Chapter 1

Introduction

With the maturity of database technologies, nowadays applications collect data in all domains at an unprecedented scale. For example, billions of social network users and their activities are collected in the form of *graphs*; Thousand sensor reports are collected per second in the form of *time series*; Hundreds of millions of temporal-locations are collected as *trajectories*, to name just a few. Flooded by the tremendous amount of data, it is emerging to provide useful and efficient analytics for various data domains. Traditional SQL analytics which comprises of operations (such as, partition, sorting and aggregation) in the relational domain become limited in non-structured domains. In SQL context, interesting analytics such as graph traversal and pattern detection often involve complex joins which are very hard to optimize without domain knowledge. In this thesis, we explore the neighborhood data analytic, which in SQL is expressed by the window function, on different domains and demonstrate how to efficiently deploy neighborhood analytic to gain useful insights.

1.1 Neighborhood Analytic

By its self-describing name, neighborhood analytic aims to provide summaries of each object over its vicinity. In contrast to the global analytics which aggregates the entire collection of data as a whole, neighborhood analytic provides a personalized view on each object per se. Neighborhood data analytics originates from the window function defined in SQL which is illustrated in Figure 1.1.

Season	Region	Sales	***sum()***	***avg()***
1	West	5100	***5100***	***5100***
2	West	5200	***10300***	***5150***
3	West	5200	***15500***	***5166***
4	West	4500	***20000***	***5000***
1	East	5000	***5000***	***5000***
2	East	4400	***9400***	***4700***
3	East	4800	***14200***	***4733***
4	East	5100	***19300***	***4825***

Window of season 3

```
SELECT Season, Region, Sales,
sum(), avg(), OVER(PARTITION
BY Region
ORDER BY Season DESC)
FROM employee;
```

Figure 1.1: A SQL window function computing running sum and average of sales. The window of season-3 is highlighted.

As shown in the figure, the sales report contains five attributes: "Season", "Region" and "Sales" are the original fields, "`sum`()" and "`avg`()" are the analytics representing the running sum and average. A window function is represented by the `over` keyword. In this context, the window of a tuple o_i contains other tuples o_j such that o_i and o_j are in the same "region" and o_j's "season" is prior to o_i's. The window of season-3 for region-"West" is highlighted.

Apart from this example, there are many other usages of the window function in the relational context. Being aware of the success of the window function, SQL 11 standard incorporates "`LEAD`" and "`LAG`" keywords which offer fine-grained specifications on a tuple's window.

Despite the usefulness, there are few works reporting the window analytics in the non-relational domain. This may dues to the usage of *sorting* in relational windows. For example, in Figure 1.1, objects need to be sorted according to "Season", and then the window of each objects is implicitly formed. However, in non-relational context, sorting may be ambiguous and even undefined.

To generalize the window function to other domains, we propose the neighborhood analytics in a broader context. Given a set of objects (such as tuples in relational domain or vertexes in graph domain), the neighborhood analytic is a composite function ($\mathcal{F} \circ \mathcal{N}$) applied on every object. $\mathcal{N}$ is the *neighborhood function*, which contains the related objects of an object; $\mathcal{F}$ is an *analytic function*, which could be aggregate, rank, pattern matching, etc. Apparently, the relational window function is a special case of the neighborhood analytics. For example, window function in Figure 1.1 can be represented as $\mathcal{N}(o_i) = \{o_j | o_i.season > o_j.season \wedge o_i.region = o_j.region\}$ and $\mathcal{F} = \texttt{avg}$. Since the *sorting* requirement is relaxed, our neighborhood analytics could enrich the semantic of relational window notations and can be applied on other domains.

1.2 Scope of the Thesis

In this thesis, we explore the neighborhood analytics in different data domains. Our efforts showcase the usefulness of the neighborhood concepts in those domains and address the efficiency issues when adopting nontrivial analytics. In particular, we looked at three most prevalent data domains, namely **attributed graph**, **time series** and **trajectory**. We then categorize two intuitive neighborhood functions as follows:

Distance Neighborhood: the neighborhood is defined based on numeric distance, that is $\mathcal{N}(o_i, K) = \{o_j | \texttt{dist}(o_i, o_j) \leq K\}$, where `dist` is a distance function and K is a distance threshold.

