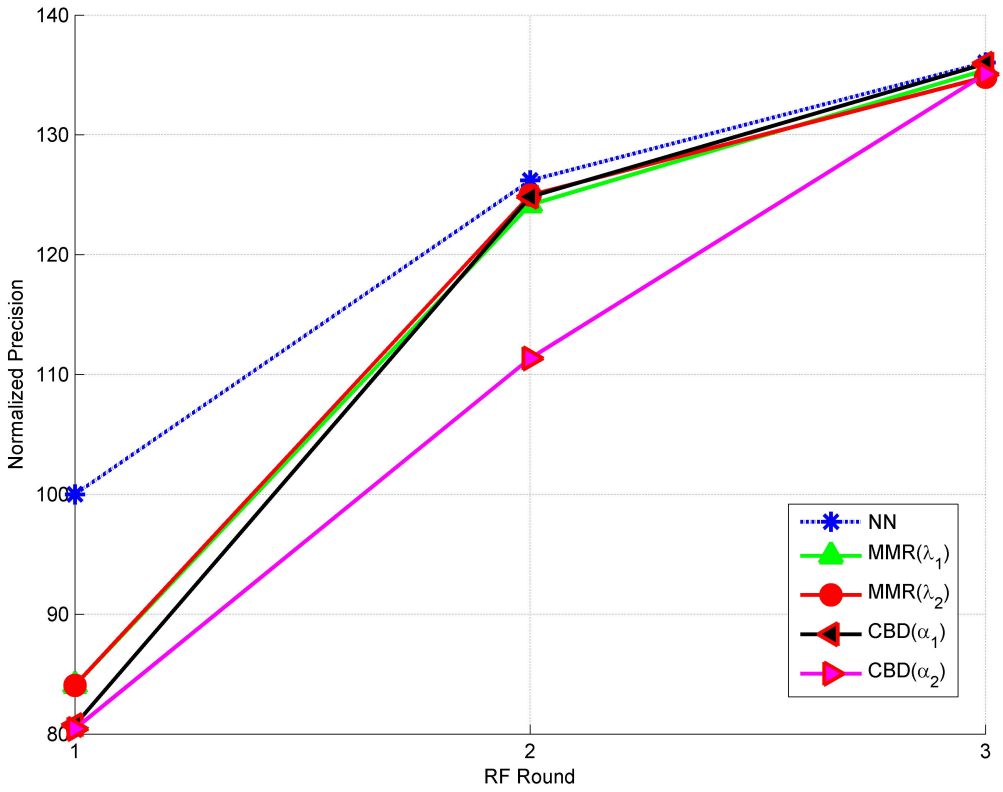


Figure 4.3: Normalized performances of top-k partitioning representation feedback methods

We also compare top-k partitioning representation feedback with the best performing method found in the item diversity experiments in Figure 4.4. The figure illustrates that our principle aim is achieved by the method and as the RF process evolves with subsequent iterations the system converges to the best performing representation. Pure NN retrieval is used in this comparison, since it performed similarly to the other diverse retrieval methods when used in combination with representation feedback.

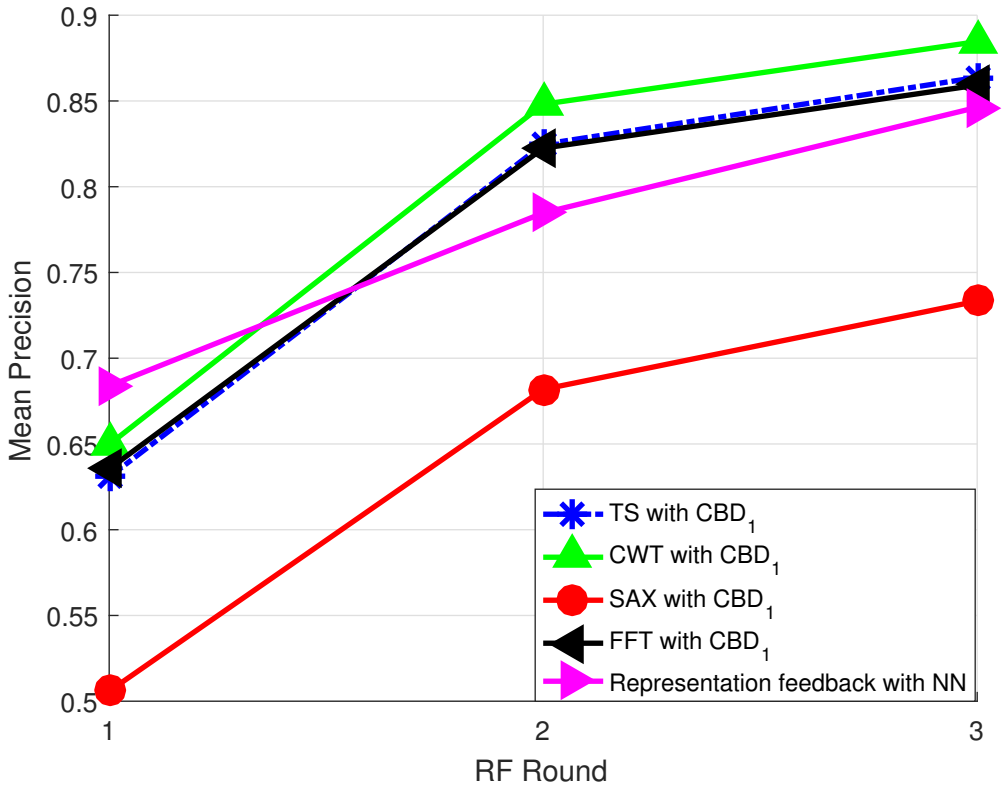


Figure 4.4: Accuracy comparison of top-k partitioning representation feedback with item-only diversity

Secondly, we present the results for representation feedback via weighting on multi-representation vectors. For clarity, we performed experiments on three randomly chosen data sets: ECG200, fish and synthetic control with CWT, TS, SAX and CWT+TS representations. NN and the CBD_1 diverse retrieval methods have been used for analyzing diversity effect on performance and the results are depicted in Figure 4.5. After analyzing the results, we observe that

the weighting mechanism is not compatible with SAX-Bitmap representation since the accuracy in the second round of RF is lower than the performance the initial round of retrieval. We consider to possible explanations to this: SAX-bitmap features may well be correlated with each other or SAX-bitmap representation is not suitable for weighted similarity measures. Leaving out the SAX-Bitmap representation, in most of the cases weighting approach can provide significant performance gains. Moreover, contrary to top-k partitioning case, diversity retrieval techniques can further enhance the accuracy of the system in the weighted representation feedback case.

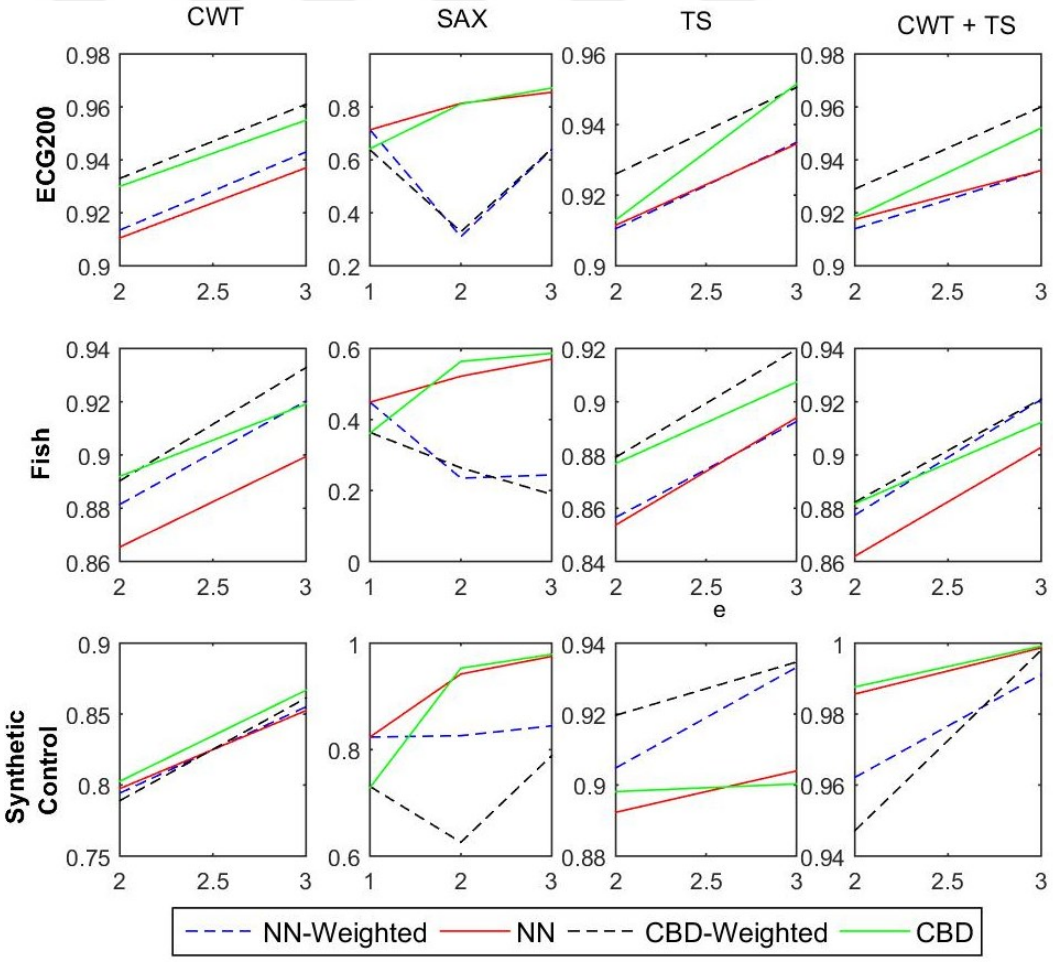


Figure 4.5: Performance of representation feedback with weighting (precision in y-axis vs RF round in x-axis)

The figure also shows that the performance of combined representations (CWT and TS representation couple is considered due incompatibility of SAX-Bitmap)

with representation feedback via weighting approach can increase the performance of the RF system. We also note an interesting case, the synthetic control data set, where the aggregated multi-representation (CWT+TS) has performed considerably better than the individual representations. This can be considered an example case to our previous assertion that the extended span of multiple representations can explain different aspects of the data or user intention more accurately leading to a more satisfactory overall system.

4.2 Time Series RF using Autoencoders

Autoencoders have been proposed in machine learning as a structure to learn and transform the input data into a set of important parameters from which the original data is synthesized back. In this respect, autoencoders are a good candidate to extract useful data-oriented features which are also low-dimensional. One can utilize autoencoders for two important prospects in time series: choosing and blending different time series representations, and reducing the already extracted set of features to important features. Since the autoencoder model does not need any teacher who dictates the class of the time series, this unsupervised learning method can be applied for RF based time-series retrieval achieving the two goals via an analysis of the dataset.

Autoencoders are implemented using neural networks, defined by the layers of neurons stacked on top of each other which can be connected in different configurations. Each artificial neuron is composed of a weighted summation unit, summing all the signals in its input and an activation function (σ) which is generally chosen as a non-linear function. A class of neural networks called multilayer perceptron (MLP) which has at least three layers (an input and output layer, one or multiple hidden layers) is constructed of fully interconnected neurons. The topology and the number of nodes in the network determine the space of possible learnable functions whereas the weights between nodes after the training phase defines the exact functionality of the network.

Time series data is used in the autoencoder network both as input and output where the input data is first 'encoded' to a new representation space $\mathbf{z}$ ($\mathbf{z} = \mathcal{H}_{encoder}(TS_i)$) and then decoded using the encoded values to the final output ($TS'_i = \mathcal{H}_{decoder}(\mathbf{z})$). The criteria for a learned model and representation is defined by a loss function of choice based on the data and application which is usually the discrepancy between the data and its generated counterpart. Although replicating the input time series instead of classification or ranking may not be considered a good learning target, the useful and important product of autoencoders is the encoded data, $\mathbf{z}$, which identifies key structures within the data. This process, which can also be considered as a non-linear dimension reduction determining local and global features, encodes the raw or transformed time series data into a sparse vector to be used in the RF based retrieval framework. Autoencoder essentially learns a data aware representation, and reduces the length of the time series which will decrease the runtime of the retrieval process. It also enables combining of different transformations and identifying important features from different representations if used with multiple representations. The overall algorithm is presented in Algorithm 4.

