February 2023

Graph Analysis on Social Networks

Shen Lu
University of South Florida

Scholar Commons Citation
Lu, Shen, "Graph Analysis on Social Networks" (2023). *USF Tampa Graduate Theses and Dissertations.*
https://digitalcommons.usf.edu/etd/9897

Graph Analysis on Social Networks

by

Shen Lu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Les A. Piegl, Ph.D.
Kandethody Ramachandran, Ph.D.
Richard Segall, Ph.D.
Robert Karam, Ph.D.
Tempestt Neal, Ph.D.

Date of Approval:
March 2, 2023

Keywords: Graph Convolution Network, Entity Detection,
Deep Learning, Attention Mechanism

Dedication

It was my pleasure to start my Ph.D. program at University of South Florida. During my doctoral education, my professional skills grew in coding, machine learning, deep learning, social network, high performance computing, and hypothesis proving. I am grateful for having so many people to help me, to teach me, to support me and to encourage me on my career development from a software engineer to a data analyst and finally to a data scientist.

First, I want to thank my supervisor, Dr. Les Piegl, for being professional, giving me guidance, and dedicating time for me, when I needed help, research directions and feedback. Graph Analysis is an interdisciplinary field which blends foundational math and physics together with computer science, human perception, and aesthetics to explore the most compelling scientific discoveries of our time. The techniques present potentials in machine learning, deep learning, social networks, and other popular applications. Facing countless theories and open issues, it is challenging to narrow down the scope of the research and to define solutions for it. It is easy for me to waste time in unlimited reading, unlimited theory exploring, unlimited choices in research directions, and even more. Dr. Les Piegl is an outstanding professor with a wide range of research experiences in different fields, such as computer-aided design and manufacturing, engineering and applied computing, software engineering and software system design, computer graphics, cyber-learning and engineering in medicine. With the help from my supervisor, I could learn to think in a bigger and higher level and to find the most important directions in problem solving. I am also grateful for his patience which gave me the chance to work full-time while working on my program.

Second, I want to thank my supervisor and friend, Dr. Richard Segall for discussing research with me, editing research papers for me, and reminding me to avoid some common mistakes in research. Dr. Segall is passionate in research. It is for many years that he has been focusing on research in data mining, text mining, web mining, big data, bioinformatics, supercomputing applications and mathematical modeling. I am fortunate enough to have a chance to exchange ideas, cooperate projects, coauthor papers and funding proposals with. His personality and work style have strongly influenced me, that helped me to insist in research and to scale career ladder quicker. With his support as a friend, I can be conscientious, emotionally stable, open to different opinions, and be agreeable to critics. These characteristics carry more weights in career success. When I enlarge my perspectives and see gains in losses, the upside of a downside situation, how far I have come in addition to how far I must go and beginnings in endings, my positive attitude comes with the potential for career success at the highest rung. Also, he was the first person who recognized my talent in research.

I also want to thank my committee members, Dr. Kandethody Ramachandran, Dr. John Licato, Dr. Robert Karam, and Dr. Tempestt Neal, for helping prepare all the paperwork, giving me all the feedback, and being supportive. Dr Kandethody Ramachandran's research covers software reliability, game theory to deregulated electricity markets, Microarray data analysis, mathematical finance, control of queues in heavy traffic, stochastic delay differential equations and controls, stochastic differential games, information theory, singularly perturbed stochastic systems, wavelet analysis for statistical signal processing. His opinions in data sampling and statistical analysis deepened my research and inspired better ideas and more thorough proofs. Dr. John Licato is an artificial intelligence researcher primarily in human-level and logical reasoning that includes computational modeling of cognitive reasoning, natural language processing, cognitive science and robotics, computational cognitive architecture, automated theorem provers, artificially intelligent reasoners, analogical deductive argumentative and hypothetic-deductive reasoning, artificial reasoning with both formal and informal logics. His feedback in natural language processing and data modeling led to the improvement of the model performance. Dr. Robert Karam is interested in hardware security, reconfigurable computing, bio-implantable devices. His feedback in research methodology gave rise to new research directions and new research ideas. Dr. Tempestt Neal focuses on the intersection of identity intelligence, cyber behavior analytics, and mobile sensing technologies. Her primary interests are behavioral biometrics, mobile biometrics, and applied machine learning, with additional interests in authorship attribution, computer vision, and soft biometric classification. With the experiences in applied machine learning, she provided valuable feedback on my research.

Finally, I want to thank my parents and my sisters for their trust, encouragement, and support, which mean everything to me.

Acknowledgments

I need to acknowledge the funding as a teaching assistantship from Department of Computer Science and Engineering during my first and second years at USF. Also, I need to acknowledge the computing facilities that was made available to me during my program at USF.

Table of Contents

List of Tables

List of Figures

Abstract

With the development of transportation network, social network, and communication network, there are many applications in streaming data. For example, traffic congestion happens between the origin and destination of daily trips. Traffic analysis can help plan the trips so that traffic congestion can be avoided. Social network and communication network represent the behaviors of the entire population. People build connections based on their hobbies, daily activities, photos, videos, simple messages, and even anonymous web surfing. All of these can be turned into commercial use, such as product marketing, business network building, and technology trending.

Data science is about how to model data for data issues, domain specific patterns, etc. If the sample set is big enough and the data is relevant, it is possible to engineer this process and to generate results. Once the data model is built, we can fit the model with the data and run proper algorithms to get answers. However, the challenges can be from data store, sample quality to information extraction.

Especially for graph analysis, it needs to deal with not only the valuation of the vertices but also the connections between vertices, and how many connections each vertex can have. According to empirical experiments, the distributions of vertices, edges, and derived measurements, such as closeness, betweenness, and clustering coefficient have different distributions. When we work on data modeling strategies, both graph properties and topology types need to be under.

In this dissertation, we discuss how to efficiently perform information extraction from graphs. Graphs can be considered as the structural representation of the social networks regarding the natural properties of information, such as recency, relevance, valuation, and validation. We focus on data loading, graph sampling, and application development.

Chapter 1: Introduction

With the development of network technology and social media, we can connect to each other easily. Internet physically connects people and business in different geographic locations. People can trade with each other through the Internet. Geographic distance is not an obstacle anymore. On top of the physical connections, the interconnections among people and things are developed as well. The network of discussions, commentary, and social media together build a virtual community, which play a significant role in our lives. All these have a common pattern in infrastructure – network, such as social network, information network, and economic network.

We can always see connections among people, organizations, and almost any entities. Connected entities can initiate activities through face-to-face meetings, transportation, telephone, mails, Internet, social media, and many other channels. Once the information exchange happened, the social flow was built upon a network and will go on and on, evolve, and thrive.

We consider that individual entities can carry information and the information can be delivered from one entity to another. The information that plays the role of the driving force in the social network can be related to anything, such as an eye contact, a handshake, a travel plan, a conversation, a letter, a web link, a friend request, a flu epidemic, a finical transaction, and even more. What can be sent through the channel depends on the information carrier and sometimes cannot be directly recognized, which make people think what really triggered the change.

In terms of data science, it is important to recognize the proper information carrier and the interesting information which can flow through the network and can be used to build the data model, to represent the network structure and to make predictions for future use. For example, people can be considered as information carrier because people initiate activities, control both people and resources, plan activities and accomplish tasks. However, the signals people send between each other can be an attitude, an opinion, a mimicry of each other's behavior and even more which can transfer information from one to another, influence people and trigger reactions so that we call them social signals. Symbolization of social signals is the key to profile a population of specific behaviors. The more detailed patterns we can catch, the more deeply we can understand a specific community, and the more accurately we can model the real-world activities with machine learning techniques.

In terms of data structure, the information carrier, and information channel can be modeled as a graph, that has a network pattern. The vertices are the social entities, the edges are the connections between entities, and the information as social signals is sent through the edges between entities. A proper graph model can clearly explain what is happening in the modern society and can be used to predict what will happen in the future. Moreover, the complexity of the network as a whole to react to the central driving force is more important than that of individual entities, such as entities and channels.

The efficiency of graph analysis can be affected the complexities of graph storage, graph sampling, graph algorithm, and graph applications. For graph storage, the challenges can be in space capacity, I/O, duration, and memory access. For graph sampling, the challenges can be how to maintain graph properties and topology types. For graph algorithms, the challenges can be graph search, dynamic update, parallel computation, partitioning, and compression. For graph application, the challenges can be community profiling, small world phenomenon, and subgraph finding.

The theories behind graph analysis include combinatorics for space management, graph theory for decomposition and reconstruction, conditional probability theorem, Markov chain theories, and attention mechanisms that helps achieve Markov-like updates in deep-learning architecture.

In this dissertation, we apply these theories to solve open issues in graph space management, sampling, entity detection, and classification. In Chapter 4, we discuss both space and duration management focusing on how to apply bin-packing theories to estimate both space capacity and time duration. In this chapter, we propose and prove eight theorems related to NP-complete for both space management and integer packing, upper/lower bounds of space complexities, upper/lower bounds for sorted input, and upper/lower bounds for algorithms. In chapter 5, we discuss graph sampling focusing on how to use Kronecker double cover and curvatures to solve sampling issues in graphs. We propose and prove the complexities of Kronecker double cover for graph sampling. In chapter 6, we discuss how to combine domain specific features to Conditional Random Fields (CRF) model for entity detection on graphs. We summarize rich features in linguistic, semantic, domain specific perspectives. In chapter 7, we discuss how to combine domain knowledge with deep learning architecture. Based on attention mechanism, we propose to use domain knowledge to influence Markov-like updates so that the domain specific entities can be promoted.