Comparison Neighborhood: the neighborhood is defined based on the comparison of objects, that is $\mathcal{N}(o) = \{o_i | o.a_m \ \texttt{cmp} \ o_i.a_m\}$, where a_m is an attribute of object and `cmp` is a binary comparator.

There could be other types of neighborhoods with more fine-grained or more general definitions. In this thesis, we demonstrate that these two simple neighborhood definitions joint with traditional analytic functions are already versatile in generating many useful analytics in various domains.

1.3 Contributions

At a high level, this thesis entitles a bi-folded contribution. First, by sewing different $\mathcal{N}$s and $\mathcal{F}$s, three interesting neighborhood analytic queries are proposed for *graph*, *time series* and *trajectory* domains respectively. Second, this thesis deals with the technical issues in efficiently deploying corresponding analytic queries to handle data in nowadays scale. The road map of this thesis is as show in Figure 1.2.

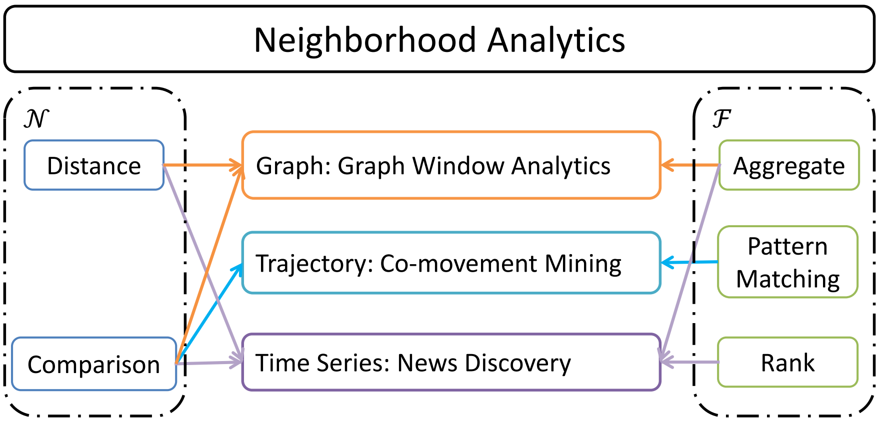


Figure 1.2: The road map of this thesis. There are three major contributions as highlighted in the center. Each contribution is a neighborhood analytic based on different $\mathcal{N}$ and $\mathcal{F}$ as indicated by arrows.

1.3.1 Graph Window Analytics

This first piece of the thesis deals with neighborhood analytics on graphs. Nowadays information network are typically modeled as attributed graphs where the vertexes correspond to objects and the edges capture the relationships between these objects. As vertexes embed a wealth of information (e.g., user profiles in social networks), there are emerging demands on analyzing these data to extract useful insights. We propose the concept of *window analytics* for attributed graph and identify two types of such analytics as shown in the following examples:

Example 1.3.1 (k-hop window)**.** In a social network (such as Linked-In and Facebook etc.), users are normally modeled as vertexes and connectivity relationships are modeled as edges. In social network scenario, it is of great interest to summarize the most relevant connections to each user such as the neighbors within 2-hops. Some analytic queries such as summarizing the related connections' distribution among different companies, and computing age distribution of the related friends can be useful. In order to answer these queries, collecting data from every user's neighborhoods within 2-hop is necessary.

Example 1.3.2 (Topological window)**.** In biological networks (such as Argocyc, Ecocyc etc.), genes, enzymes and proteins are vertexes and their dependencies in a pathway are edges. Because these networks are directed and acyclic, in order to study the protein regulating process, one may be interested to find out the statistics of molecules in each protein production pathway. For each protein,we can traverse the graph to find every other molecule that is in the upstream of its pathway. Then we can group and count the number of genes and enzymes among those molecules.