We employ an MLP based autoencoder network whose configuration is provided in Figure 4.6 where TS_i is the input time series and TS'_i is the synthesized series using the encoded values ($\mathbf{z}$) in the hidden layer. Constraining the number of neurons in the encoder (hidden) layer to be considerably less than the input layer forces the model to learn a subset of important features within the data. We denote the parameter $\theta < 1$ (defined in Algorithm 4 Line 8) as the ratio of the number of neurons in the encoder to the number of input layer neurons to quantify compression ratio.

We aim to minimize the difference between original and regenerated counterparts ($(TS_i - TS'_i)^2$) or ($(\mathcal{F}(TS_i) - \mathcal{F}(TS_i)')^2$)) in the training phase. Training of the autoencoders (Algorithm 4 Line 10-13) is carried out by backpropagation which is a gradient descent based optimization technique used widely in training neural networks. We also include regularization parameters to the cost function used in the optimization process so that each neuron in the hidden layer activates with respect to a group of time series specializing to specific features present in

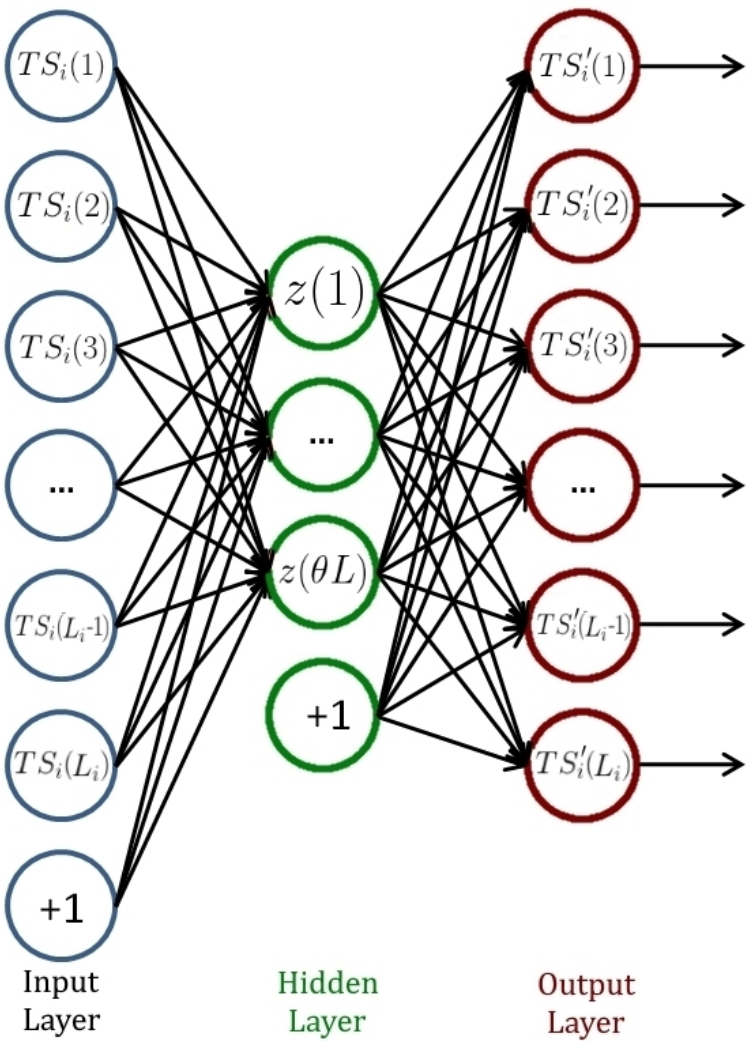


Figure 4.6: Autoencoder network structure

this particular group. The result of the training phase provides us the weight matrix which characterizes the encoder functionality $\mathcal{H}_{encoder}$.

After the training phase, the time series database and queries are encoded into the newly learned representation using the weight matrix (Algorithm 4 Line 15-18) and the retrieval phase with the diversity achieving methods can be executed without any change.

The database ($TSDB$ in Algorithm 4 Line 6) can be constituted of the following: time series (TS), transformation (FFT, CWT, SAX, etc.) of time

series, a combination of time series and/or its representation (e.g. TS, FFT, CWT features concatenated). If a combination is used the system can extract different perspectives from multiple representations from the data similar to the representation feedback process presented in Section 4.1.

4.2.1 Algorithmic Complexity

The initial size of time series database, $N \cdot L$ is reduced to $N \cdot [\frac{L}{\theta}]$ after the encoding process which changes the values of the coefficients in the computational complexity given in Section 3.4.1. This proportional decrease is achieved both in terms of computational load and memory. The updated complexity values are provided below:

The NN based retrieval first calculates distances to all items in dataset ($O(N \cdot [\frac{L}{\theta}])$) and finds k nearest items ($O(k \cdot N)$) which corresponds to a total complexity of $O(N \cdot [\frac{L}{\theta}] + kN) = O(N)$.

For the MMR case we have two possibilities with respect to Equation 3.3:

- Without memoization: Distance calculations to all items in the dataset ($O(N \cdot [\frac{L}{\theta}])$), distance calculations for relevant set items ($O(N \cdot [\frac{L}{\theta}] \cdot (k-1) \cdot (k-2)/2) = O(N \cdot [\frac{L}{\theta}] \cdot k^2)$, finding the minimum distance element k times ($O(k \cdot N)$) with an overall complexity of $O(N \cdot [\frac{L}{\theta}] \cdot (1 + k^2)) = O(N)$.
- With memoization: Distance calculations to all items in the dataset ($O(N \cdot [\frac{L}{\theta}])$), distance calculations from lookup table for relevant set items ($O(k-1) \cdot (k-2)/2) = O(k^2)$), finding the minimum distance element k times ($O(k \cdot N)$) with an overall complexity of $O(N \cdot [\frac{L}{\theta}] + N \cdot k) = O(N)$ (where $N \cdot [\frac{L}{\theta}] \gg k^2$).

For the CBD case, we first find αk nearest neighbors ($O(N \cdot ([\frac{L}{\theta}] + \alpha k))$ and cluster the results. K-means clustering (based on Lloyd's which has a limit i for

the number of iterations) is considered $O(N \cdot k \cdot [\frac{L}{\theta}] \cdot i) = O(N \cdot k \cdot [\frac{L}{\theta}])$ algorithm. The total complexity for CBD case is $O(N \cdot ([\frac{L}{\theta}] + \alpha k + [\frac{L}{\theta}] \cdot k \cdot i)) = O(N)$.

4.2.2 Results for Diverse RF Using Autoencoder

We execute the same experimental setup, using TS, CWT, SAX representations and NN, MMR_1, MMR_2, CBD_1, CBD_2 methods to evaluate the RF system with autoencoders. Additionally, we also experimented on the combination of all the representations (TOTAL) as the input to the system.

We have varied the sparsity index $\theta \in [3, 6, 9]$ to observe the effects of compression on the results. Autoencoder hidden layer node numbers are selected as $[\frac{L}{\theta}]$ and are trained using MATLAB neural network toolbox with default values except for sparsity regularization term which is selected as 4 instead of default 1 to emphasis sparsity in the encoder.

We have experimented on the full 85 data sets, containing time series data of very different properties, to assess the generality of the autoencoder based method. We denote the different cases with the respective input representation and sparsity index, e.g. CWT^3 denotes the outputs of a trained autoencoder with $\theta = 3$ (length of data is reduced to a third of the original length) and CWT transformed time series as input.

We use the normalized precision to quantify the performance, such that precision in the first round of RF with NN is normalized to 100 and all the other precision values are scaled respectively. The normalization is performed for each dataset and transformation (each θ case is also considered a new transformation since the data input for the RF system changes) separately to illustrate the effect of RF more discretely. The results provided in Table 4.1 demonstrate that diverse retrieval methods achieve similar accuracy improvements with the encoded data. We also observe that even though we reduce the data into a much reduced form in the $\theta = 9$ case the diverse RF system is still working with graceful degradations instead of a sudden breakdown in performance. Precision improvements for the

Table 4.1: Normalized precision improvements with varying autoencoders for third round of RF

	NN	**MMR_1**	**MMR_2**	**CBD_1**	**CBD_2**
TS	119.7	120.0	119.7	122.3	123.5
TS^3	120.3	121.4	121.5	123.2	124.3
TS^6	119.7	120.4	120.2	122.5	123.7
TS^9	119.0	120.0	120.0	122.4	123.6
CWT	119.2	119.5	119.0	121.4	122.3
CWT^3	117.7	118.6	118.3	120.1	120.7
CWT^6	117.6	118.5	118.4	119.8	120.3
CWT^9	117.9	118.8	118.5	120.3	121.0
SAX	126.8	127.4	130.3	129.6	131.2
SAX^3	121.4	119.8	121.8	123.7	124.7
SAX^6	121.0	119.2	122.3	123.3	124.7
SAX^9	119.8	118.2	121.1	122.4	123.9
FFT	119.5	119.9	119.4	121.7	122.3
FFT^3	120.6	121.1	120.6	122.7	123.3
FFT^6	119.5	120.3	119.9	121.8	122.6
FFT^9	119.3	119.7	119.6	121.8	122.2
TOTAL	119.1	119.7	119.1	121.1	121.8
$TOTAL^3$	118.5	119.3	118.7	120.4	121.0
$TOTAL^6$	118.8	119.4	119.0	120.7	121.2
$TOTAL^9$	118.5	119.3	118.9	120.4	121.1

$TOTAL$ representation are also on a similar scale.

The average precision levels (scaled to 100) over all of the datasets are provided in Table 4.2 with respective methods for the third RF round. We can see that autoencoded features are performing without significant losses until $\theta = 6$ value except for SAX-Bitmap case which is considered important since the data is reduced to 16.7% of its original size. The relatively close results are mainly due to the averaging of the significant number of high performing databases in the dataset which are evident in Figure 3.6.

We note the performance of the $TOTAL$ representation which increases its

Table 4.2: Average precision levels for diverse RF with varying configurations

	TS	**TS^3**	**TS^6**	**TS^9**
NN	85.0	84.8	83.9	82.3
MMR$_2$	84.8	85.4	84.1	82.8
CBD$_2$	86.9	86.9	86.0	84.8
	CWT	**CWT^3**	**CWT^6**	**CWT^9**
NN	87.1	85.9	85.2	84.2
MMR$_2$	86.9	86.2	85.6	84.6
CBD$_2$	88.9	87.6	86.8	86.0
	SAX	**SAX^3**	**SAX^6**	**SAX^9**
NN	71.9	65.5	64.5	63.4
MMR$_2$	73.3	65.7	64.9	63.9
CBD$_2$	74.1	67.2	66.4	65.4
	FFT	**FFT^3**	**FFT^6**	**FFT^9**
NN	84.7	84.0	82.3	80.9
MMR$_2$	84.4	84.0	82.5	81.0
CBD$_2$	86.3	85.6	84.1	82.5
	TOTAL	**TOTAL^3**	**TOTAL^6**	**TOTAL^9**
NN	86.9	88.7	88.3	88.2
MMR$_2$	86.8	88.8	88.4	88.4
CBD$_2$	88.5	90.2	89.8	89.9

performance as it gets sparser and outperforms all the other configurations, supporting the expectation that the autoencoder can remove unnecessary features from the representations and amplify the useful features directly by training on the data. We also looked at the lowest performing 15 datasets (which have precision levels below 70% after 3 rounds RF with the NN method), i.e., cases that need most improvements. The results for this subset are provided in Figure 4.8 under different transformations and methods for the third RF iteration. We present the results for NN and CBD_2 with $\theta = 6$ autoencoders to illustrate the general case based on our previous findings. The performance increases are visible for these datasets more explicitly with CBD_2 for $TOTAL^6$ approximately the upper bound and the base case NN with TS is the lower bound for performance. We can see from the figure that the proposed transformations, autoencoder structure and the diverse retrieval methods can increase the accuracy considerably with nearly 0.20 point improvement (around one-third increase relatively). In addition to these analysis, we experimented on using principal

component analysis (PCA) with similar compression values as an alternative to autoencoders on randomly selected datasets and have observed that autoencoder learned representations perform better in terms of precision.