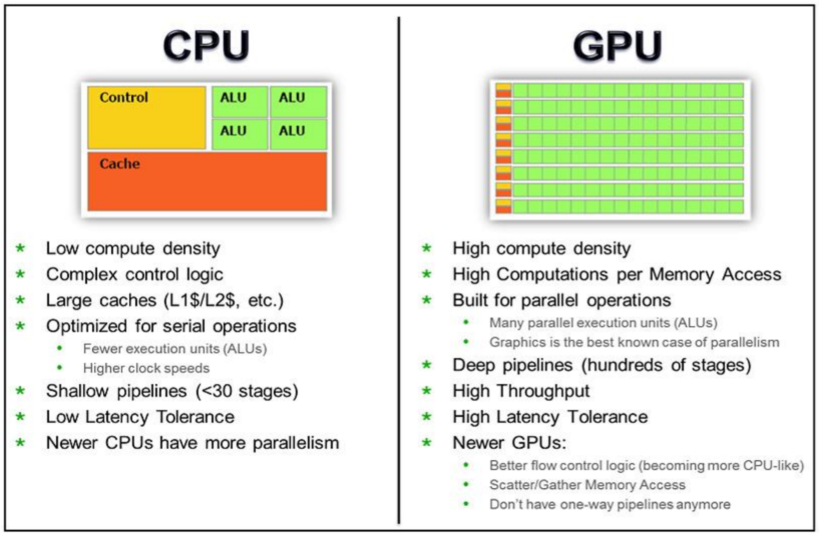
Chapter 2: Background

2.1 GPU and Parallel Algorithms

Parallelism modifies the computing processing and boosts computing performance through improving the infrastructure from single thread to multiple threads. Although the basic process of a computing task stays the same, but, data partition, data synchronization and the integration of the computing results must be under consideration. GPU provides a massive amount of parallelism with the potential to outperform CPU. Especially, in graph processing, the individual vertex in graph matrix can be mapped to each one of the processing units in GPU grid, which provides a full utilization of the infrastructure, simplifies the data model and makes it possible to improve the computing performance in an order of magnitude. There are several classical parallel algorithms, such as prefix sum, sorting, and filter. These algorithms efficiently model the parallelism in high performance computing.

Figure 2.1 below shows the highlights of comparisons between Central Processing Unit (CPU) versus Graphics Processing Unit (GPU). Note that a CPU has a low compute density, and a GPU has a high compute density. A CPU has a low latency tolerance, and a GPU has a high latency tolerance, where latency refers to a measure of the time delay required for information to travel across a network. A CPU has shallow pipelines compared to a GPU with high throughput.

GPU has a large amount of device and on-chip memory as well which avoid the memory stall issue often happening on CPU. GPU algorithms are a counterpart of CPU algorithms. If we can hide memory latency and solve read/write conflicts on GPU, we can convert CPU algorithm to GPU algorithm with a few changes. Memory latency problem can be solved through pipelining.



- Low compute density
- Complex control logic
- Large caches (L1$/L2$, etc.)
- Optimized for serial operations
  - Fewer execution units (ALUs)
  - Higher clock speeds
- Shallow pipelines (<30 stages)
- Low Latency Tolerance
- Newer CPUs have more parallelism

- High compute density
- High Computations per Memory Access
- Built for parallel operations
  - Many parallel execution units (ALUs)
  - Graphics is the best known case of parallelism
- Deep pipelines (hundreds of stages)
- High Throughput
- High Latency Tolerance
- Newer GPUs:
  - Better flow control logic (becoming more CPU-like)
  - Scatter/Gather Memory Access
  - Don't have one-way pipelines anymore

Figure 2.1. Comparison of CPU vs. GPU ([278] Public Domain)

We use Nested-loop join as an example to show how to convert a loop into a parallel algorithm through an additional data structure – a histogram [106]. Nested-loop join is central of many database operations which can be done in two steps: sort and merge. As shown in Figure 2.2, during data sorting, data is hashed into histogram and then distribute data from histogram to output memory. Each thread group maintains a histogram and the histogram stores the counts of each value hashed into the histogram. In histogram, data are sorted because it is organized through radix hash. Since histogram also stores the group index of the data, we can scatter data into new memory locations by reading the new memory location of the data in the histogram and then data is sorted. The strength of this idea is that, by adding a histogram to each through group, data sorting can be done in parallel. The weakness of this algorithm is more space cost.

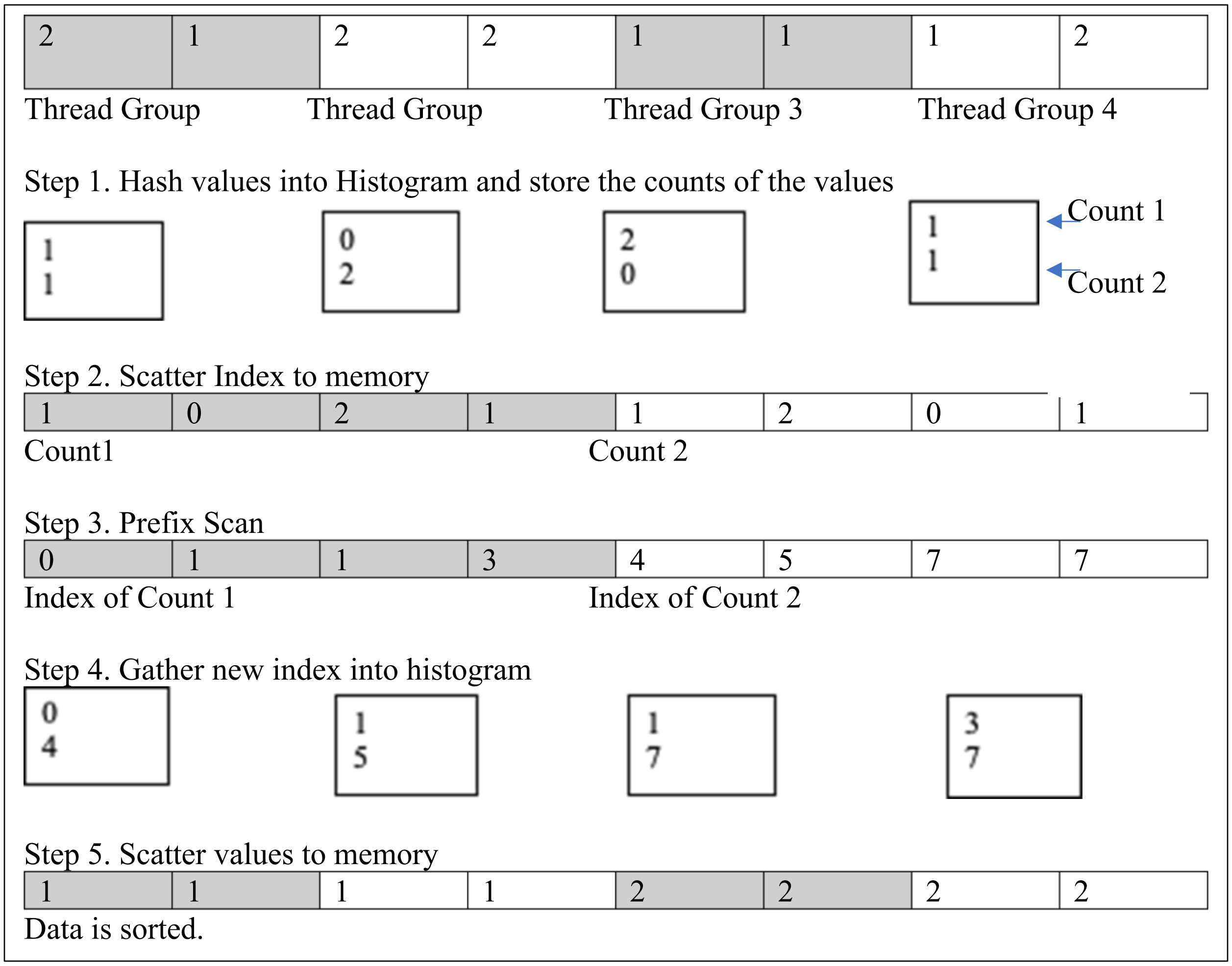


Figure 2.2. Scatter and Gather Process in Parallel Sorting

Graph processing can also be accelerated through parallel mechanism. We use graph splitting as an example to show how graph processing can be sped up through parallelism. Graph analytics naturally allows more complicated reasoning by exploring many-to-many relationships so that the workload we need to balance is the degree of each node. After graph splitting, the degree of each node will be less than a threshold, which means the workload on each node is reduced and balanced as well. During parallel processing, processor will assign threads to each one of the nodes and conduct computation simultaneously.

Degree distribution of the graph follows power law that a small subset of nodes own the majority of neighbors. The high irregularity of the degree distribution makes partitions imbalanced and slow down parallel performance. Uniform-Degree-Tree (UDT) Transformation was proposed and used in Tigr [210]. Tigr changes the topology of the graph without partitioning. It can transform irregular graph into regular ones with provable correctness, efficiency and effectiveness. Graph regulation was performed on an extra virtual layer instead of the physical layer, which was handled by GPU parallel computing and returned results to the host machine. By combining virtual layer and physical layer together, the entire graph was regulated.

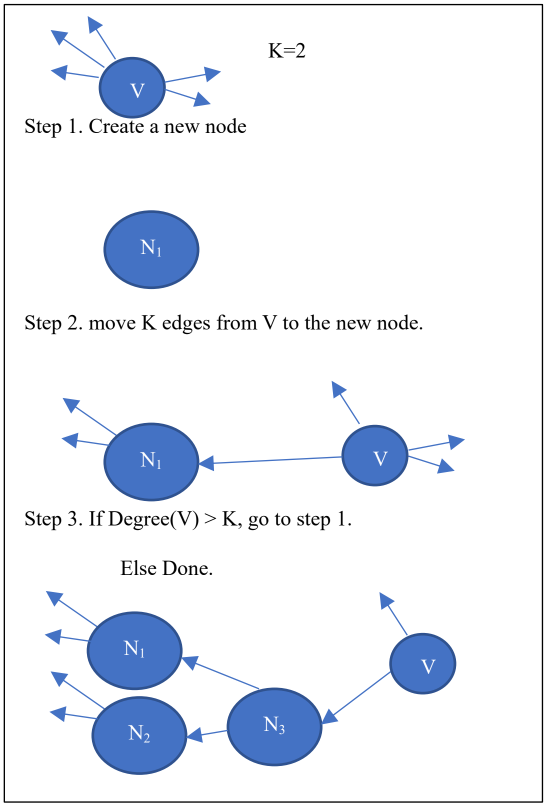

K=2
V
Step 1. Create a new node
N1
Step 2. move K edges from V to the new node.
N1
V
Step 3. If Degree(V) > K, go to step 1.
Else Done.
N1
V
N2
N3

Figure 2.3. UDT Transformation Process.

Figure 2.3 shows the UDT transformation process. Given a vertex v with degree d greater than the threshold K, we apply UDT iteratively until the degree of vertex v is less than the threshold

K. It ends up with we have more vertices in a graph, but the number of outgoing edges of each vertex is reduced to no more than 2. When parallel processing is applied, each thread can take care of one vertex, all of the vertices can be processed simultaneously, and the computing cost is O(E) which is equal to 2, instead of 5, in this example.