The two *windows* shown in the examples are essentially neighborhood functions defined for each vertex. Specifically, let $G = (V : E : A)$ be an attributed graph, where V is the set of vertexes, E is the set of edges, and each vertex v is associated as a multidimensional points $a_v \in A$. The *k-hop* window is a *distance* neighborhood function, i.e., $\mathcal{N}_1(v, k) = \{u|\texttt{dist}(v, u) \leq k\}$, which captures the vertexes that are k-hop nearby. The *topological* window, $\mathcal{N}_2(v) = \{u|u \in v.ancestor\}$, is a *comparison* neighborhood function that captures the ancestors of a vertex in a directly acyclic graph. The analytic function $\mathcal{F}$ is an aggregate function (sum, avg, etc.) on A.

Apart from demonstrating the useful use-cases on these two windows, we also strive to support efficient window processing. We propose two different types of indexes: Dense Block Index (DBIndex) and Inheritance Index (I-Index). The DBIndex and I-Index are specially optimized to support k-hop window and topological window processing. These indexes integrate the aggregation process with partial work sharing techniques to achieve efficient computation. In addition, we develop space and performance efficient techniques for index construction. In our experiments, DBIndex saves upto 80% of indexing time as compared to the state-of-the-art competitor.

1.3.2 Automatic News Discovery in Time Series Data

The second piece of the thesis proposes a neighborhood query to support automatic discovery of news in time series data. Nowadays events are collected in a timestamped format and journalists needs to manually synthesize those events to produce interesting news. We propose the automatic *rank-aware news theme* detection with the aim to alleviate human efforts in poring over a large amount of data. Coincidentally, the *rank-aware news theme*s are very prevalent in real life new reporting:

1. (Feb 26, 2003) With 32 points, Kobe Bryant saw his 40+ scoring streak end at **nine** games, tied with Michael Jordan for **fourth** place on the all-time list[1].
2. (April 14, 2014) Stephen Curry has made 602 3-pointer attempts from beyond the arc,... are the **10th** most in NBA history in a season (**82 games**)[2].
3. (May 28, 2015) Stocks gained for the **seventh consecutive day** on Wednesday as the benchmark moved close to the 5,000 mark for **the first** time in seven years[3].
4. (Jun 9, 2014) Delhi has been witnessing a spell of hot weather over the **past month**, with temperature hovering around 45 degrees Celsius, ... **highest** ever since 1952[4].

[1] http://www.nba.com/features/kobe_40plus_030221.html
[2] http://www.cbssports.com/nba/eye-on-basketball/24525914/stephen-curry-makes-history-with-consecutive-seasons-of-250-3s
[3] http://www.zacks.com/stock/news/176469/china-stock-roundup-ctrip-buys-elong-stake-trina-sol
[4] http://www.dnaindia.com/delhi/report-delhi-records-highest-temperature-in-62-years-1994332

5. (Jul 22, 2011) Pelican Point recorded a maximum rainfall of 0.32 inches for **12 months**, making it the **9th driest** places on earth[5].

In the above news themes, there are a subject (e.g., Kobe Bryant, Stocks, Delhi), an event window (e.g., nine straight games, seventh consecutive days, past month), an aggregate function on an attribute (e.g., minimum points, count of gains, average of degrees), a rank (e.g., fourth, first time, highest), and a historical dataset (e.g., all time list, seven years, since 1952). These news theme indicators are summarized in Table 1.1.

E.g.	**Aggregate function**	**Event window**	**Rank**
1	`min`(points)	9 straight games	4
2	`sum`(shot attempts)	82 games	10
3	`count`(gains)	7 consecutive days	1
4	`avg`(degree)	past months (30 days)	1
5	`max`(raindrops)	12 months	9

Table 1.1: News theme summary

We model these news themes using nested neighborhood analytics. Let $e_s(t)$ denote the event of subject s at time t. Then the news themes are generated using neighborhood analytics in a two-step manner:

(1) a *distance neighborhood* $\mathcal{N}_1(o_i, w) = \{o_j | o_i.t - o_j.t \leq w\}$ groups a consecutive w events for each event. Let $\overline{v}$ be the aggregate value associated with $\mathcal{N}_1$, then the output of this step is a set of *event windows* of the form $n = \langle o_i, w, t, \overline{v}\rangle$.