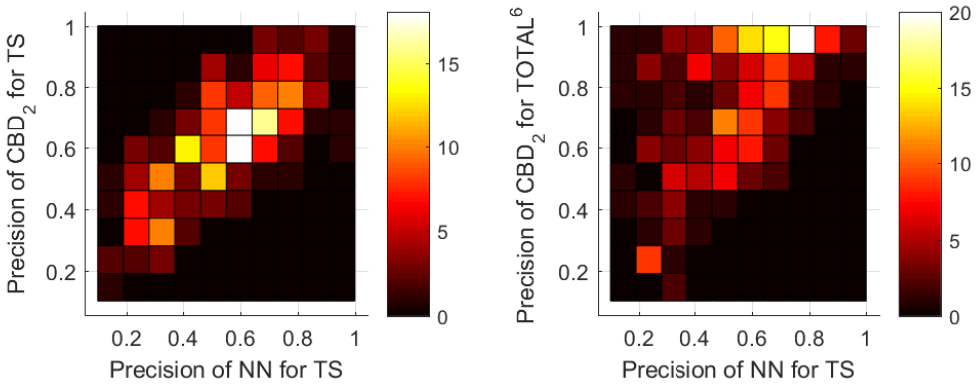


Figure 4.7: 2-D histograms (number of queries) of query precision under different methods and transformations in the third iteration of RF for Worms dataset

We illustrate the findings for query precisions and individual queries, over Large Kitchen Appliances and Worms datasets, to get more insights on the problem and the proposed approach. The method returns diverse results in the first RF iteration, which directs the system for subsequent iterations. We see that the queries are performing better under the CWT/FFT transformations which identify more distinguishable features in these cases. We also observe that, encoded $TOTAL^6$ representation can distinguish the better performing transformation and amplifies it to increase the retrieval performance. This is depicted in Figure 4.7 in which we plotted how the performance of individual queries change with respect to retrieval method (NN vs CBD_2) and transformation (TS vs $TOTAL^6$). If the precision in the first round of RF is very low for the query, diversity RF has a minimal positive effect. The highest gains are seen in the middle range of precision (0.2-0.7) where there is room for improvement and enough information for the RF mechanism to work. It is also evident that choice of representation can change the end result significantly for all query cases.

Algorithm 4 Overview of RF system using autoencoders

Initialize k : number of items in result set
Initialize $RFRounds$: number of RF iterations
Initialize $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_{RFRounds}]$: MMR parameters
Initialize $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_{RFRounds}]$: CBD parameters
Input q_1 : initial query (transformed if needed)
Input $TSDB$: time series database (transformed if needed)
// Parameters for autoencoder
Initialize θ for sparsity level of the autoencoder
σ as activation function of ANN
Train Autoencoder
Initialize network with L_i input nodes, $\theta.L_i$ input nodes and L_i output nodes
Train the network using back propagation
Extract the weight and bias $(\mathbf{W}, \mathbf{b})$ matrices for encoder
Diverse Retrieval System
for $i = 1 \rightarrow N$ **do**
    $TSDB'(i) = \sigma(\mathbf{W}.TS_i + \mathbf{b})$
end for
Transform the query : $q_1' = \sigma(\mathbf{W}.q_1 + \mathbf{b})$
for $i = 1 \rightarrow RFRounds$ **do**
    // Find Top-k results
    if Nearest Neighbor **then**
        R = Top-K(q_1',...,k,q_i,$TSDB'$)
    else if MMR **then**
        R = Top-K_MMR(q_1',...,q_i,k,λ_i,$TSDB'$)
    else if CBD **then**
        R = Top-K_CBD(q_1',...,q_i,k,α_i,$TSDB'$)
    end if
    // User annotation of the result set
    (**Rel**,**Irrel**) = User Grade(**R**)
    // Expand query points via relevance feedback
    q_{i+1} = Relevance_Feedback(**Rel**,**Irrel**)
end for

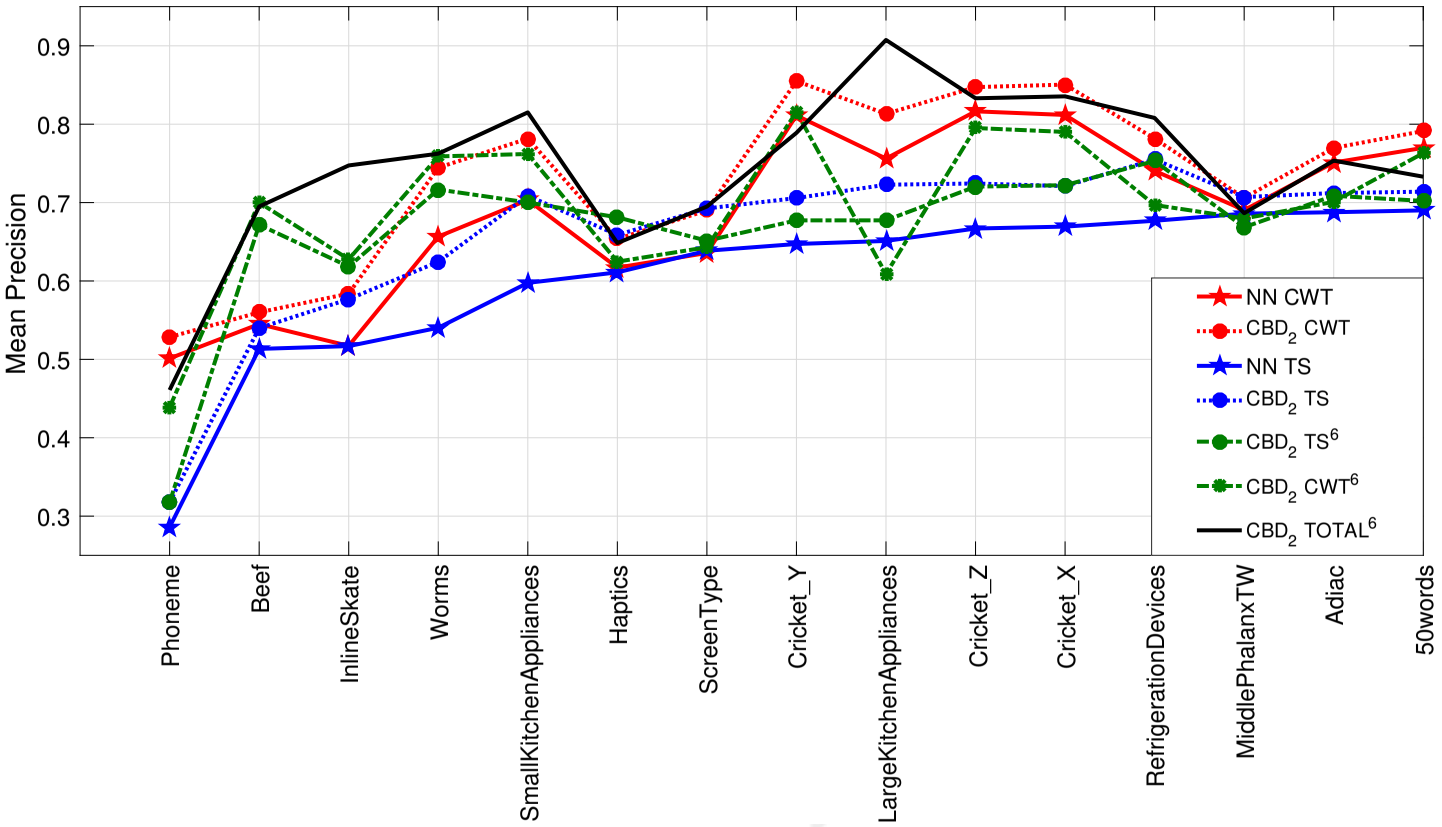


Figure 4.8: Performance of RF with various configurations for datasets with low precision

4.2.2.1 Runtime Performance

We present the runtime performance of the methods, initially with the times of the transformations into different representations. The computation platform is MATLAB running on a Windows 10 with Intel i7 4720HQ 2.6 GHz processor and 16 GB of RAM. The accumulated runtime for all the 85 datasets is provided in Table 4.3. SAX-Bitmap transformation is the slowest transform with a significant difference. We think that this is mainly due to very efficient FFT libraries available in MATLAB which is also used extensively in CWT transformation code.

Table 4.3: Total transformation runtime for all the datasets (minutes)

CWT	**SAX**	**FFT**
4.28	62.87	0.76

We also examined how the training time of the autoencoder varies with respect to θ parameter and to different transformations. The results for all the datasets summed up is provided in Table 4.4. We observe that training time increases with the length of encoded data. This is expected since the length of the time series affects the number of nodes and the total number of weights in the network which directly affects the total training times.

Table 4.4: Total training time for autoencoders (minutes)

	$\theta = 3$	$\theta = 6$	$\theta = 9$
TS	127.29	92	79.44
CWT	176.02	119.48	99.84
SAX	63.56	56.78	53.83
FFT	66.69	57.53	53.85
TOTAL	719.17	405.98	308.78

The total experiment runtime (over all the datasets total) is provided in Table 4.5 with respect to different retrieval methods, sparsity index(θ) and transformation methods. We note the significant reduction in runtime for autoencoded data in which some cases we have 7 fold decrease with respect to full time series data. Each transformation has a different runtime performance

which depends on the length of the transformed time series, which is expected, as mentioned in Section 4.2.1 (Average length for different transformation is as follows: TS: 422.2, CWT: 564.3, SAX: 256.0, FFT: 212.0, $TOTAL$: 1454.5). We also see that the diversification methods, MMR and CBD, have comparable runtime performances.

Table 4.5: Total runtime of experiments (minutes)

	NN	**MMR_1**	**MMR_2**	**CBD_1**	**CBD_2**
TS	119.5	148.7	157.3	136.8	151.8
TS^3	41.6	58.1	66.9	55.4	68.1
TS^6	21.8	35.0	43.9	34.7	46.6
TS^9	15.4	27.6	36.7	28.1	39.6
CWT	125.2	156.1	164.4	142.3	157.8
CWT^3	48.7	66.5	75.9	64.4	77.1
CWT^6	25.2	39.1	49.0	39.8	52.2
CWT^9	18.3	31.3	40.5	31.3	43.4
SAX	65.5	86.9	96.2	80.1	92.9
SAX^3	29.7	44.7	54.3	42.4	54.5
SAX^6	16.5	29.2	38.5	29.0	40.5
SAX^9	11.3	23.1	32.4	23.5	34.8
FFT	61.2	81.1	90.0	76.6	90.0
FFT^3	21.9	35.0	43.9	34.7	46.5
FFT^6	12.0	23.6	32.3	24.6	36.0
FFT^9	8.8	19.9	28.7	21.2	32.7
TOTAL	358.2	426.0	451.3	394.5	439.7
$TOTAL^3$	134.4	166.0	175.1	152.3	168.9
$TOTAL^6$	68.6	90.0	99.0	84.6	98.4
$TOTAL^9$	46.8	64.1	73.1	61.6	74.6

4.3 Conclusion

In this chapter, we propose two different extensions to further enhance the time series RF framework defined in Chapter 3 aiming to solve the representation selection problem and to decrease the computational load of the system.

First of all, we propose a representation feedback method which automates the off-line process of representation selection for time series data. This is an important aspect of time series data since it can affect the end result of the analytics task significantly. Among the two methods proposed, top-k partitioning method forms the result set from a variety of different representations. According to the relevancy of the selected items, the system self-regulates to increase the items from the successful representation. Experimental results show that as the RF iterations progresses the system converges to an optimal state in terms of representations.

In the second representation feedback method, a hyper-vector aggregated from the different representations is used and the system selects the appropriate parts of the hyper-vector using the user preference. We report from the experimental results that the hyper-vector representation can surpass the accuracy of individual representations for some of the datasets which shows that the aggregate vector can be more significant than the individual parts.

These on-the-fly representation selection methods can enhance the precision of retrieval systems especially for dynamic data scenarios. We also note that these mechanisms can tackle the problem of multiple user groups and multiple user goals.

As a second extension, we propose the adoption of autoencoders to learn representations directly from the data instead of transforms developed by humans. While learning the representations, the autoencoder network is forced to compress the time series into a more compact form by varying the parameters of the structure and the training objective.

We show, both theoretically and empirically with the experimental results, that the computational load of the retrieval system decreases significantly with negligible degradation in accuracy. We also observe that for some specific cases, the use of the sparser data-aware representation can out-perform the full time series case in terms of accuracy while still benefiting the decrease in computational load.

Chapter 5

Temporal Graphs with Sparse Time Series

This chapter studies two relatively new aspects of temporal data: sparsity and networked time series. Sparse series is generally non-uniformly sampled time series data which can be caused by either missing data or because of the nature of the process. We direct our attention to sparsity based on missing or incomplete time series data with an application that lacks the full time series data because of scarce number of data recorders.