2.2 Parallel Optimization

Classical graph processing algorithms can be parallelized, such as BFS [22], SSSP [54][175], betweenness centrality [35]. Synchronous parallel model was discussed in [243]. Parallel graph processing algorithms are compared in [253]. Based on graph data structure, parallel graph processing can be done through push and pull based schemes [Betta, M. et.al. 2017], input-guided graph traversals [188][273][274]. A distributed graph programming system library were implemented in Boost Graph Library [223]. Vertex centric parallel algorithms can be done through different partitioning between high degree and low degree vertices [44] and are implemented in several graph processing frameworks, such as Pregal [169], Apache Graph [13], GraphLab [161], Power Graph [88]. Graph processing framework can be either on single PC, such as GraphChi [140], GraphQ [252][253], or on parallel machines through memory sharing, such as Ligra [222], Galois [202], Charm++ [116], STAPL [103] [105].

2.3 GPU Optimization

GPU has several bottlenecks, such as data transfer, kernel invocations and memory latencies [264]. Optimize memory access efficiency on GPU was discussed in [271]. Load balancing can be done through local balance [183][184], a queue-based task balancing in irregular graph [240], decomposition of imbalanced load [146][266], and irregular graph processing [183][184]. GPU warp optimization can be done also through virtual warps [25] and the trade-off between path divergence and accuracy of result by forcing all the warp lanes to follow the majority [213].

2.4 Parallel Join Operation

Maximal pipelined parallelism normally cannot lead to the superior performance. Segmented bushy processing strategy combines pipelined parallelism with alternate forms of parallelism to achieve an overall effective processing strategy in complex multi-join queries on shared-nothing parallel system, in which the pipelined segment can be attached to the query tree, not only at the first join operation but also in the middle of the join operation [156]. Partition tuning for data skews through three new parallel hash join algorithms [117]: tuple interleaving parallel hash join (TIJ), adaptive load balancing parallel hash (ABJ), extended adaptive load balancing parallel hash join (ABJ+). The three algorithms use best fit decreasing strategy to tune the load of each partition. TIJ and ABJ+ are good skew avoidance technique, which are robust against skewed data, and ABJ is also a good skew resolution as well. Machine-specific communication primitive to develop parallel join algorithms on SIMD connection machines [15]. Database operations, such as Multi-threaded hash join in shared memory system [162] are tested to explore the implication that the larger on chip multithreading can have for parallelism in database operations. The conclusion is that the most important feature of using multi-threading is the cost of high latency memory operations are hidden through multithreading [50]. Database join operation algorithm with GPU can be done through scatter-gather mechanism and a hash table in the form of histogram [106].

2.5 Parallel Query Operations

Spatial database operations are done in two steps: Filter step and refinement step. Refinement step requires heavy computation which can be improved through GPU accelerated parallel mechanisms, especially in rendering and search capabilities for spatial selection and join operations [233], bitonic sort [91], fast quantile frequency estimation [90]. Multi-pass scheme can be used to improve the scatter and gather operations on the GPU [106]. Similarity join algorithm - LSS [155], uses simple GPU operations, such as sorting and search, to implement parallel similarity join in two steps: one is to create space filling curves on one of its input data sets through sorting, the other is to process each point of the other data set in parallel to search an interval of one of the space filling curves which contain all the pairs the point participates.

2.6 Graph Processing with GPU

Graph Models are useful in identify influencers in social networks [179], spotting frauds in band transactions [211], optimizing supply chain distribution [249], developing recommendations [61], and effective medical treatments [37].

Graph processing can be optimized through the utilization of GPU based on coalesced accesses [132], the maximization of warp [113], and compiler-level optimization [192]. The connectedness of the graph makes it easy to utilize GPU in processing [94][226]. Graph processing algorithms can also be accelerated through GPU, such as BFS [174][164], SSSP [57][172], betweenness centrality [123][172][212], vertex degree reduction [210] and graph reduction [217][100]. Muti-GPU graph processing have several implementations, such as TOTEM [81], Medusa [275], METIS [131], CPU-GPU methods [82][112]. GPU graph processing framework is also implemented in [102].

2.7 Graph Store, Task Scheduling and Parallel Planning

Data stream is a sequence of unbounded data which need to be captured, stored and processed [182] [171]. Normally, we capture a snapshot of the stream instead of the entire stream. If the size of the time window for each snapshot is bigger than the time cost of processing data plus the time cost of read and write, it is feasible to continuously conduct analysis through the entire data stream. However, a memory and a parallel plan must be made first.

Elements

1	0	1	1	0	1	0	1	1	…

Arrival Time

21	20	19	18	17	16	15	14	13	…

Time Stamp

9	8	7	6	5	4	3	2	1	…

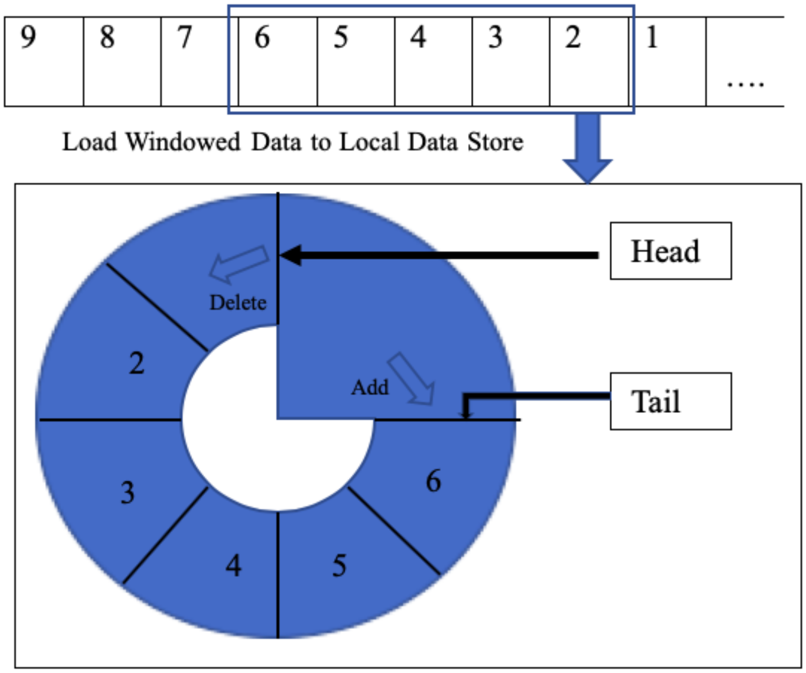


Figure 2.4. Circular Queue Data Structure for Sliding Window

Streaming data window is a snapshot of the data stream. The data window moves along the data stream one slice every time interval. Streaming data are a sequence of tuples with time stamps, partially ordered by time. The windowed data are stored locally for operational purpose. If we store the data in the order of time stamps, data updates can be implemented within a small range of data store: delete the outdated data from the head of the queue and add new data to the tail of the queue. To handle this process repeatedly, we define the data structure of sliding window as a circular queue, in which data can be updated always within the area between the head and tail of the queue. If the circular queue has enough storage, the process can keep running along with the movement of the data stream. The challenge in streaming window is the tradeoff between the space capacity and the accuracy of the results. The optimal solution is that we can decrease the space cost as much as we can, but also, increase the accuracy of the results as much as possible.

Bin-packing problem deals with the limited space capacity and unlimited items. Bin-packing solution can be used to plan the space and the other resources in the first place because it provides a way to utilize the space and to load as many items as possible. Given a list I of items, and a list J of bins, all the items have the same size, and all of the bins have the same size. Without splitting any items, we need to load as many items in bins as possible and to use as few bins as possible. This problem can be modelled as the following

Minimize Σy_j, (2.1)

Subject to $\Sigma a_i x_{ij} \leq C y_j,\ j \in J$, (2.2)

$\Sigma x_{ij} = 1,\ i \in I$, (2.3)

$x_{ij},\ y_j \in \{0, 1\},\ i \in I,\ j \in J$. (2.4)

in which, the set of boxes $Y = \{y_j \mid y_j \in \{0, 1\},\ j = 1, \ldots, m\}$, the set of items $X = \{x_i \mid x_i \in \{0, 1\},\ i = 1, \ldots, n\}$, a_i is the size of the item i. The constraint (2.1) can ensure that the number of boxes needs to be minimized. The constraint (2.2) can ensure that the total size of items in each box cannot exceed the capacity of the box. The constraint (2.3) can ensure that one item can only be in the box. We demonstrate the model in Figure 2.5, as shown below.

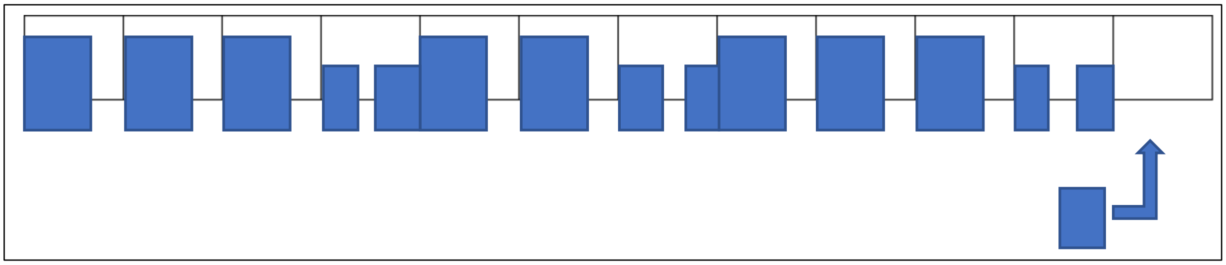


Figure 2.5. Bin-packing Items with Different Sizes.

2.8 Performance Ratio

Bin-packing problem is NP-complete [124]. When a problem cannot be solved with polynomial algorithms, we use a near optimal solution to solve it. The near-optimal solution is called approximation algorithm. The performance of approximation algorithms cannot be evaluated with polynomial time complexity because its time complexity is not polynomial. We use approximation ratio as the performance ratio of the approximation algorithm to do evaluation.

$$\max\left(\frac{C}{C^*}+\frac{C^*}{C}\right) \leq \rho(n) \qquad (2.5)$$

in which C is the cost of the approximate solution, C* is the cost of optimal solution, $\rho(n)$ is the performance ratio of an approximate solution, indicating that the algorithm has $\rho(n)$ - approximation. $\rho(n)$ must be greater than or equal to 1. A large approximation ratio means that the approximation solution is much worse than the optimal solution. When it is hard to get the worst case performance ratio, we use lower bounds and upper bounds, instead of performance ratio, to measure the performance of the algorithms.