(2) a *comparison neighborhood* $\mathcal{N}_2(n_i) = \{n_j | n_j.w = n_i.w \wedge n_i.\overline{v} \geq n_j.\overline{v}\}$ ranks a subject's event window among all other event windows with the same window size. The result of the this step is a tuple $\langle o_i, w, t, r\rangle$, where r is the *rank*.

The resulting news themes from $\mathcal{N}_2 \cdot \mathcal{N}_1$ is our proposed *rank-aware news themes*. Rooted from the *rank-aware news themes*, we further address the problem of controlling the result size. We propose a novel concept named *Sketch* to avoid outputting

[5]`http://www.livescience.com/30627-10-driest-places-on-earth.html`

near-duplicate themes. A sketch contains k most representative rank-aware news themes under a scoring function that considers both strikingness and diversity. Our objective is to discover sketches for each subject in the domain.

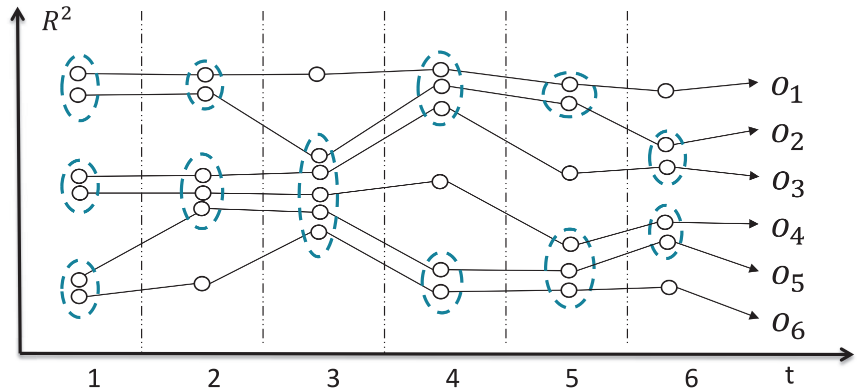
We study the problem in both offline and online scenarios, and propose various window-level pruning techniques to find striking candidate themes. Among those candidates, we then develop approximation methods, with theoretical bounds, to discover the k most representative themes. We conduct experiments on four real datasets, and the results demonstrate the efficiency and effectiveness of our proposed algorithms: the running time achieves up to 500 times speedup as compared to baseline and the quality of the detected news themes is endorsed by the anonymous users from Amazon Mechanical Turk [6].

1.3.3 Mining Co-Movement Patterns in Trajectory Databases

The third piece of the thesis addresses the neighborhood query on trajectories. Discovering co-movement patterns from large-scale trajectory databases is an important mining task and has a wide spectrum of applications. Existing studies have identified several types of interesting co-movement patterns called *co-movement* pattern. A co-movement pattern refers to a group of moving objects traveling together for a certain period of time and the group of objects is normally determined by their spatial proximity. A pattern is prominent if the group size exceeds M and the length of duration exceeds K. Inspired by the basic definition and driven by different mining applications, there are a bunch of variants of co-movement patterns that have been developed with more advanced constraints.

Figure 1.3 is an example to demonstrate the concepts of various co-movement patterns. The trajectory database consists of six moving objects and the temporal dimension is discretized into six snapshots. In each snapshot, we treat the clustering