Networked temporal data specifies an underlying graph with time series as its properties or edges. To study different aspects of networked temporal data, we generate time-varying graph data using vehicle trajectories in a road network. A query we study is computation of shortest paths, which has been a focus of theoretical research interest for years with the most important application in route planning in road networks. The continuously increasing demand for mobility and the growth of online routing services promises ever increasing interest for these algorithms and systems.

Previous research on graphs have focused mainly on static features and calculation of network properties based on static properties of the nodes.

However, analysis of the time varying properties of the network has a lot of potential real world applications. In this context, we can define a time varying network where each network node is denoted with v_i, where i stands for the numeration of the nodes. We can use a directional model for the interactions between nodes and denote an edge going from v_i to v_j as e_{ij}^{α} which extends the frequent notations used for graphs. Since there can be numerous forms of relations between two particular nodes, we have used α for the α^{th} relation between these two particular nodes v_i and v_j. Each of these relations is also time varying which means that each e_{ij}^{α} is a time series on its own i.e. $e_{ij}^{\alpha} = [\cdots \; e_{ij}^{\alpha}(t-1) \; e_{ij}^{\alpha}(t) \; e_{ij}^{\alpha}(t+1) \; \cdots]$. We also see properties of the node itself which are captured in our model with the property set for each node i: $p_i^1, p_i^2, \cdots, p_i^{\beta}$. These parameters are also captured in time, each being a time series: $p_i^{\beta} = [\cdots \; p_i^{\beta}(t-1) \; p_i^{\beta}(t) \; p_i^{\beta}(t+1) \; \cdots]$. An example network with three nodes is depicted in Figure 5.1 showing the related properties for the nodes and the relationships between the nodes.

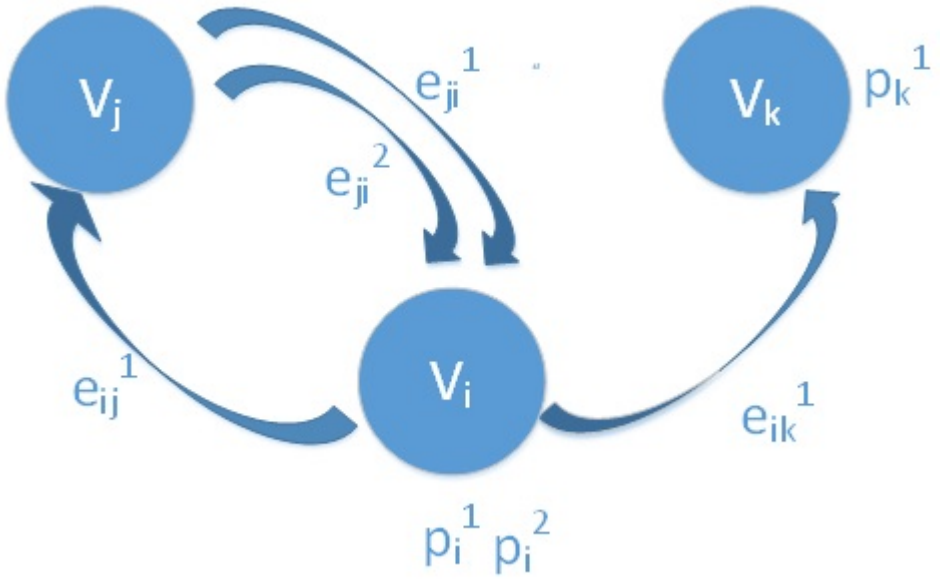


Figure 5.1: Sample network

The road network generated in this work is a specific case of the defined model with nodes being the road interconnections and the edges being the time varying travel durations as the single bi-directional relation between nodes. α being the

total number of relations per edge is one and β, the total number of properties per node, is zero in our case.

We describe our data interpolation systematic using the available sparse GPS trajectories followed by the data management system provided for the research community. We also present experiments to show the effect of static and time varying road network graphs on the shortest path solutions.

5.1 Data Model

This section describes the raw dataset that we use for further processing to generate the dataset. We also present the time varying graph model used in the process along with its parameters.

5.1.1 Trajectory Dataset

We have used the Floating Car Dataset provided under Telecom Italia Data Challenge as the underlying raw data to generate the time varying graph dataset. This dataset contains 65,956,914 different trajectories for the 61 days between March and April, 2015 for the city of Milan [4]. The trajectory traces include latitude-longitude pairs, time and speed information covering the rectangular area between the minimum and maximum latitude-longitude pairs within 45.3335945°N, 8.9415892°E and 45.5725183°N, 9.376938°E (Figure 5.2).

Figure 5.3 shows the distribution of the collected Telecom Italia trajectories over time. It depicts the percentage of all the trajectories as y-axis and the time slots in a day as x-axis. The figure shows that the density of the trajectories increases during the rush hours, i.e., between 07:30-10:00 and 17:00-19:30, and reaches to the maximum level at 18:15. The trajectories captured during rush hours cover approximately 38% of the total. The time slots having the least number of trajectories belong to the night hours, from 00:00 to around 05:00,

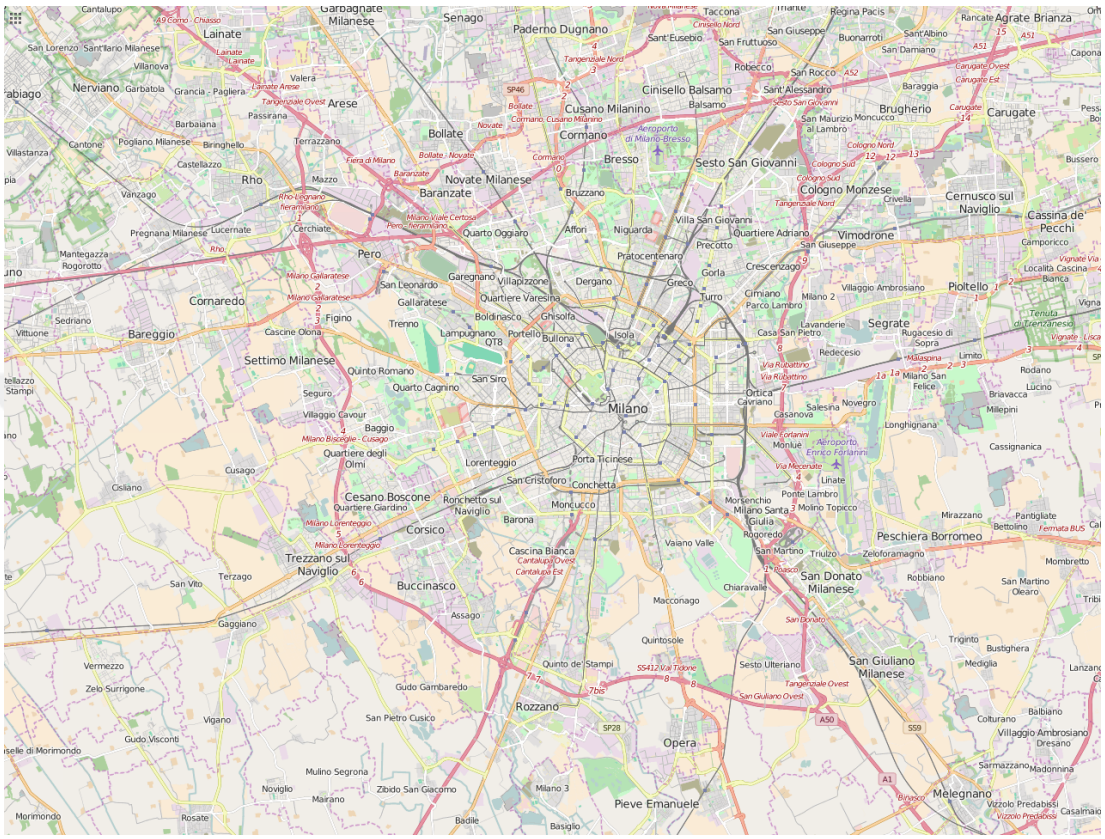


Figure 5.2: Road Map of Milan

and reaches the minimum level at 03:55.

5.1.2 Time Varying Graph Structure

We generate the road network as a directed graph with vertices as latitude-longitude pairs and edges with both fixed and time-dependent weights. For time-dependent weights, we seek to identify accurate travel time values for 288 time slots in a day starting from 00:00 to 23:55. In the static weight case the average travel time of the edge is considered.

We use OpenStreetMap (OSM) [70] to gather the topology of the underlying road network which provides an XML file based on given geo-location boundaries with latitude and longitude values along with sequences of roads. We build the

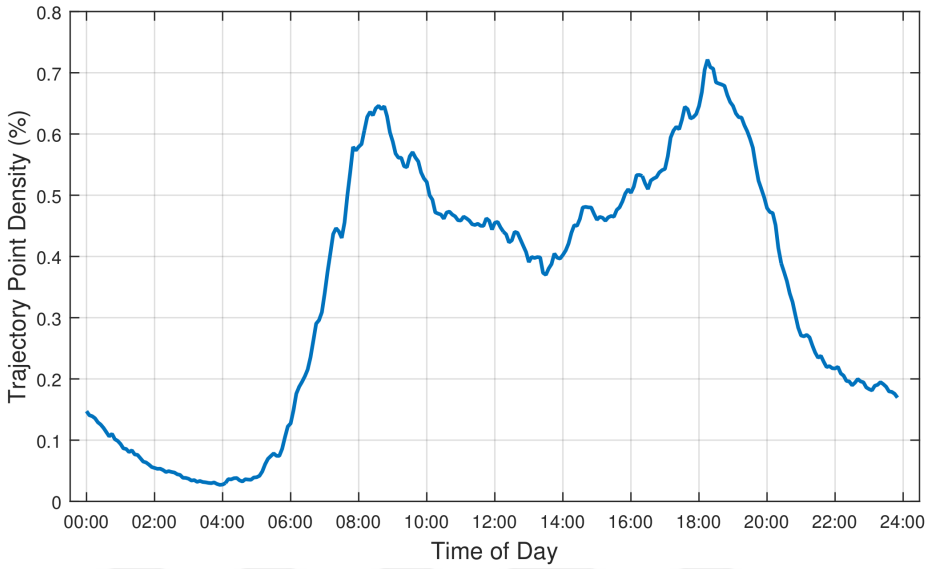


Figure 5.3: Trajectory Density over Time

network by representing the road/street intersections and end points as vertices (v_i) and the road/street segments as edges (e_i). We also treat each direction of double roads as different edges which generates a bi-directional graph.

For storage and manipulation of the data, we employ Sparksee, formerly known as DEX, which is a scalable graph database [6]. Figure 5.4 shows the layers of the system structure.

5.2 Sparse Time Series Interpolation Process

In this section, we explain our process to generate the time-varying graph from the sparse floating car database. We first clean the data, build the road network and match the GPS traces with the underlying network to form a sparse time series dataset for the network. We then estimate the incomplete parts of the dataset using time series analysis approaches.