For example, we have N bins with size B, and 2NB items: NB size-B/2 items and NB size-1 items. If we use Next Fit algorithm to load items into bins, and items come in the order of B/2, 1, B/2, 1, …, the approximate solution needs to use NB bins and the optimal solution needs to use (NB/2 + N) bins, as shown in Figure 2.6. The performance ratio is

$$(NB/(NB/2+N)) \leq \rho(n) \quad (2.6)$$

$$(2/(1+1/B)) \leq \rho(n) \quad (2.7)$$

$$2 \leq \rho(n) \quad (2.8)$$

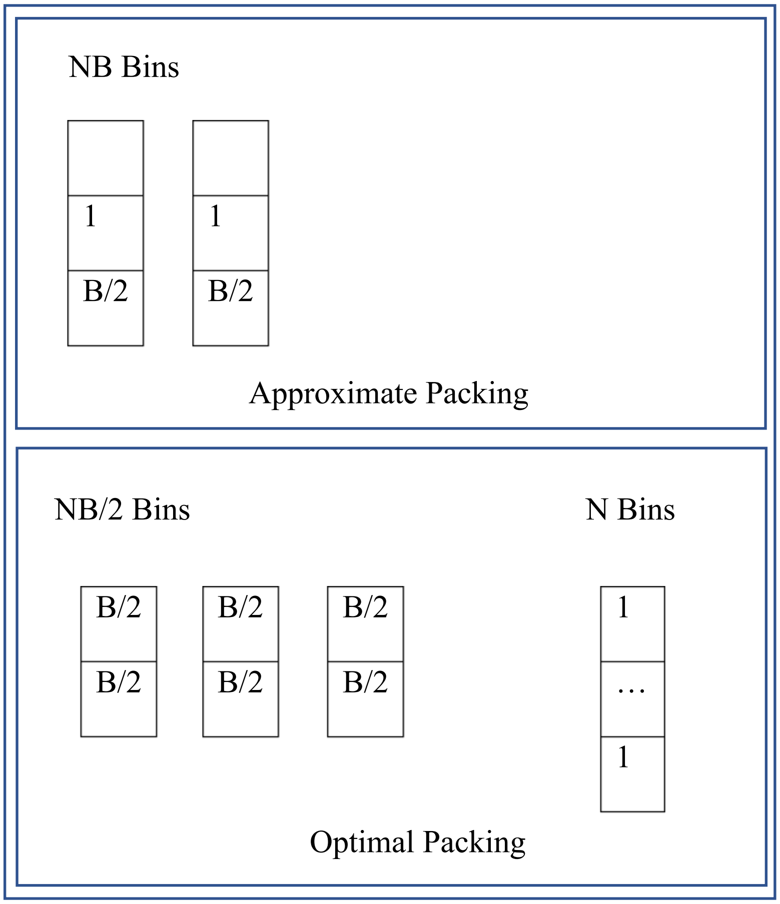


Figure 2.6. Next Fit Bin-packing Algorithm with Performance Ratio $\rho = 2$

Description of the data distribution is central in data management. Cumulative distribution is characterized by the quantile of the data, especially in an online stream fashion [163].

Computational complexity of streaming data requires the estimation of upper bound and lower bounds. This gives rise to the strategy of bin packing, which can be used to manage both the space and the schedule. With a bin packing strategy, we can manage resources for a system with or without parallelism: the upper bounds and lower bounds of both memory and time costs of the jobs can be estimated, data loading and task scheduling follow a predefined atomic process, and load balancing and load efficiency are also under consideration. Given a list L of n numbers, for every positive e, there exists an O(n) time algorithm S such that, S(L) is the number of bins used by L, L* is the minimum number of bins needed to pack L, then $S(L)/L^* < 1+\varepsilon$ [72].

2.9 Bin-packing Algorithms

2.9.1 Classic Bin-packing Algorithms

To solve bin-packing problem, several algorithms are generally used, such as Next Fit (NF), First Fit (FF), Best Fit (BF) [124]. The difference among these algorithms is how to arrange the next item. Next Fit algorithm keeps the last bin open for packing at any time. If the next item can fit into the last bin, the problem is solved. Otherwise, a new bin will be added for the new item to use and the item will be put into the new bin. The time complexity of Next Fit algorithm is O(n), because, given n items, we can directly put each item into the last bin, no other options. When the sizes of the items are random, there is a big waste of space. First Fit algorithm leaves all partial filled bins open for packing. When a new item comes, the algorithm searches from the beginning to the end to find the first box which has enough space to load the new item. So, the time complexity of the algorithm is at least O(nlogn), in case bins are built on a tree structure. First Fit algorithm sacrificed time cost for a better space complexity. Best Fit algorithm is similar to First Fit algorithm, in which the new item will be loaded into an enough but least space of the bin. The space complexity of best fit algorithm is better than First Fit algorithm. The time cost of Best Fit algorithm is also at least O(nlogn), in case, bins are built on a tree structure.

Another modification of First Fit algorithm is First Fit Decreasing (FFD), in which the items are first sorted in decreasing order, and then loaded into different bins with First Fit algorithm. Similar to this algorithm, Best Fit Decreasing also needs to sort items in decreasing order before running Best Fit algorithm to load items into bins. The time complexity and space complexity of First Fit Decreasing and Best Fit Decreasing are almost the same. The performance of the two algorithms are also as good as First Fit algorithm and Best Fit algorithms. In some scenarios, the First Fit Decreasing and Best Fit Decreasing (BFD) are better than First Fit and Best Fit.

Different from NF, FF, FFD, BF, and BFD, Sum of Squares (SS) [55] uses a utility function to decide how to arrange the next item. Two terms are introduced in this algorithm: One is the level (l) of the bin, the other one is the utility function, N(l). The level of the bin indicates how much the bin has been filled up. For example, if the total size of the items in a bin is 5, the level of the bin l is 5. The utility function N(l) indicates how many bins have level l. Given a new item, we put the item into every bin which has enough space for hold it and then calculate the results of $\sum_{i=1\ldots B} N^2(l_i)$. Whichever solution can generate the largest result will be final solution and the arrangement of the solution is the final arrangement of the new item. The time complexity of Sum of Squares algorithm is O(nB). The performance of Sum of Squares is not much better than Next Fit, First Fit, Best Fit, First Fit Decrease, Best Fit Decrease, but, in each iteration, the total sizes of the items in each bin are distributed evenly.

2.9.2 Off-line Bin-packing

There are three classical bin packing problems: multiprocessor scheduling, bin packing, and the knapsack problem. Surveys [48] [51] reviewed approximation algorithms for bin packing and the results, which includes online bin packing, offline bin packing, variations on size and item packed, and packing with additional constraints. One dimensional bin packing solution was discussed in [130].

Vector scheduling and vector bin packing are related to integer programming, which is to pack a maximum number of vectors in a single bin of unit height [42]. Low-order polynomial time algorithms for near-optimal solutions to the problem of bin packing, with linear time approximation to these packing rules have better worst case behavior than FIRST FIT under large variety of restrictions on the input [112]. The absolute approximation ratio for first fit bin packing is exactly 1.7, and also matching lower bounds for a majority of values of OPT for any values of OPT [62]. In [108], bin packing problem can be solved with O(logOPT) bins.

The methodology for stock cutting, has been extended and adopted to the specific fully scale paper trim problem. In [84], a new and faster knapsack solution was provided with methodology, experiments, formulation changes for multiple cutting knives available, n balancing of multiple machine usage and m introduction of a rational objective function. Experimental study of Gilmore-Gomory cutting-stock heuristic and related LP-based approaches to bin packing was presented in [10]. By using dynamic programming to solve the unbounded knapsack problems that arise in this approach, the average running time can be $O(m^4)$ and feasible for m in excess of 1,000. In case d=2, the existence of an asymptotic polynomial time approximation scheme is P=NP [259]. 2-dimensional bin packing can have tight approximate ($1.5+\varepsilon$), instead of 2. The result also be rounded by exploiting various structural properties of optimal packings and using multi-objective multi-budget matching techniques and expanding the round and approximation framework to go beyond rounding based algorithms [18], a multiplicity scheduling, and d-dimensional bin packing algorithm with constant d number of types [86], d-dimensional vector bin packing with lower bound 1+ε-approximation and run time $(1/\varepsilon)^{O(d\log\log d)}$ [18].

2.9.3 Online Bin-packing

Efficient energy consumption can be estimated in real time through resource sharing through workload consolidation. After spreading workload to a small set of machines, all the work can be done in a short period of time and uses as few machines as possible.

Ja´nos Balogh, et al., revisited the online bin packing problem in [17] and, based on weight function, provided analysis to illustrate online algorithms for the online bin packing problem. Online placement of tasks can be done in real time but cannot arrange the tasks properly because it cannot predict which task might come next. Therefore, the consolidation of online placement has a poor performance in terms of wastage of resources, throughout, and consumption of electricity. Offline placement can optimize the solution but scarify the solving time and space. Semi-online framework is the trade-off between online and offline approaches in resource utilization and the overall run time duration.

The constraints on bin packing is in both the bin capacity and the number of bins. The solution of generalized bin packing problem to vector-capacity bin packing with or without precedence constraints was provided in [77]. Due to the online nature of the assignment, we only consider information available up to time i and the total duration in which the items can lock the resources on its host bin. Based on the basic constraints, a First Merged Fit (FMF) [12] was proposed to allow each task to be associated with either another task or a machine. Tasks can be arranged together before assigned to a machine. When the tasks are assigned to machines, the order of the tasks are already optimized to ensure the minimization of the duration for all available tasks. In each iteration, the order of the machines prioritizes the longest remaining run time, and merging happens between the best ranked bin and the next compatible bin in the list. The performance of online bin packing was improved to O(d^{1-e}) and upper bound can be B [15], the performance ratio less than 1.692 in Harmonic [145], lower bound equal to 1.54037 for one dimensional online bin packing for their asymptotic worst-case behavior [17]

2.9.4 Vector Bin-packing

Vector bin packing is associated with space management, resource scheduling and other resource sharing. Although these resources are independent of each other, the optimal solution is the number of bins can be minimized and the utilization of all the resources in each bin can be maximized. As shown in Figure 2.7, we use the space and duration optimization problem as an example to demonstrate vector bin packing problem. Given maximum duration and maximum space in each bin, we want to maximize the total duration and the total space so that the area between the total duration and the total space can be maximized.