[6] `https://requester.mturk.com`



Flock (M=2, K=3)	$\langle o_3, o_4$: 1,2,3$\rangle$ $\langle o_5, o_6$: 3,4,5$\rangle$
Convoy (M=2, K=3)	$\langle o_3, o_4$: 1,2,3$\rangle$ $\langle o_5, o_6$: 3,4,5$\rangle$
Group (M=2, L=2)	$\langle o_1, o_2$: 1,2,4,5$\rangle$ $\langle o_3, o_4, o_5$: 2,3$\rangle$ $\langle o_5, o_6$: 3,4,5$\rangle$
Platoon (M=2,K=3,L=2)	$\langle o_1, o_2$: 1,2,4,5$\rangle$ $\langle o_3, o_4$: 1,2,3$\rangle$ $\langle o_4, o_5$: 2,3,5,6$\rangle$ $\langle o_5, o_6$: 3,4,5$\rangle$
Swarm (M=2,K=3)	$\langle o_1, o_2$: 1,2,4,5$\rangle$ $\langle o_3, o_4$: 1,2,3$\rangle$ $\langle o_4, o_5$: 2,3,5,6$\rangle$ $\langle o_5, o_6$: 1,3,4,5$\rangle$ $\langle o_2, o_3$: 3,4,6$\rangle$

Figure 1.3: Trajectories and co-movement patterns. The example consists of six trajectories across six snapshots. Objects in spatial clusters are enclosed by dotted circles. M is the minimum cluster cardinality; K denotes the minimum number of snapshots for the occurrence of a spatial cluster; and L denotes the minimum length for local consecutiveness.

method as a black-box and assume that they generate the same clusters. Objects in proximity are grouped in the dotted circles. As aforementioned, there are three parameters to determine the co-movement patterns and the default settings in this example are $M = 2$, $K = 3$ and $L = 2$. Both the *flock* and the *convoy* require the spatial clusters to last for at least K consecutive timestamps. Hence,$\langle o_3, o_4 : 1, 2, 3\rangle$ and $\langle o_5, o_6 : 3, 4, 5\rangle$ remains the only two candidates matching the patterns. The *swarm* relaxes the pattern matching by discarding the temporal consecutiveness constraint. Thus, it generates many more candidates than the *flock* and the *convoy*. The *group* and the *platoon* add another constraint on local consecutiveness to retain meaningful patterns. For instance, $\langle o_1, o_2 : 1, 2, 4, 5\rangle$ is a pattern matching local consecutiveness because timestamps $(1, 2)$ and $(4, 5)$ are two segments with length no smaller than $L = 2$. The difference between the *group* and the *platoon* is that the *platoon* has an additional parameter K to specify the minimum number of snapshots for the spatial clusters. This explains why $\langle o_3, o_4, o_5 : 2, 3\rangle$ is a *group* pattern but not a *platoon* pattern.

We notice that these patterns can be unified using *neighborhood* query. Let $C_t(o_i)$ be the cluster which o_i belongs to at snapshot t, then a co-movement pattern can be viewed as a *comparison neighborhood* $\mathcal{N}(o_i) = \{o_j, T | \exists t \in T, C_t(o_i) = C_t(o_j)\}$. For each generated neighborhood, the analytic function is the pattern discovery based on temporal constraints. Therefore, we propose a *General Co-Movement Pattern* (GCMP) based on such neighborhood to capture all existing co-movement patterns. The GCMP can be reduced to any of the existing co-movement patterns by adapting with different analytic functions.

Technical wise, we study efficiently processing GCMP in a MapReduce platform to gain scalability for large-scale trajectories. We propose two parallel frameworks: (1) TRPM, which partitions trajectories by replicating snapshots in the temporal domain. Within each partitions, a line-sweep method is developed to find all patterns. (2) SPARE, which partitions trajectories based on object's neighborhood. Within each partitions, a variant of Apriori enumerator is applied to generate all patterns. We then show the efficiency of both our methods in the Apache Spark platform with three real trajectory datasets upto 170 million points. The results show that SPARE achieves upto 14 times efficiency as compared to TRPM, and 112 times speedup as compared to the state-of-the-art centralized schemes.

1.4 Thesis Organization

The remaining part of the thesis are organized as follows: in Chapter 2, we summarize related literature on neighborhood related analytics in different data domains. In Chapter 3, we present the window function on graph data. In Chapter 4, we present a news discovery application on time series data. In Chapter 5, we present a pattern mining framework on trajectory data. Chapter 6 summarizes this thesis and highlights the future directions.