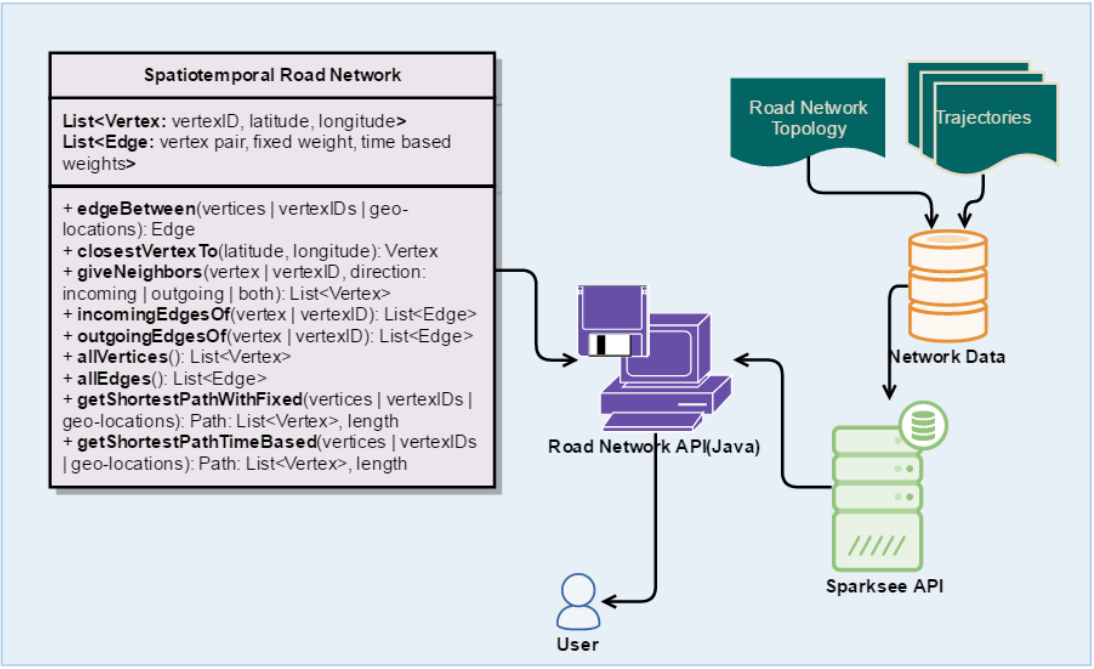


Figure 5.4: System Structure

5.2.1 Data Preparation

We aim to assign accurate time-varying travel times to the edges of the underlying graph using the trajectory dataset. After an initial analysis, we observe that the GPS trajectories are typically noisy and sparse, thus, careful examination and cleansing are required. For example, we observe that 33% of the trajectories (21,706,508) have a speed of less than 10 km/h in our dataset. Around 27% of this subset is considered noise in terms of our application, corresponding to the starting of the engine or finalization of the trip which does not represent the actual traffic. The remaining data are captured during an active driving session and represent real traffic conditions such as waiting at the traffic lights or an interruption due to an obstruction on the road. We distinguish these two cases via interpolation based on the location and speed information of two consecutive traces of the same trajectory, x' and x. We remove the data for the starting and ending traces of the trip. As a final step, we apply the following modifications in order to detect unusual decelerations or stops, which also do not represent real traffic conditions, and might be caused by instantaneous circumstances:

$$
\begin{aligned}
&\forall\ trajectory\ traces\ x\ with\ v < 10\ km/h: \\
&\quad v' = \Delta distance(x', x)/\Delta time(x', x) \\
&\quad v = \begin{cases} v' & v' > v \\ v & otherwise \end{cases}
\end{aligned}
\tag{5.1}
$$

where $\Delta distance(x', x)$ is the physical distance between two traces, x', and x. $\Delta time(x', x)$ is the time spent to arrive from the location of trace x' to that of trace x. v is the recorded speed provided by the trace, and v' is the speed computed according to the movement and time difference between x' and x. If the calculated speed value indicates that the vehicle is faster, we replace the recorded speed v with movement-based speed v'.

To summarize, we calculate a synthetic speed value considering the distances from the current trace to the previous and the next traces of the driver and the time spent. We then compare this value with an actual recorded value. If there is a significant difference between these two values, we consider this trace as a sudden stop/break and update the speed value to the calculated speed.

To efficiently match the trajectory points with the edges, we partition the graph into spatial subgraphs by exploiting the planarity of the road network. We compute the geographically closest edge for each trajectory point in the corresponding subgraph. The double-ways are represented by two edges with almost the same distance to the trajectory points. The direction is disambiguated using the previous road matches for that trajectory, if available. If it is not available, we make the direction assignment based on the geographic position of the point.

5.2.2 Sparsity Analysis of Time Series

After we match all trajectory points with the corresponding network, we sort the trajectory points of each edge by date and time. For each edge, we divide each day

into 288 time slots, i.e., with 5-minute periods in a day from 00:00 to 23:55 and distribute each trajectory point of the corresponding edge to the corresponding time slot.

The different dates show similar patterns for all the days of the week. Figure 5.5 depicts the autocorrelation function for a sample edge. The figure clearly shows an autocorrelation of the signal within a day lag with peaks at one day and at multiples of a day. This strengthens our intuition that the data has a daily periodicity with a relatively high level of confidence. However, the autocorrelation function does not exhibit a clear weekly period since we do not observe a significantly higher peak at lag seven (day).

We observe that the patterns for weekdays and weekends is similar, which apparently seems counter-intuitive. Thus, in practice, we do not observe any significant difference in Milan. Hence, in order to avoid space inefficiency and lessen sparsity, we merge the same time slots of all days' (including weekdays and weekends) trajectory data together to form the aggregated data for each edge.

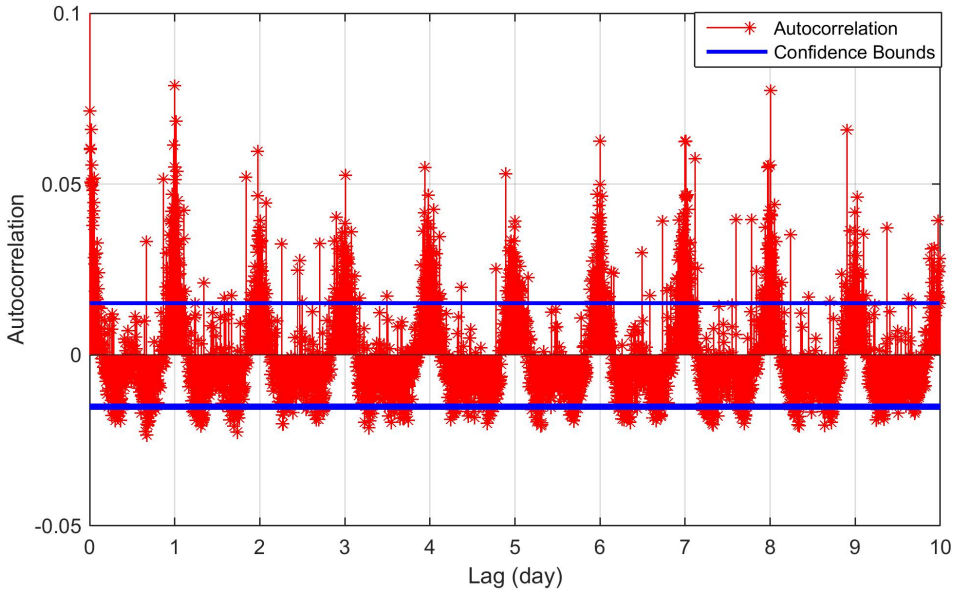


Figure 5.5: Autocorrelation of An Edge Data

Note that we do not have any value for some time slots of different edges due to the sparsity of the traces. Figure 5.6 represents the load ratio of edges

with respect to the time slots. The x-axis represents the time slots while y-axis (load ratio) shows the percentage of edges that have travel time values based on existing real data for the corresponding time slot. The time slots having the maximum, 37%, and the minimum, 3.5%, load factors correspond to 18:20 and 03:55, respectively. This figure clearly shows the level of sparsity of the data. Using commercial providers' data with larger amounts of data per time slot, like TomTom [71], would lead us to larger load ratios. In our case, the obtained ratios expose the need for further estimations as we explain in the following sections.

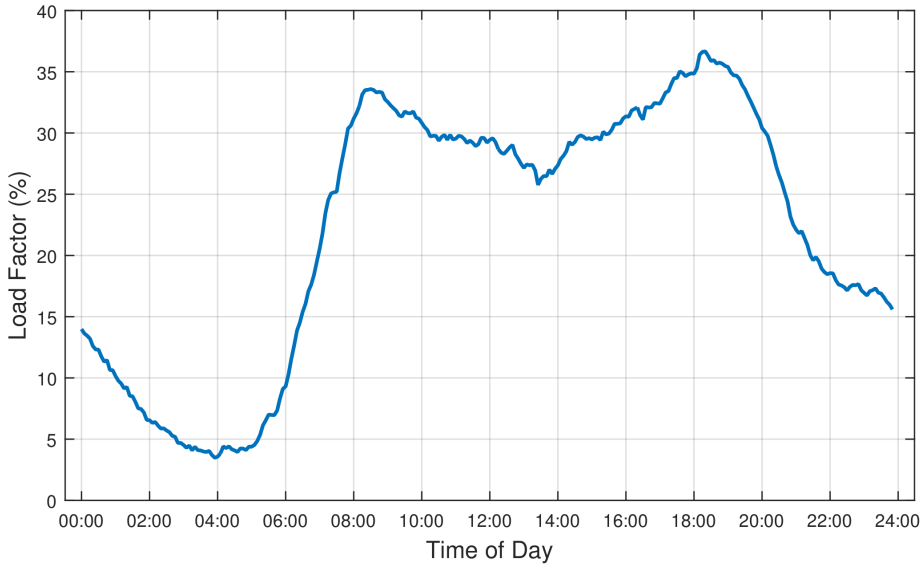


Figure 5.6: Sparsity of time series related with edges

We also need to identify outlier speed values, i.e., recorded data not due to traffic but due to various other reasons, such as cars run over the speed limits, etc. To detect and exclude those outliers we apply Generalized Extreme Studentised Deviate (ESD) Test [72], which is a generalization of Grubb's Test for more than one outlier. Once the outliers are removed, we expect that the remaining speed values enable us to have a better understanding of the traffic condition at each time slot. Because we employ travel time as weights in the time-varying graph, we aggregate the length of the edges from OSM, then divide it by the speed values.

5.2.3 Interpolation and Filtering

We need to address two major issues to finalize the temporal graph with the resultant vector with 288 weights spanning a day with 5-minute intervals:

- **Missing Data:** After forming the daily vectors, i.e., that contain 288 slots per day per edge, we observed that 77.68% of the vector slots do not contain any data.
- **Noise:** The data after the aggregation stage has noise, i.e., high frequency components apparent as jumps in the data, associated with it.

To address these issues we compute the frequency spectrum of the most populous edges to give an idea about the nature of the time series involved. The average power of the respective frequencies is plotted in Figure 5.7.

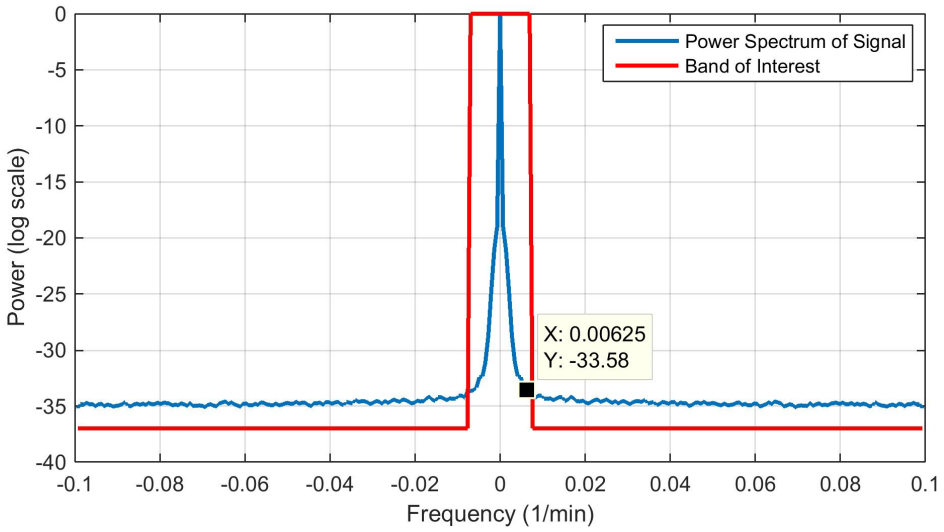


Figure 5.7: Average Frequency Spectrum of The Most Populous 4000 Edges

The spectrum shows a band-limited signal with the majority of the content in the low-pass region. The other components out of the region are the noise that we observe in the aggregated signal. The signal of interest is depicted with a red band in Figure 5.7. This shows us that the signal is band-limited. The cutoff

frequency for the signal is $0.0069/min$ which we select according to the noise level.