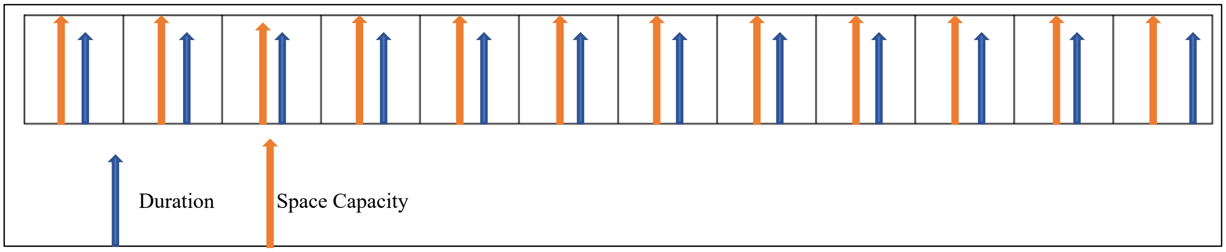


Figure 2.7. Space and Duration Optimization Bin-packing

The difference between vector bin packing and multi-dimensional bin packing is the overlap. For vector bin packing, there is no overlap in any dimension. For multi-dimensional bin packing, in a particular dimension, there is no overlap for its corresponding variable, but overlaps are allowed for other variables, as shown in Figure 2.8.

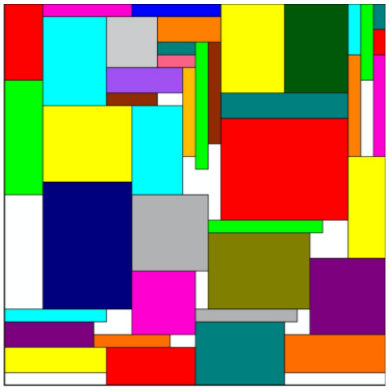


Figure 2.8. 2-Dimensional Bin-packing

Online streaming vector bin packing can be based on approximation of the number of bins. When we determine the number of bins, we can estimate based on the big items only and expand it based on the approximation ratio [53]. Round Approximation solution [18] was proposed to solve d-dimensional and d-dimensional vector bin packing both with and without rotations. If the boundaries are estimated properly, the estimate can result in a tighter integrality gap for linear programming relaxation. Multi-objective/Multi-budget matching before applying rounding and approximation framework [18] fixed a limitation of rounding and approximation algorithm, which is a solution with better than d-approximation. Vector scheduling is a natural generalization of makespan. The upper and lower bounds of vector scheduling is a double exponential dependency on d [19].

In [120], the online complexity of the vector scheduling was solved that, for identical machines, the optimal competitive ratio is O(log(d)/log(logd)) as lower bound through an online coloring game and randomized coding scheme, for unrelated machines, the optimal competitive ratio is O(log(m)+log(d)).

In [73], for non-preemptive scheduling of jobs with known processing times on m identical machines, when m goes to infinity, its competitive ratio is 1+ $\sqrt{\frac{1+ln2}{2}}$< 1.9201. For vector bin packing, the competitive ratio is $(1/B)^{d}$ [15]. For d-dimensional vector bin packing, the lower bound is 1+ε-approximation and run time $(\frac{1}{\varepsilon})^{O(d*\log(\log(d)))}$ [19]. For a Full Polynomial-Time Approximation Scheme (FPTAS) of $(\frac{1}{\varepsilon})^{O(m^{(1-\sigma)})+n^{O(1)}}$ has running time O(n)+ $(\frac{1}{\varepsilon})^{O(m)}$. If FPTAS of $(\frac{1}{\varepsilon})^{O(m^{(1-\sigma)})+n^{O(1)}}$ has running time $2^{O(\frac{1}{\varepsilon^2}*(\frac{1}{\varepsilon})^3)}$ is optimal and prove dynamic programming algorithm with running time $2^{O(n)}$ is the best possible [43].

2.10 Sampling

Bin-packing problem can be solved through sampling. In [20], a weighted uniform sampling was provided with time cost O(n^2*poly(1/ε))+g(1/ε) time and O(blogs * loglogn/($\sum a_i$ + $\frac{1}{\varepsilon}^{O(\frac{1}{\varepsilon})}$) in [24]. Moser-Tardos framework was generalized in [104] to partial resampling in graph traversals for packet routing algorithm.

The upper bound and lower bound of the bin packing solution are the significant factors we care about, instead of the number of bins and the capacity of each bin, when design the strategy. Bin-packing problems are to minimize the number of bins used and to maximize the space capacity of each bin. Although both the number of bins and the space capacity of each bin need to be optimized, the number of bins can be calculated based on the total space capacity required and the bin space capacity we choose. In terms of space packing, given the total space cost of the sample set, space capacity of each bin and the number of bins can be determined by each other. For parallel scheduling, we normally map the number of bins to the number of processors and map the total space capacity to the size of the assigned memory space. For streaming data processing, the task must be done before next window slot coming, which sets an upper bound to the total time costs.

In other words, space management and scheduling are highly restricted by both the infrastructure and the operations. In practice, although we need to estimate the total number of bins and total space capacity for each task, the priority is to design a strategy which can waste less space and balance time costs in different bins.

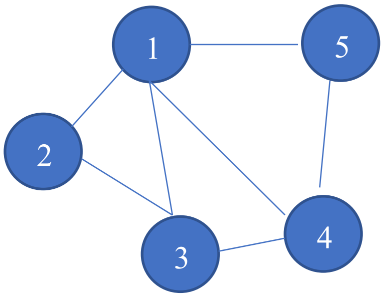
2.11 Triangle Computations and Subgraph Finding

Graphs are increasingly used to study interactions in a variety of context. Triangle is the smallest unit of a community. There is a growing need for graph analysis to profile behaviors of the population, the online community, the transportation, the business and the consumers. Among various metrics of interests, the triangle computation provides structural information about the network being studied.5.4.1

2.11.1 Triangle Counting

In network modeling, a triangle represents a community and also big communities are formed through triangle combination. Triangle computation, such as triangle counting, triangle listing, triangle finding, is central in complex network analysis. For example, it can be used to compute neighborhood density which is the number of the triangles over the number of wedges, clustering coefficient which is the average of neighborhood density, and transitivity which is the total number of triangles in a graph over the total number of wedges in a graph.

The mathematical model of a graph is an adjacent matrix so that triangle computation problems are solved through matrix multiplication. For example, given matrix A, as shown in Figure 2.9, the number of neighbors of each vertex v is equal to the value of v_{ii} in A*A, and the number of triangles of each vertex v is equal to the value of v_{ii} in A*A*A. When the number of vertices and the number of edges increase, this computation can go exponential.



(a) A Undirected Graph

A=0,1,1,1,1. A*A=4,1,2,2,1. A*A*A/2=3,3,3.5,3.5,3
1,0,1,0,0. 1,2,1,2,1. 3,1,2.5,1.5,1.5
1,1,0,1,0. 2,1,3,1,2 3.5,2.5,2,3.5,1.5
1,0,1,0,1. 2,2,1,3,1 3.5,1.5,3.5,2,2.5

(b) Triangle Counting

Neighbor(A) > 2 and Triangle(A) > 1

A*A=4,1,2,2,1 A=0,1,1,1,1. A*A*A/2=3,0,0,0,0,0
1,2,1,2,1 1,0,1,0,0. 0,0,0,0,0,0

(c) Triangle Counting for Threshold=k with Skips

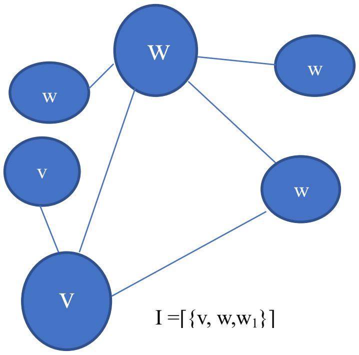
Figure 2.9. Triangle Counting in an Undirected Graph.

In theory, matrix multiplication solution can be expanded to solve subgraphs which consist of 4-vertex, 5-vertex patterns, and n-vertex patterns. However, with the increase of the number of vertices in a subgraph pattern, the cost of matrix multiplication makes it possible to compute so that matrix multiplication is not a feasible solution. This problem can be solved through parallel computing, direct pattern matching and heuristic learning.

Other than matrix multiplication that is a straightforward solution for triangle counting, the triangle counting algorithms can be based on graph traversal with O(|V|^2) time complexity, as listed in Figure 2.10. In Figure 2.10(b), we pick two vertices v, and w, in which the degree of v is less than the degree of w and also R(v) is less than R(w) as well. In Figure 2.10(a), on line 1, vertices are sorted by degrees, and the list of result is stored in array R. On line 2-4, the intersections between the neighbors of vertex v and the neighbors of vertex w are counted, and the result is stored in array C.

```
Algorithm Triangle Counting
Procedure Triangle-Counting(G(V,E))
   1. Compute an array R to store a selected list of
      neighbors for each vertex v, such that if R[v] <
      R[w], then d(v) < d(w)
   2. For w in R[v]
   3.       I= intersect(R[v], R[w])
   4.       C[v,w] = |I|
   5. Total Number of Triangles = sum(C)
```

(a) Triangle Counting Algorithm



(b) A Graph with d(v) < d(w)

Figure 2.10. Triangle Counting Algorithm

Triangle computation is useful in network analysis [29][116]. Triangle computation can be used for social network small community detection [185], thematic structure of the network [65], spam and fraud detection [23][39], link classification and recommendation [237], join three relationships in database [186], data query optimization [21].

Triangle computation can be done in memory [49][116][134][140][173][191] or in distributed system [11][88][193][194][256].