Algorithm 5 Algorithm for interpolation and filtering

Input:
1) *S: Incomplete time series for the edge of the given network*
2) δ*: Minimum travel time for edge n*
Output: *S: Completed time series for the edge of the given network*

procedure DataInfo(S, δ)
$S[n]$, time series for the respective edge
Calculate μ and σ from the complete data
Define $I \triangleright$ *a set for missing data intervals*
for $n = 1 : size(S)$ **do**
    $I[n] \leftarrow MissingDataIntervals(S[n]) \triangleright$ *the time slots of* $S[n]$ *having no data*
end for
$S'[n] = S[n]$
for all $I[n] \in I$ **do**
    if $Len(I[n]) \geq 14$ **then**
        $Fill\ I[n]\ with\ \delta$
    else
        $Fill\ I[n]\ with\ samples\ drawn\ from\ \mathcal{N}(\mu, \sigma^2)$
    end if
    $Update\ S'[n]\ missing\ intervals\ with\ I[n]$
end for
$S_{LPF}[n] = Low\ pass\ filter\ S'[n]$
$Fill\ missing\ data\ of\ S[n]\ with\ S_{LPF}[n]$

According to the signals and systems theory, this signal has to be sampled with at least Nyquist rate which is the twice the highest frequency (F_{max}) in the signal [73]. The minimum sampling interval for the signals of interest is given in Equation 5.2.

$$\begin{aligned} F_{Nyquist} &\geq 2.F_{max} \geq 0.0139 \\ T_{sampling} &\leq \frac{1}{F_{Nyquist}} \leq 72 \text{ minutes} \leq 14 \text{ samples} \end{aligned} \tag{5.2}$$

Equation 5.2 illustrates that any signal which has data with 14 consecutive missing data is undersampled and will cause aliasing. We observe from the data that this requirement does not hold for most of the edges of the data. To overcome

these problems we develop two solutions for the respective cases. If the length of consecutive missing data is larger than 14 samples we use the minimum duration for that edge. This minimum duration is calculated using the length of the edge and the speed limit where available, or the value for urban areas (50 km/h). Else, we draw a sample data from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ where the parameters μ and σ are calculated from the existent data for the edge. With these corrections in the time series, we use a low pass filter with cutoff frequency as depicted in Figure 5.7 (this filter is applied in the spectral domain) to compute the final time series. This interpolation technique is also called sinc interpolation or Whittaker-Shannon interpolation for band-limited signals. The flow of the process is outlined in Algorithm 5.

5.3 Evaluation

In this section, we compare traditional and time-dependent versions of shortest paths solutions using the developed temporal graph and system. We aim to quantify the difference in using a static weight graph and a time-varying graph in finding the shortest path applications. Since the edges of the graph have different travel time values (edge weights) for different time slots within a day, in time-dependent shortest paths (TDSP) the employed travel time (edge weight) of an edge changes depending when the path includes the edge. This information has a cascading dependence on the travel times of the previous edges. The steps of the TDSP we employ is given in Algorithm 6. The algorithm spreads from the start vertex s to the destination vertex d, by iterating until there is no new vertex to be processed, i.e., the shortest paths of all vertices that can be reached by s are discovered, or d is already found (Line 8). For the edge weight between two vertices, the algorithm uses the weight belonging to arrival time of the preceding vertex as in Line 14.

For the standard shortest path problem, we utilize the same algorithm by modifying the edge weight related parts (Line 12, 14). We discard the usage of t_{start} and get the fixed edge weights, i.e., based on the average travel time on the

corresponding edges.

5.3.1 Experimental Setup

We sample the dataset by using the start and destination locations of randomly selected 4,620 real trajectories. The distribution of the selected trajectories (Figure 5.8) is similar to the trajectory density over the time slots (Figure 5.3).

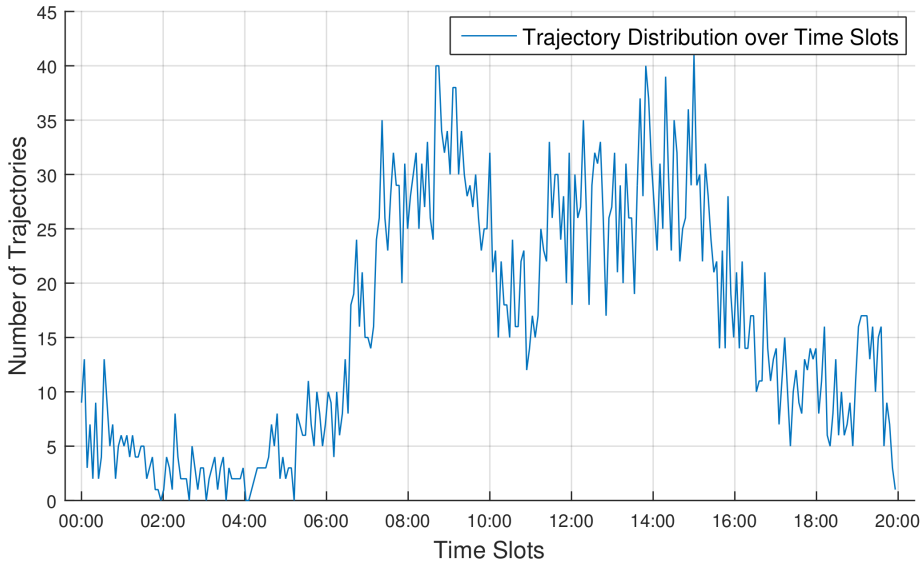


Figure 5.8: Sample Trajectory Distribution over All Time Slots

We compare the time-varying versus static weight shortest paths using two measures: similarity of resulting paths and gain of time-dependent shortest path regarding the travel time. The similarity of paths is based on the common edges of two given paths. Here we calculate the similarity between TDSP and traditional shortest path using the Jaccard distance (Eq. 5.3). Hence, the range of similarity index is [0,1].

$$sim(p_1, p_2) = \frac{edgeSet(p_1) \cap edgeSet(p_2)}{edgeSet(p_1) \cup edgeSet(p_2)} \tag{5.3}$$

where p_1 and p_2 represent paths, i.e., an ordered vertex set.

Algorithm 6 Time Dependent Shortest Path (TDSP)

Input:
1) *G(V,E): the spatiotemporal network whose each edge e has travel times for different time slots*
2) *s: source vertex over the network*
3) *d: destination vertex over the network*
4) t_{start}*: start time of desired path for s and t.*
Output: *p: a path including vertex list*

1: **procedure** TDSP(G, s, d, t_{start})
2: $p \leftarrow \emptyset$ ▷ *result path*
3: Define D ▷ *a priority queue of vertices with travel time as priority index*
4: $D[s] \leftarrow 0$
5: Define P ▷ *a list of preceding vertices*
6: $P[s] \leftarrow NIL$
7: Define S ▷ *vertices having the shortest paths*
8: **while** $D.size \neq S.size \ \vee \ d \in S$ **do**
9: $v \leftarrow ExtractMin(D)$
10: $S \leftarrow S \cup v$ ▷ *add v to discovered list*
11: **for all** $u \in outNeighbors(v)$ **do**
12: **if** $u \notin S \wedge D[u] > D[v] + edge(v,u).weight[t_{start} + D[v]]$ **then**
13: ▷ *the index of a vertex not included in D is* $+\infty$
14: $D[u] \leftarrow D[v] + edge(v,u).weight[t_{start} + D[v]]$ ▷ *update travel time to u*
15: $P[u] \leftarrow v$ ▷ *update preceding of u*
16: **end if**
17: **end for**
18: **end while**
19: **if** $d \in D$ **then**
20: $cur \leftarrow d$
21: **while** $cur \neq NIL$ **do**
22: $p.addHead(cur)$
23: $cur \leftarrow P[cur]$
24: **end while**
25: **end if**
26: **return** p

The gain on the path length, or more accurately path duration is as follows:

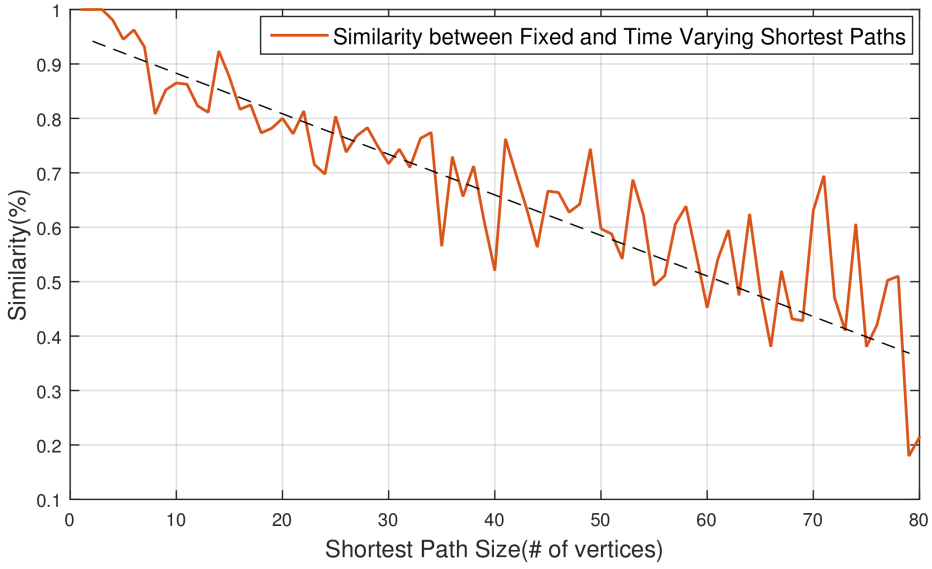
$$\frac{\Theta(SP_f(s,d),t) - \Theta(SP_{tv}(s,d),t)}{\Theta(SP_f(s,d),t)} \tag{5.4}$$

where the Θ function computes the duration of the path starting at t. SP_{tv} and SP_f are the shortest paths obtained with the time-varying and fixed weights, respectively. For each query, i.e. source-destination pair (s-d), we measure how much gain one would get using a time-varying network instead of employing the static network. For simplicity, we denote the path retrieved with time-varying weights as SP_{tv} and the one with fixed weights as SP_f.

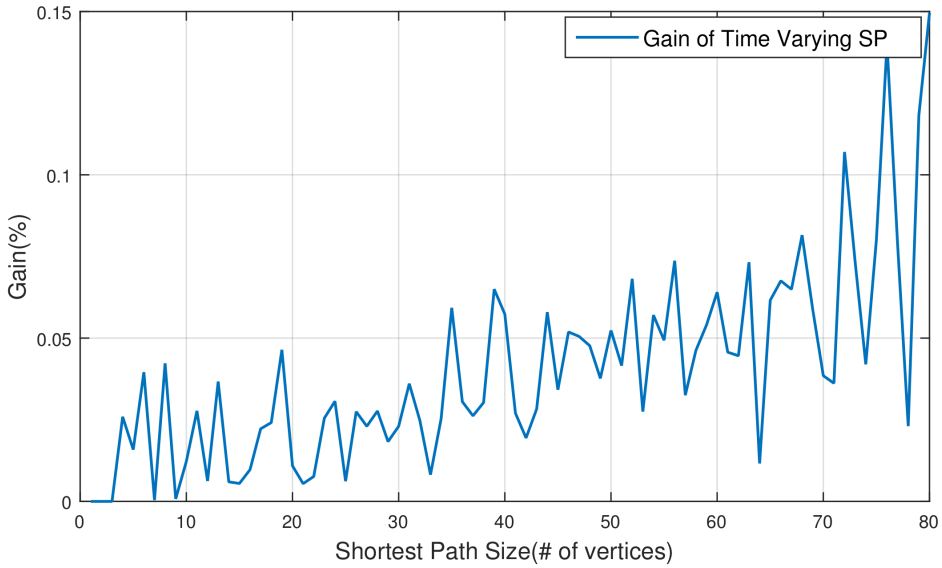
5.3.2 Experimental Results

Figure 5.9 presents the result of each measure with respect to path sizes, i.e., number of vertices of the paths, ranging from 1 to 80 for our road network. The average similarity ranges from 0.18 to 1, for path sizes of 79 and 5, respectively. Similarity between SP_{tv}s and SP_fs decreases as the path size increases (Figure 5.9a). Note that the fluctuations on the figures stem from varying the number of samples we have for each path size. For example, there are 161 paths of length 23 vertices, and there are only 2 of length 76. Figure 5.9b illustrates that the shortest path queries over time-varying network get higher gains, i.e., SP_{tv} has much lower travel time than SP_f, as the number of vertices increases. Note that the gain cannot be lower than 0, because if SP_f had lower travel time at t, SP_{tv} would be the same as SP_f according to TDSP algorithm (Algorithm 6). In other words, if a path for a query provides more benefit, i.e., less travel time, than any other paths starting at the same time, the path would be the resultant path of TDSP. For the temporal graph the gain increases to 0.14, for the path with 80 vertices.

Figure 5.10 shows the results related to one query started at different times to show the changes in the result paths. We choose a representative query with an average of 37 vertices in all its resulting paths having the start times with half-hour intervals. It means there are 48 different queries with the same source-destination pair yet different start times. The x-axis of the figure represents the



(a) Similarity of Time-varying Shortest Paths vs Static Paths



(b) Gain of Time-varying Shortest Paths over Static Paths w.r.t. Path Length

Figure 5.9: Path Size based Analysis

start times of the query. The red line illustrates the similarity index between SP_{tv} and SP_f with the corresponding start times. Similarity between SP_{tv}s and SP_f varies in time reaching exact paths in some cases, i.e., similarity index 1, and not exactly equal paths in other cases, with a worse case similarity index 0.5. Additionally, we present the similarity between the consecutive shortest paths with the green dashed line. According to the results, all result paths are at least 45% similar. In total, 15 different paths are retrieved and SP_f is observed in the time-varying shortest paths twice. On average, the consecutive paths are similar in an hour period, probably because of the similar traffic patterns, in the period. After the period, the selected path at the next start time change drastically, i.e., not smoothly any more. It may indicate the changes in the traffic conditions may not evolve gradually. The paths belonging to night hours, i.e., 22:00-6:00, do not change as much as the ones between 7:00 and 18:00.

The results confirm that time-varying shortest paths can reveal alternative paths with shorter time routes and the paths are not necessarily similar to the paths computed in static networks.

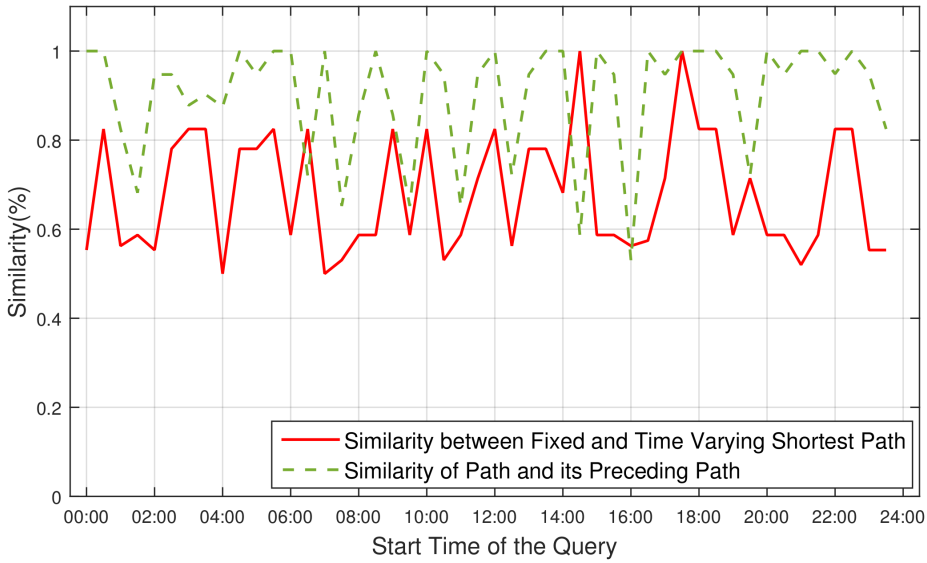


Figure 5.10: Comparisons on Time-Varying Paths for the Same Query with Different Start Times

5.4 Conclusion

We presented a methodology to generate a time varying graph using a sparse raw data to share with fellow researchers interested on time-varying graphs in the context of road networks.

The applied data generation method consists of the following steps:

- matching the traces of a sparse GPS data with the underlying road network
- analyzing the time series properties and statistical features to define the parameters of the the time-varying graph
- aggregating the matched data to a single day version with 5 minutes sampling interval and cleaning the data with respect to our end application
- develop a time-series interpolation model for inferring and estimating the missing data

In the interpolation phase, we analyze the near-full edges to understand the nature of the signal using the spectral content of available time series data. We fill in the missing data using two different values based on the length of missing portion. After the complete time series is formed, we apply a band-limited filter to smooth the signal with respect to the model that we have seen in near-full edges.

We utilize a scalable database, Sparksee, for the storage and manipulation of the resultant graph. We also provide a Java API for easy utilization for research purposes. Besides the basic operations on graphs such as retrieving edges and vertices, we also include the shortest path computation methods for time-varying and static weighted versions.

We employ the system in a use case, evaluating the time-varying shortest path solution. We observe that the difference between the shortest paths using time-varying versus fixed weights increases as the number of vertices increases. The

travel durations of these two types of shortest paths give different results favoring the use of the time-varying road network.

Our work is a step towards generating data and systems for analysis and management of time-varying graphs. One of the future directions is to build larger time-varying graphs, e.g., a road network with a larger amount of GPS traces. One can feed the database with more trajectory data and increase the ratio of the real data against the synthetic (estimated) data; redistribute trajectories considering each day of the week independently, instead of merging all trajectories into one sample day. An accurate and scalable dataset and graph manipulation tool can be used in smart city applications, route recommendation systems and location based services.

One can also model social networks as temporal graphs with nodes being the users and edges between the follow relation and retweet relation. Each node can also have a number of parameters e.g. number of retweets, number of followers, influence of users, measures of centrality which are all time dependent and calculation of these parameters for time varying networks can be an important extension of this study. As an example, estimation of time varying influence is an application in social networks especially under challenging conditions like limited network access [74]. Approximating the whole network to correctly track the different time varying properties is also an important future research possibility.

Chapter 6

Conclusion and Future Work

Temporal data is created, stored, processed and analyzed in a wide range of applications and the present momentum is increasing with the growing number of data sources. There is a wealth of efforts to make the most out of this data to enhance our understanding of temporal relations and also present the tools to extract the required knowledge by the system users. Time series literature has an extensive coverage for traditional data analytics tasks like classification and clustering. We also see a unique side of time series data mining with different transformations, various distance measures and forecasting applications.

This dissertation deals with two important applications for time series data introducing adaptations from various fields. Our initial focus is on increasing the accuracy of retrieval systems by exploiting the user annotations in different perspectives. We also explore ways to reduce the computational load of such methods. Secondly, we concentrate on generating a temporal graph dataset with time series data as edges to form an open dataset to be used for validation of similar applications by generating it from a significantly sparse data source.

We first build a retrieval system which utilizes relevance feedback and diversity amongst the result set to fulfill the user requirements. We propose a framework to improve the accuracy and user satisfaction while illustrating the statistical

background of the proposed perspective. A wide range of experiments with real datasets demonstrates precision improvements even in a single iteration of user feedback confirming the suitability of our approach. Including diversity within the result set achieves accuracy improvements.

We further extend the time series RF framework by proposing a representation feedback method automating the representation selection process. This is an unique aspect of time series data and the chosen representation can affect the end result of the retrieval task significantly. Among the two methods proposed, the top-k partitioning method forms the result set from a variety of different representation top-k results and weighting based feedback forms a hyper-vector by concatenating different representations. In both of the methods, the system self-regulates the selected items according to the user feedback to increase the items relevant to the user. Experimental results show that as the RF iterations progresses the system converges to an improved state in terms of representations. Results also illustrate that representation feedback diversifies the result set implicitly because of the use of various representations, which in turn improves the precision even in a case of simple nearest neighbor retrieval. The proposed on-the-fly representation selection methods can enhance the precision of retrieval systems especially for dynamic data scenarios.

We develop an adoption of autoencoders by analyzing the data, to learn data-aware representations to be used in the RF framework, while also compressing the time series into a more compact form. This compressed sparse representation decreases the computational load of the system significantly. We also report that for some challenging data cases, data-aware representation can outperform the full time series case in terms of accuracy in addition to the decrease in computational load. An autoencoder trained with combinations of representations as input has yielded meaningful performance improvements which shows the potential of this approach.

We are witnessing a rise in time series data coupled with an underlying network structure and incomplete time series raw datasets. An important special case is a time-varying road network with edges constituting of time dependent travel

durations between nodes. We generate the time varying graph data by cleaning and processing a sparse GPS trajectory dataset. We have analyzed the time series properties and statistical features of the trajectories to develop a time-series interpolation model for inferring and estimating the missing data. We evaluated the generated temporal graph by comparing the shortest paths using time-varying versus static weighted graph.

In the light of the results of this dissertation, the following interesting research topics arise which can enrich the time series literature further:

- Fine tuning of the diversity balance using the parameters of MMR and CBD method by taking the dataset properties and user objectives into account can increment the improvements of the proposed RF framework. The promising performance of the proposed CBD diversification method can be extended to other data types such as text or image for similar purposes.

- We have reported that autoencoders with MLP structure have been successfully adopted. There is potential in using other neural network structures to extract data-aware representations. Use of autoencoder, and even stacked-autoencoders with larger datasets, to extract useful representations is a potential direction for further research. To this end, a universal time series autoencoder deep network can produce promising results and may surpass the performance of human crafted representations.

- The time-varying graph dataset generated in our study can be enhanced by building a larger time-varying road network with larger amount of GPS traces from heterogeneous data sources. This will provide the research community with a more accurate real dataset for validating new ideas.

- Incremental update methods to develop the time varying graph on-the-fly as data arrives is also an important research area. This will allow the use of a continuously revised and up-to-date state of the road network to be used in route planning applications increasing its effectiveness.

- Efficient and accurate calculation of time varying network parameters such as centrality can further enhance our understanding of networks and pave way for time dependent analysis of clusters withing networks. There is also prospect of improving accuracy of time series forecast tasks by taking the underlying network into account as proposed under the general time varying graph framework.

Bibliography

[1] Wikipedia, "Dual-tree complex wavelet transform figure," 2005.

[2] B. Eravci and H. Ferhatosmanoglu, "Diversity based relevance feedback for time series search," *PVLDB*, vol. 7, no. 2, pp. 109–120, 2013.

[3] B. Eravci and H. Ferhatosmanoglu, "Diverse relevance feedback for time series with autoencoder based summarizations," *IEEE Transactions on Knowledge and Data Engineering*, 2018. (accepted for publication).

[4] "Source of the dataset: Tim big data challenge 2015." `www.telecomitalia.com/bigdatachallenge`. Accessed: 2015-06-1.

[5] E. Eser, F. Kocayusufolu, B. Eravci, H. Ferhatosmanolu, and J. L. Larriba-Pey, "Generating time-varying road network data using sparse trajectories," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 1118–1124, Dec 2016.

[6] N. Martínez-Bazan, V. Muntés-Mulero, S. Gómez-Villamor, J. Nin, M.-A. Sánchez-Martínez, and J.-L. Larriba-Pey, "Dex: high-performance exploration on large graphs for information retrieval," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 573–582, ACM, 2007.

[7] T.-C. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164 – 181, 2011.

[8] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of*

the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03, (New York, NY, USA), pp. 2–11, ACM, 2003.

[9] D. J. Berndt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," in *Knowledge Discovery and Data Mining*, pp. 359–370, 1994.

[10] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani, "Mining the stock market (extended abstract): which measure is best?," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 487–496, 2000.

[11] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *Proc. VLDB Endow.*, vol. 1, pp. 1542–1552, Aug. 2008.

[12] A. Camerra, T. Palpanas, J. Shieh, and E. J. Keogh, "isax 2.0: Indexing and mining one billion time series," in *Proceedings of the The 10th IEEE International Conference on Data Mining (ICDM 2010)*, pp. 58–67, 2010.

[13] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, pp. 69–84, 1993.

[14] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space.," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.

[15] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos, "Approximate embedding-based subsequence matching of time series," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, (New York, NY, USA), pp. 365–378, ACM, 2008.

[16] S. Rhea, E. Wang, E. Wong, E. Atkins, and N. Storer, "Littletable: A time-series database and its uses," in *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, (New York, NY, USA), pp. 125–138, ACM, 2017.

[17] A. L.-C. Wang, C. J. P. Barton, D. S. Mukherjee, and P. Inghelbrecht, "Method and system for identifying sound signals," May 2014. US Patent 8,725,829.

[18] J. Rocchio, *Relevance feedback in information retrieval. In: The SMART Retrieval System Experiments in Automatic Document Processing.*, pp. 313–323. 1971.

[19] G. Salton, ed., *The SMART Retrieval System Experiments in Automatic Document Processing.* 1971.

[20] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American Society for Information Science*, vol. 41, pp. 288–297, 1990.

[21] H. Zamani, J. Dadashkarimi, A. Shakery, and W. B. Croft, "Pseudo-relevance feedback based on matrix factorization," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, (New York, NY, USA), pp. 1483–1492, ACM, 2016.