Several triangle computation algorithms were compared in [260], a simple classification scheme is designed to analyze the strengths and weakness of several existing algorithms in motifs detection. Local topology structure of the network is central in network problems. In [207], graphlet frequency distribution (GFD) was used as an analysis tool for understanding the variance of local topological structure in a network and also developed an algorithm - GRAFT - to approximate the Graphlet frequency for all graphlets that have up to five vertices. Also, the NP-Hard list-colored graph motif problem was studied in [31], with fixed-parameters (|M| and |S|) algorithms in the context of querying protein-protein interaction networks.

Vertex sampling can efficiently improve triangle computation. In [2], the proposed Graphlet counting algorithms count a few graphlets and with these counts along with the combinatorial arguments, we obtain the exact counts of others. GUISE [32] uses Markov chain Monte Carlo sampling method to construct the approximate of a large network for Graphlet frequency counting. In [122], based on 3-path sampling and a special pruning scheme to decrease the variance in estimates. In [218], a new sampling-based methods for counting the number of triangles or the number of triangles with vertices of specified degree in an undirected graph and for counting the number of each type of directed triangle in a directed graph. The number of samples depends only on the desired relative accuracy and not on the size of the graph. DOULION [237] is a new triangle counting algorithm: 1. Reweighs an edge with 1/p survive and 1-1/p delete, 2. Count each triangle as the product of the weights of the edges comprising the triangle. In [110], a new combinatorial method for counting graphlets and orbit signatures of network nodes builds a system of equations that connect counts of orbits from graphlets with up to five nodes, which allows to compute all orbits from graphlets with up to five nodes and allows to compute all orbit counts by enumerating just a single one.

Based on approximation, a randomized algorithm [238] is introduced to approximately count the number of triangles in graph G: keep each edge independently with probability p, enumerate the triangles in the sparse graph G’ and return the number of triangles found in G’ multiplied by P^{-3}. A new fast approximation algorithm for the weighted clustering coefficient [215] also gives very efficient approximation for the clustering coefficient with O(1) time complexity and for the transitivity with O(n) time complexity.

Some other ideas are also given to accelerate triangle computation, such as MapReduce, combinatorial algorithm and distributed message passing. Large graph problems can be solved through the power of the cloud if graph operations can be decomposed into MapReduce steps [52]. In [234], a new distributed triangle counting algorithm allows for a smooth tradeoff between the memory available on each individual machine and the total memory available to the algorithm and can be adapted to MapReduce setting. Edge searching of a graph was given in [47], which is useful to the various subgraph listing problems, can help create four new algorithms: triangle listing algorithm with O(a(G)m) time, quadrangles finding in O(a(G)m) time, complete subgraph listing in O(pow(la(G), l-2)*m) time, cliques listing in O(a(G)m) time. The theory behind the new distributed 3-profile counting algorithm is that sparse graphs can be used to approximate the full 3-profile counts for a given large graph [67]. Edge pivoting allows us to collect 2-hop information without maintaining an explicit 2-hop neighborhood list at each vertex. This enables the computation of all the local 3-profiles in parallel with minimal communication. Meanwhile, a novel distributed algorithm was given in [68], for counting all four-node induced subgraph in a big graph. This algorithm is a local, distributed message-passing scheme on the graph and computes all the local 4-profiles in parallel. We proved the theory that local 4-profiles can be calculated using compressed two-hop information, and also establish novel concentration results that show that graphs can be substantially sparsified and still retain good approximation quality for the global 4-profile. A generalization of fastest 5-node graphlet counting algorithm was given in [109]. The algorithm requires the existence of a vertex with certain properties which exists for graph lets of arbitrary size, except for complete graphs and a cycle with four nodes which are treated separately. In [110], a combinatorial method for counting graphlets and orbit signatures of network nodes builds a system of equations that connect counts of orbits from graphlets with up to five nodes, which allows to compute all orbits from graphlets with up to five nodes and allows to compute all orbit counts by enumerating just one.

2.11.2 Subgraph Finding

Subgraph computation has many applications. In [38], opinions and behaviors are more homogeneous within than between social groups. Structural holes exist between organizations. Brokerage can help span connections across the structural holes between different groups. The between-group brokers express ideas which are never dismissed and are evaluated as valuable so that brokerage becomes the social capital, which can be identified through subgraph computation. Triadic configurations are the basis for theoretical claims and substantive outcomes. By considering constraints that lower order graph features place on the triad census, triad censuses from 159 social networks of diverse species and social relations are largely explained by their lower order graph features through formal constraints that force triads to occur in narrow range of configuration. Within these constraints, a majority of networks exhibit significant triadic patterning by departing from expectation [71]. The definition of social capital emphasizes the role of social control, in family support, and in benefits mediated by extra familial networks. Several examples were given to review four consequences. We also describe the conceptual stretch of this concept from individual asset to a feature of the communities and even nations, and also examine its limitations, the potential to lead to its heuristic value [203].

Simple models of networks can be tuned through this middle ground: regular networks rewired to introduce increasing amounts of disorder. We find that these systems can be highly clustered, like regular lattices, yet have small characteristic path lengths, like random graphs, which can be called small world network, by analogy with the small-world phenomenon, (popularly known as six degrees of separation). Models of dynamical systems with small-world coupling display enhanced signal-propagation speed, computational power, and synchronizability. Also, infectious diseases spread more easily in small-world networks than in regular lattices [257]. Drawn on the theory of graph homomorphisms to formulate and analyze the representation of domain-independent coordinate system for a collection of graphs from social network, based on the subgraph frequencies, the subgraph frequencies are governed by combinatorial properties as well as empirical properties. The coordinate system examined can be used to classify the structures of different types of social graphs [241]. Standardized measurements in [111] are developed to model the structure in interpersonal relations. A theorem is presented which yields expectations and variance for measurements based on triads.

Although the degree distributions of the scale-free network are all similar, the key topological features of such networks depend heavily on the specific model and the seed graph used. For example, if starting with the right seed graph, the duplicated model can capture many topological features of publicly available protein-protein interaction networks [114]. Protein-protein interaction network use newly introduced measurements of local network structure as well as standardly used measurements of global network structure. Four different Networks models were built, such as Erdos-Renyi, scale-free and geometric random network models. Scale-free model of PPI network fails to fit the data in several respects and a random geometric model provides a much more accurate model of the PPI data. We hypothesize that only the noise in these networks is scale-free [205]. To uncover the structural design principles, network motifs are defined as patterns of interconnections occurring in complex networks, such as biochemistry, neurobiology, ecology, and engineering. In comparison of motifs in different networks, we can draw the conclusion that motifs may be used to define universal classes of networks [178]. Community structure plays a significant role in the analysis of social networks and similar graphs. We hypothesize that any graph with a heavy tailed degree distribution and community structure must contain a scale free collection of dense Erdos-Renyi (ER) subgraphs. A block two-level ER model was proposed and demonstrate that it accurately captures the observable properties of many real-world social networks [218].

Subgraph computation needs to be conducted often, such as counting, enumeration, summarization, especially frequent subgraph listing. The challenge is still disk based solutions which can process massive graphs without maintaining exponential numbers of partial results, in which pattern matching is the solution [200].

2.12 Community Detection and Clustering

Communities are small groups of vertices with common patterns, which means vertices within the groups are similar to each other, and vertices between the groups are different. Community clustering are based on the structure of the edges to group vertices into clusters. The generally used similarity measurements are PageRank scores which are based on the connection between pages and indicates the importance of the pages, conductance which indicates how well-knit the small group is, centrality measurement, such as degree centrality, closeness centrality, betweenness centrality, and eigenvalue centrality.

Before we discuss the methodology of community detection, we can keep in mind some interesting conclusions. After studying and comparing over 100 real world networks [151], these conclusions are generally accepted that large networks have very different structure, tight communities barely connected to the rest of the network, large networks gradually merge into the core of the network and becomes densified, and the growth of the network through forest fire burning process can produce a community profile, and the size of the community can affect the quality of the community [152] so that size-resolved version of the optimization can be considered as a solution, rather than changing the objectives. Size-resolved community structures are useful with these assumptions [121]: (1) the best small groups of nodes can be better than the best large groups, (2) the best small groups of nodes can have a quality that is comparable to the best medium-sized and large groups, and (3) the best small group of nodes can be worse than the best large groups.

2.12.1 PageRank

Pages play important roles in the network. The motivation of PageRank [190] is that pages can be both endorsers and endorsees. Endorsers send endorsement to endorsees across their outgoing links. The more important the endorser pages, the stronger the endorsements they send. All the pages start out with the same PageRank, after infinity number of steps, their PageRanks are converged to several values.

PageRank can be used in many different scenarios, such as citation network in which one author cites several papers and his paper is also cited in other papers, friendship network in which one individual person reached out to make several friends and that person is also in other people's circle of friends, and product recommendation in which you pick one product online, and also receive recommendations of several highly co-purchased products.

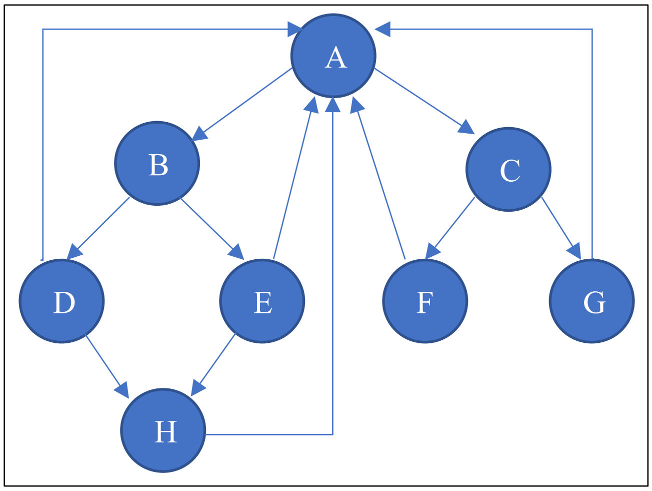


Figure 2.11. A Collection of Eight Pages.