[22] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: Querying databases through multiple examples," in *VLDB 98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pp. 218–227, 1998.

[23] X. S. Zhou and T. S. Huang, "Relevance feedback in image retrieval: A comprehensive review," *Multimedia systems*, vol. 8, no. 6, pp. 536–544, 2003.

[24] M. L. Kherfi, D. Ziou, and A. Bernardi, "Combining positive and negative examples in relevance feedback for content-based image retrieval," *J. Visual Communication and Image Representation*, vol. 14, no. 4, pp. 428–457, 2003.

[25] Y. Rui, T. S. Huang, S. Mehrotra, and M. Ortega, "A relevance feedback architecture for content-based multimedia information retrieval systems," in *Content-Based Access of Image and Video Libraries, 1997. Proceedings. IEEE Workshop on*, pp. 82–89, IEEE, 1997.

[26] Z. Su, H. Zhang, S. Li, and S. Ma, "Relevance feedback in content-based image retrieval: Bayesian framework, feature subspaces, and progressive learning," *Image Processing, IEEE Transactions on*, vol. 12, no. 8, pp. 924–937, 2003.

[27] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," in *Proceedings of the ninth ACM international conference on Multimedia*, MULTIMEDIA '01, pp. 107–118, 2001.

[28] J. Carbonell and J. Goldstein, "The use of mmr, diversity-based reranking for reordering documents and producing summaries," Proc. of SIGIR '98, pp. 335–336, 1998.

[29] A. Borodin, A. Jain, H. C. Lee, and Y. Ye, "Max-sum diversification, monotone submodular functions, and dynamic updates," *ACM Trans. Algorithms*, vol. 13, pp. 41:1–41:25, July 2017.

[30] C. Charles, K. Maheedhar, C. Gordon, V. Olga, A. Azin, B. Stefan, and M. Ian, "Novelty and diversity in information retrieval evaluation," in *Proceedings of SIGIR*, SIGIR '08, pp. 659–666, 2008.

[31] D. Rafiei, K. Bharat, and A. Shukla, "Diversifying web search results," in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, (New York, NY, USA), pp. 781–790, ACM, 2010.

[32] H. Chen and D. R. Karger, "Less is more: probabilistic models for retrieving fewer relevant documents," in *Proceedings of SIGIR*, SIGIR '06, pp. 429–436, 2006.

[33] M. Hasan, A. Mueen, and V. Tsotras, "Distributed diversification of large datasets," in *2014 IEEE International Conference on Cloud Engineering*, pp. 67–76, March 2014.

[34] X. Zuobing, A. Ram, and Z. Yi, "Incorporating diversity and density in active learning for relevance feedback," in *Proceedings of the 29th European conference on IR research*, ECIR'07, pp. 246–257, 2007.

[35] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, "Diversifying search results," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pp. 5–14, 2009.

[36] F. Radlinski and S. Dumais, "Improving personalized web search using result diversification," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pp. 691–692, 2006.

[37] O. Kucuktunc and H. Ferhatosmanoglu, "l-diverse nearest neighbors browsing for multidimensional data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 3, pp. 481–493, 2013.

[38] B. Liu and H. V. Jagadish, "Using trees to depict a forest.," *PVLDB*, vol. 2, no. 1, pp. 133–144, 2009.

[39] F. Radlinski and T. Joachims, "Active exploration for learning rankings from clickthrough data," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, (New York, NY, USA), pp. 570–579, ACM, 2007.

[40] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), pp. 784–791, ACM, 2008.

[41] T. Joachims, A. Swaminathan, and T. Schnabel, "Unbiased learning-to-rank with biased feedback," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, (New York, NY, USA), pp. 781–789, ACM, 2017.

[42] K. Hofmann, S. Whiteson, and M. de Rijke, "Balancing exploration and exploitation in learning to rank online," in *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, (Berlin, Heidelberg), pp. 251–263, Springer-Verlag, 2011.

[43] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, pp. 225–331, Mar. 2009.

[44] N. Rubens, D. Kaplan, and M. Sugiyama, "Active learning in recommender systems," in *Recommender Systems Handbook* (P. Kantor, F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 735–767, Springer, 2011.

[45] E. Keogh and M. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," in *Proceedings of 4th KDD*, 1998.

[46] E. Keogh and M. J. Pazzani, "Relevance feedback retrieval of time series data," in *Proceedings Of The 22 Th Annual International ACM-SIGIR Conference On Research And Development In Information Retrieval*, pp. 183–190, 1999.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," ch. Learning Internal Representations by Error Propagation, pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.

[48] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[49] N. Gianniotis, S. D. Kgler, P. Tio, and K. L. Polsterer, "Model-coupled autoencoder for time series visualisation," *Neurocomputing*, vol. 192, pp. 139 – 146, 2016.

[50] N. Ahmed, A. Atiya, N. E. Gayar, and H. El-Shishiny, "An Empirical Comparison of Machine Learning Models for Time Series Forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[51] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artificial Intelligence in Medicine*, vol. 45, no. 1, pp. 11 – 34, 2009.

[52] S. Baisch and G. H. Bokelmann, "Spectral analysis with incomplete time series: an example from seismology," *Computers & Geosciences*, vol. 25, no. 7, pp. 739 – 750, 1999.

[53] M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. R. Pota, "Fast prediction for sparse time series: Demand forecast of ev charging stations for cell phone applications," *IEEE Transactions on Industrial Informatics*, vol. 11, pp. 242–250, Feb 2015.

[54] P. Mrazovic, B. Eravci, J. L. Larriba-Pey, H. Ferhatosmanoglu, and M. Matskin, "Understanding and predicting trends in urban freight transport," in *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, pp. 124–133, May 2017.

[55] D. Schultes, *Route Planning in Road Networks*. Saarbrücken, Germany, Germany: VDM Verlag, 2008.

[56] B. George and S. Shekhar, "Time-aggregated graphs for modeling spatio-temporal networks," in *Advances in conceptual modeling-theory and practice*, pp. 85–99, Springer, 2006.

[57] S. E. Dreyfus, "An appraisal of some shortest-path algorithms," *Operations research*, vol. 17, no. 3, pp. 395–412, 1969.

[58] H. Wang, G. Li, H. Hu, S. Chen, B. Shen, H. Wu, W.-S. Li, and K.-L. Tan, "R3: a real-time route recommendation system," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1549–1552, 2014.

[59] J. Letchner, J. Krumm, and E. Horvitz, "Trip router with individualized preferences (trip): Incorporating personalization into route planning," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, p. 1795, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[60] L. Cao and J. Krumm, "From gps traces to a routable road map," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pp. 3–12, ACM, 2009.

[61] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and travel time inference from gps probe vehicle data," *NIPS Analyzing Networks and Learning with Graphs*, vol. 12, no. 1, 2009.

[62] N. Kumar, N. Lolla, E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Time-series bitmaps: a practical visualization tool for working with large time series databases," in *SIAM 2005 Data Mining Conference*, pp. 531–535, SIAM, 2005.

[63] I. Selesnick, R. Baraniuk, and N. Kingsbury, "The dual-tree complex wavelet transform," *Signal Processing Magazine, IEEE*, vol. 22, pp. 123 – 151, nov. 2005.

[64] J. Miao, J. X. Huang, and Z. Ye, "Proximity-based rocchio's model for pseudo relevance," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, (New York, NY, USA), pp. 535–544, ACM, 2012.

[65] I. Ruthven and M. Lalmas, "A survey on the use of relevance feedback for information access systems," *Knowl. Eng. Rev.*, vol. 18, pp. 95–145, June 2003.

[66] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana, "The ucr time series classification/clustering homepage," 2011.

[67] S. Cai and K. Li, "Dual-tree complex wavelet transform matlab code," 2003.

[68] A. K. Tung, R. Zhang, N. Koudas, and B. C. Ooi, "Similarity search: a matching based approach," in *Proceedings of the 32nd international conference on Very large data bases*, pp. 631–642, VLDB Endowment, 2006.

[69] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 25, no. 3, pp. 235–238, 1977.

[70] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[71] "Tomtom city." `http://www.tomtom.com/en_gb/traffic-news/`. Accessed: 2016-07-1.

[72] "Michael thomas flanagan's java scientific library." `http://www.ee.ucl.ac.uk/~mflanaga/java/`.

[73] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & Systems*. Prentice-Hall signal processing series, Upper Saddle River, N.J. Prentice Hall London: Prentice Hall international, 1997.

[74] K. Bingol, B. Eravci, C. O. Etemoglu, H. Ferhatosmanoglu, and B. Gedik, "Topic-based influence computation in social networks under resource constraints," *IEEE Transactions on Services Computing*, 2016. (accepted for publication).

Appendix A

Datasets

The properties of the 85 real data sets, all the data currently available in the UCR time series repository[66] is found in Table A.1.

Table A.1: Datasets used for experiments

No.	**Name**	**Number of classes**	**Time series length**	**Dataset size**
1	50Words	50	270	905
2	Adiac	37	176	781
3	ArrowHead	3	251	211
4	Beef	5	470	60
5	Beetle Fly	2	512	40
6	Bird Chicken	2	512	40
7	CBF	3	128	930
8	Car	4	577	120
9	Chlorine Concentration	3	166	4307
10	CinC ECG Torso	4	1639	1420
11	Coffee	2	286	56
12	Computers	2	720	500

13	Cricket_X	12	300	780
14	Cricket_Y	12	300	780
15	Cricket_Z	12	300	780
16	Diatom Size Reduction	4	345	322
17	Distal Phalanx Outline Age Group	3	80	539
18	Distal Phalanx Outline Correct	2	80	876
19	Distal Phalanx TW	6	80	539
20	ECG 200	2	96	200
21	ECG 5000	5	140	5000
22	ECG Five Days	2	136	884
23	Earthquakes	2	512	461
24	Electric Devices	7	96	16637
25	Fish	7	463	350
26	Face (all)	14	131	2250
27	Face (four)	4	350	112
28	Faces UCR	14	131	2250
29	Ford A	2	500	4921
30	Ford B	2	500	4446
31	Gun-Point	2	150	200
32	Ham	2	431	214
33	Hand Outlines	2	2709	1370
34	Haptics	5	1092	463
35	Herring	2	512	128
36	Inline Skate	7	1882	650
37	Insect Wingbeat Sound	11	256	2200
38	Italy Power Demand	2	24	1096
39	Large Kitchen Appliances	3	720	750
40	Lightning-2	2	637	121

41	Lightning-7	7	319	143
42	MALLAT	8	1024	2400
43	Meat	3	448	120
44	MedicalImages	10	99	1141
45	Middle Phalanx Outline Age Group	3	80	554
46	Middle Phalanx Outline Correct	2	80	891
47	Middle Phalanx TW	6	80	553
48	Mote Strain	2	84	1272
49	Non-Invasive Fetal ECG Thorax1	42	750	3765
50	Non-Invasive Fetal ECG Thorax2	42	750	3765
51	OliveOil	4	570	60
52	OSU Leaf	6	427	442
53	Phalanges Outlines Correct	2	80	2658
54	Phoneme	39	1024	2110
55	Plane	7	144	210
56	Proximal Phalanx Outline Age Group	3	80	605
57	Proximal Phalanx Outline Correct	2	80	891
58	Proximal Phalanx TW	6	80	605
59	Refrigeration Devices	3	720	750
60	Screen Type	3	720	750
61	Shapelet Sim	2	500	200
62	Shapes All	60	512	1200
63	Small Kitchen Appliances	3	720	750
64	Sony AIBO Robot Surface	2	70	621
65	Sony AIBO Robot SurfaceII	2	65	980

66	Star Light Curves	3	1024	9236
67	Strawberry	2	235	983
68	Swedish Leaf	15	128	1125
69	Symbols	6	398	1020
70	Toe Segmentation1	2	277	268
71	Toe Segmentation2	2	343	166
72	Trace	4	275	200
73	TwoLead ECG	2	82	1162
74	Two Patterns	4	128	5000
75	UWave Gesture Library All	8	945	4478
76	Wine	2	234	111
77	Word Synonyms	25	270	905
78	Worms	5	900	258
79	Worms Two Class	2	900	258
80	Synthetic Control	6	60	600
81	uWave Gesture Library_X	8	315	4478
82	uWave Gesture Library_Y	8	315	4478
83	uWave Gesture Library_Z	8	315	4478
84	Wafer	2	152	7174
85	Yoga	2	426	3300