Table 2.1. PageRank Results in Two Steps on Eight Pages

	A	B	C	D	E	F	G	H
Start	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
Step 1	½	1/16	1/16	1/16	1/16	1/16	1/16	1/8
Step 2	5/16	1/4	1/4	1/32	1/32	1/32	1/32	1/16

Table 2.2. PageRank Computation from the Beginning to Step 1 on Eight Pages

Endorsees	Step 0 Results	Endorsers	Formulas	Step 1 Results
A	1/8	D, E, F, G, H	1/2D+E+F+G+H	½
B	1/8	A	1/2A	1/16
C	1/8	A	1/2A	1/16
D	1/8	B	1/2B	1/16
E	1/8	B	1/2B	1/16

Table 2.2. (Continued)

F	1/8	C	1/2C	1/16
G	1/8	C	1/2C	1/16
H	1/8	D, E	1/2D+1/2E	1/8

Table 2.3. PageRank Computation from Step 1 to Step 2 on Eight Pages

Endorsees	Step 1 Results	Endorsers	Formulas	Step 2 Results
A	½	D, E, F, G, H	1/2D+E+F+G+H	5/16
B	1/16	A	1/2A	¼
C	1/16	A	1/2A	¼
D	1/16	B	1/2B	1/32
E	1/16	B	1/2B	1/32
F	1/16	C	1/2C	1/32
G	1/16	C	1/2C	1/32
H	1/8	D, E	1/2D+1/2E	1/16

There is a similar algorithm - Authority Hub algorithm [136]. Both the two algorithms follow the principle of repeated improvement and emphasize the property of Eigen values. The difference of the Authority Hub algorithm is that Authority Hub algorithm uses two update rules: one is authority update rule, and the other is hub update rule. For authority score updates, the connections in adjacent matrix are in-links. For hub score updates, the connections in adjacent matrix are out-links. We can start with either authority update rule or hub update rule, and then switch to the other rule, repeat this process until the changes of the scores between two steps are smaller than a threshold and then stop. No matter what Initial scores we choose, after several steps of updates, the final results can converge to limiting values, which are also a property of the link structure.

PageRank model is a math model which can be modified through its math properties. Preconditioned Chebyshev iteration [227] can run nearly-linear algorithms with preconditioners to solve symmetric, diagonally-dominant linear systems for graph partitioning and graph sparsification. Another useful statistic of social network is heave-tailed degree distribution and large clustering coefficients - imply good communities, also known as ego nets with the existence of vertex neighborhoods. Conductance scores of neighborhood community is as good as Fiedler cut. Also, the conductance of neighborhood communities shows similar behavior as the network community profile computed with personalized Pagerank and neighborhood communities can also help find good PageRank seeds [85]. The second eigenvalue of Laplacian matrix and its associated eigenvector are fundamental features of an undirected graph. They have found their widespread use in scientific computing, machine learning, and data analysis. Graphs have local regions of interest in which the second eigenvector fail to provide information fine-tuned to the local region which is called the locally-biased analogue of the second eigenvector. In [167], local properties of data graphs in a semi-supervised manner provide a solution as a generalization of a personalized PageRank vector. Locally-biased vector is useful for identifying and refining clusters locally.

Based on the empirical discussion that small size communities are more meaningful than large size communities, which give rise to the local PageRank clustering algorithms. A local graph partitioning algorithm [7] finds a cut near a specified vertex, with a running time depending on the size of the small side of the cut, rather than the size of the graph. We proposed a local graph partitioning algorithm using personalized PageRank vectors: approximate PageRank vectors, derive a mixing result for PageRank vectors, derive an analogue of Cheegar inequality for PageRank so that the cut with conductance p can be found in PageRank vectors. In [8], a randomized local partitioning algorithm was presented that finds a sparse cut by simulating the volume-biased evolving set process, which is a Markov chain on sets of vertices. Protein-protein interaction can reveal unknown functional ties between proteins. In [248], Protein network was partitioned to find small communities in the networks. Local Protein Community Finder is a toolkit which quickly finds a community close to a queried protein in any network available from bioGrid or specified by users. Nibble and PageRank-Nibble are implemented in the toolkit. Heat kernel, similar to PageRank, plays the role of diffusion in community detection around a nearby starting seed node. A deterministic local algorithm [137] was presented to compute the diffusion and to study the community. This algorithm use relaxation method within a linear system to estimate the matrix exponential in a degree-weighted norm. It only depends on the parameters of the diffusion not the size of the graph.

Subgraph partition is NP-Complete. The starting seed set plays an important role in generating high quality clusters. Overlapping community [258] is based on local optimization and expansion of a community metric around a seed set of vertices. The key idea is to find good seeds and expand the seed sets using the personalized PageRank clustering procedure. Seed expansion is suited for local community detection. In [138], several variants were evaluated to explore the properties of seed set, use active learning solution, and also explored the topology properties of communities and seed sets that correlate with algorithm performance and explain the empirical observation with theoretical ones.

2.12.2 Conductance

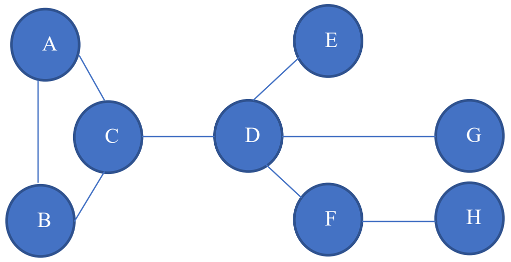
Conductance of a graph G(V, E) is also called Cheeger Constant, which is the ratio of edges across the subgraph over edges within a subgraph. It can measure how well-knit the graph is. The smaller the value of the conductance, the better the cluster. The conductance of a cut $(S, \bar{S})$of a graph G is defined below

$$\varphi(S) = \frac{\sum_{i \in S, j \in \bar{S}} a_{ij}}{\min\,(a(S), a(\bar{S}))} \quad (2.9)$$

$$a(S) = \sum_{i \in S} \sum_{j \in V} a_{ij} \quad (2.10)$$

in which, a_{ij} are entries of the adjacent matrix. In Figure 2.12, we give an example of the conductance measurement for graph clustering.

In [265],13 structural definition of network communities is examined based on their sensitivity, robustness, and performance. Conductance and triad-participation-ratio are proved to be better measurements in evaluating the quality of the community. Also, the local spectral clustering algorithm was expanded into a heuristic parameter-free community detection method that easily scale to network with more than hundreds of nodes. In [83], a similar measurement was given to provide a worst-case guarantee for the quality of the approximation, which is as good as the Cheeger Constant measure.



Cluster	Conductance
{A}	2/min(2,16-2)=1
{A,B}	2/min(4,16-4)=1/2
{A,B,C}	1/min(7,16-7)=1/7
{A,B,C,D}	3/min(11,16-11)=3/5

Figure 2.12. Graph Clustering with Conductance Measurement.

Conductance measurement is generally used in graph partitioning and graph clustering. In [139], bicriteria approximation algorithms for the small sparsest cut problem can be implemented locally using truncated random walk with running time almost linear to the output size. It provides a local graph partitioning algorithm with a better conductance guarantee when k is sublunar. A modification of normalized cuts is provided in [168] for constrained image segmentation: 1. It allows us to use noisy top-down information 2. Constrained solution can be computed in small additional time. Free riders are irrelevant vertices in a community. In [262], systematic explanation of the existence of free rider and an algorithm is developed, based on query biased node weighting scheme to reduce the free rider effect. A new clustering algorithm is proposed in [228], which can be used to partition graphs into subgraphs and find an approximate sparsest cut with near optimal balance. Protein-protein interaction network analysis can be split into two categories: local alignment and global alignment. In [248], global alignment through a multiple network alignment tool was provided, which was based on spectral clustering on induced graph of pairwise alignment scores to better identify functional orthologs across species.

2.12.3 Deep Learning

Deep learning [63][180][216] combines traditional neural network and machine learning algorithms together to perform classification, clustering, and sequence analysis. This prototype has several advantages. First, feature engineering process is built into deep learning framework. During parameter tuning, significant features can be promoted and unsignificant features can be suppressed. Secondly, the number of dimensions and the number of parameters can be reduced significantly. Also, linear algebra naturally fits into the deep learning prototype which makes it easy to operate both vectors and matrix.

Deep learning prototype can be divided into several categories: feature function learning framework [98], gated learning framework [153], and attention learning framework [245]. Convolution network can be considered as a special case of attention learning framework.

Feature function learning framework trains a model to generate embeddings by engineering features for a node with its local neighbors. Learned feature functions can be trained with low dimensional embeddings of nodes in large graphs. Local neighborhood information is integrated together to interpret the characteristics of the nodes. After training, learned feature functions can operate on unseen nodes as well.

Gated Graph sequence neural networks (GGs-NNs) expands the output neural network to a sequence instead of a single class. GGs-NNs has two settings: specifying all intermediate annotations and training the full model end-to-end given graphs and target sequences. The former generates sequence outputs with observed annotations which can improve performance when we have domain knowledge about specific intermediate information that should be represented in the internal state of the nodes. The latter generates sequence outputs with latent annotations which is more general. After training, GGs-NNs learns features on graph structures data through neural networks.

Similar to GGs-NNs, graph attention network also output a sequence of features. However, attention networks can learn the characteristics of the nodes not only with neighborhood information but also beyond neighbor nodes on the networks. The attention mechanism is a single-layer feedforward neural network, fully expanded out to the entire network by normalizing the coefficients of the entire graph and by repeatedly train the layer for several literatures.

Chapter 3: Notation and Terminology

3.1 Notation

This section provides a reference for the most used notation. In each chapter, additional notation may be introduced if necessary.

Table 3.1. Notation Symbols and Explanations

Symbol	Explanation
G	A graph
V(G)	Vertices of graph G
E(G), (v,w), vw	Edges of graph G. v and w are vertices of G
n	The number of vertices.
m	The number of edges.
deg(v), N(v)	The number of edges incident with vertex v is he degree of v.
Δ	The maximum degree in G
k-regular	If all the vertices of G have the same degree k, then G is regular of degree k.
D	A directed graph D is the one in which each edge is assigned a direction.

3.2 Graph

A *Countable Graph* is the one in which V(G) and E(G) are finite or countably infinite. A locally finite graph is the one in which the number of edges incident with each vertex is finite. If V(G) and E(G) are infinite, the graph is called infinite graph.

Two graphs G and H are *isomorphic* if there is a one-to-one correspondence between their vertex sets that preserves that adjacency of vertices. An automorphism of G is a one-to-one mapping ϕ of V(G) onto itself with the property that ϕ (v) and ϕ (w) are adjacent if and only if v and w are adjacent. An automorphisms of G form a group Γ(G) under composition, called the automorphism group of G.

For digraph D, if e=vw is an arc of a digraph D, then v and w are adjacent, and e is incident from v and incident to w. If v is a vertex of a digraph d, then its out-degree outdeg(v) is the number of arcs in D of the form vw, and its in-degree indeg(v) is the number of arcs in D of the form wv.

Let G be a graph with vertex set {v_1, v_2,…, v_n} and edge set {e_1, e_2, …, e_n}. The *adjacency matrix* of G is the n*n matrix A(G) = (a_{ij}) where

$$a_{ij} = \begin{cases} 1, if\ v_i\ and\ v_j\ are\ adjacent, \\ 0, if\ not, \end{cases} \quad (3.1)$$

and the incident matrix of G is the n*m matrix B(G)=(bij), where

$$b_{ij} = \begin{cases} 1, if\ v_i\ is\ incident\ with\ e_j, \\ 0, if\ not, \end{cases} \quad (3.2)$$

The eigenvalues of A(G) are independent of the way in which the vertices are labelled, called the eigenvalues of G, and the characteristic polynomial of A(G) is the characteristic polynomial of G.

If an automorphisms of G form a group Γ(G) under composition, Γ(G) is *transitive* if it contains automorphisms mapping each edge of G to every other edges.

A sequence of edges of the form v_0v_1, v_1v_2, …, $v_{r-1}v_r$, is a walk of length r between v_1 and v_r. If these edges are all distinct then the walk is a *trail*. If the vertices v_0, v_1, …, v_r are also distinct, then the walk is a *path*. Two paths are edge-disjoint if they have no common edges and are vertex-disjoint if they have no common vertices. A walk or trail is closed if $v_0=v_r$, and for $r > 0$, a closed walk with all v_0, v_1, …, v_{r-1} distinct is a *cycle*.

A graph G is connected if there is a path joining each pair of vertices of G; otherwise, it is disconnected. Every disconnected graph can be split into maximal connected subgraphs, called *components.*

A *subgraph* of a graph G=(V(G), E(G)) is a graph H=(V(H),E(H)) such that $V(H)\subseteq V(G)$, and $E(H)\subseteq E(G)$. If e is an edge of G, then the edge-deleted subgraph G-e or G\e is the graph obtained from G by removing the edge e. If v is a vertex of G, then the vertex-deleted subgraph G-v is the graph obtained from G by removing the vertex v together with all its incident edges.

We can also obtain a new graph from G by removing the edge e=vw and identifying v and w so that the resulting vertex is incident to all edges that were originally incident with v or w. This is called contracting the edge e and G is contractible to H=G-e. A minor of G is any graph obtained from G by a succession of edge-deletion and edge-contractions.

If e=vw is an edge of G, then we obtain a new graph by replacing e by two new edges vz and zw, where z is a new vertex. This is called *subdividing* the edge. Two graphs that can be obtained from the same graph by *subdividing* its edges are homeomorphic.

The *complement* of $\bar{G}$ of G is the graph with the same vertex set as G, but where two vertices are adjacent whenever they are not adjacent in G.

A connected graph G is *Eulerian* if and only if each vertex of G has even degree. A connected digraph D is Eulerian if and only if the in-degree and out-degree of each vertex are equal.

If G is a simple graph with n (≥3) vertices, and if deg(v)+deg(w) ≥n for each pair of non-adjacent vertices v and w, then G is *Hamiltonian*.

Given a graph X, a permutation α of V(X) is an *automorphism* of X if for all u, v∈ V(X)

$$\{u, v\} \in E(X) \Leftrightarrow \{\alpha(u), \alpha(v)\} \in E(X) \qquad (3.3)$$

The set of all *automorphisms* of a graph X, under the operation of composition of functions, forms a subgroup of the symmetric group on V(X) called the *automorphism* group of X, and it is denoted Aut(X).

Let the components of X be $X_1, X_2, \ldots, X_k$. Then

$$Aut(X) = \prod_{i=1}^{k} Aut(X_i) \qquad (3.4)$$

$$Aut(X) = Aut(\bar{X}) \qquad (3.5)$$

A group G of permutations of a set S acts transitively or is transitive on S if for every x, y ∈ S, there exists $\alpha \in G$, such that $\alpha(x)=y$, is vertex-transitive if Aut(X) acts transitiviely on V(X), and acts doubly transitively on S if for any two ordered pairs of distinct elements $(x_1, x_2),(y_1,y_2) \in S\times S$ there exists $\alpha \in$ G such that $\alpha(x_1)=y_1$ and $\alpha(x_2)=y_2$.

For i = 1,2, let G be a group of permutations of the set S_i. We say that G_1 and G_2 are isomorphic as permutation groups if there exist a α group-isomorphism: $G_1\rightarrow G_2$ and a bijection f: $S_1\rightarrow S_2$ such that $f(\alpha(x)) = [\Phi(\alpha)](f(x))$ for all $\alpha \in G_1$, $X \in S_1$.

An edge-isomorphism from a graph X_1 to a graph X_2 is a bijection η: $E(X_1) \rightarrow E(X_2)$ such that edges e_1 and e_2 are incident with a common vertex of X_1 if and only if $\eta(e_1)$ and $\eta(e_2)$ are incident with a common vertex of X_2.

An edge-automorphism is an edge-isomorphism from a graph to itself. The set of edge-automorphisms forms a subgroup of the symmetric group on E(X), called the edge-group of X.

A graph product of graphs X and Y is a graph with vertex set V(X) ×V(Y), whose edge set is determined in a prescribed way by and only by the adjacency relations in X and in Y. A graph product & is associative if (W&X)&Y≅W&(X&Y), for all graphs W, X, Y.

A graph X is a divisor of a graph Z with respect to a product &, if there exists a graph Y such that

$$Z=X\&Y \wedge Z=Y\&X \qquad (3.6)$$

For all graphs W, X, Y. Interest is usually restricted to products that are associative. Four commonly used associative graph products: Let Z be a graph product of arbitrary graphs X and Y. Let x_1, x_2 be vertices of X, and let y_1, y_2 be vertices of Y. Suppose that

$$\{(x_1, y_1), (x_2, y_2)\} \in E(Z) \qquad (3.7)$$

- In the Cartesian product Z=X ⊡Y

$$[\{x_1, x_2\} \in E(X) \wedge y_1 = y_2] or [\{x_1 = x_2\} \wedge \{y_1, y_2\} \in E(Y)] \qquad (3.8)$$

- In the strong product Z=X ⊠Y

$$[\{x_1, x_2\} \in E(X) \wedge y_1 = y_2]\ or$$

$$[x_1 = x_2 \wedge \{y_1, y_2\} \in E(Y)]\ or$$

$$[\{x_1, x_2\} \in E(X) \wedge \{y_1, y_2\} \in E(Y)] \quad (3.10)$$

- In the weak product Z=X ×Y

$$\{x_1, x_2\} \in E(X)\ and\ \{y_1, y_2\} \in V(Y) \qquad (3.11)$$

- In the lexicographic product Z=X[Y]

$$\{x_1, x_2\} \in E(X)\ or\ [x_1 = x_2 \wedge \{y_1, y_2\} \in E(Y)] \qquad (3.12)$$

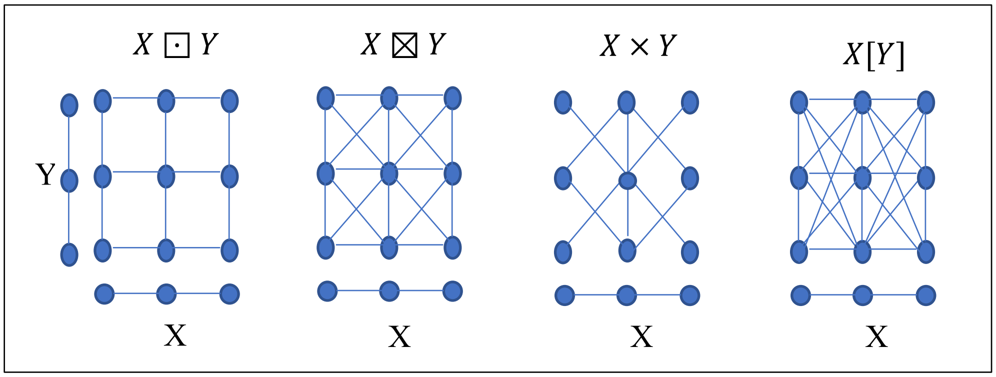


Figure 3.1. The Four Products of the 2-Path by the 2-Path

These four products are illustrated in Figure 3.1. Graph X and Y are relatively prime if they have no common proper divisor.

3.3 Topologies

We list pure topology types for networks in Figure 3.2. Graph sampling needs to keep both the graph topology and other graph properties. The topology type of a network can be determined based on either the network property or the scenario of the application.

1. Topology 1. Ring Lattice Network. As shown in Figure 3.2(1), each node is connected to its neighbors, according to the ring-induced distance. Ring lattice network forms a regular graph and is vertex-transitive. To identify ring lattice network, we can start with any vertex and then check the degree of the vertex and the edges between the vertex and its adjacent neighbors.
2. Topology 2. Small world network. As shown in Figure 3.2(2), each node is connected to several of its neighbors and few distant nodes, according to the ring-induced distance. We can consider small world network is a ring lattice network with several random edges. From ring lattice network, to small world network to Erdos random network, the randomness of the network increases.

3. Topology 3. Erdos Random network. As shown in Figure 3.2(3), each node is connected to a random set of the remaining nodes. Erdos random network has random connections between vertices. The degree distribution follows a power law.
4. Topology 4. Core-Periphery network. As shown in Figure 3.2(4), nodes belong exclusively to either the core or the periphery. Core and periphery nodes are connected to core nodes, while there are no edges among periphery nodes. Core-periphery network has two different sets of vertices: core vertices and periphery vertices. Core vertices have one or more than one degree. Periphery vertices have one degree.
5. Topology 5. Scale free network. As shown in Figure 3.2(5), most of the nodes are connected to few other nodes, while few nodes are connected to many other nodes. This relation is formally described with a power law, between the number of edges and the number of connections. We can consider scale free network as a core-periphery network with several random connections. In scale free network, most of the vertices have a small number of degrees, and a few vertices have a large number of degrees.
6. Topology 6. Cellular network. As shown in Figure 3.2(6), nodes are divided into cells. Connections are frequent between nodes within each cell, and rare between nodes in different cells. Cellular network consists of several subgraphs which are cells. Within the cells, vertices are connected to each other closely so that there are more connections. Between the cells, vertices are rarely connected to each other so that there are less connections. Subgraphs can be identified through clustering.

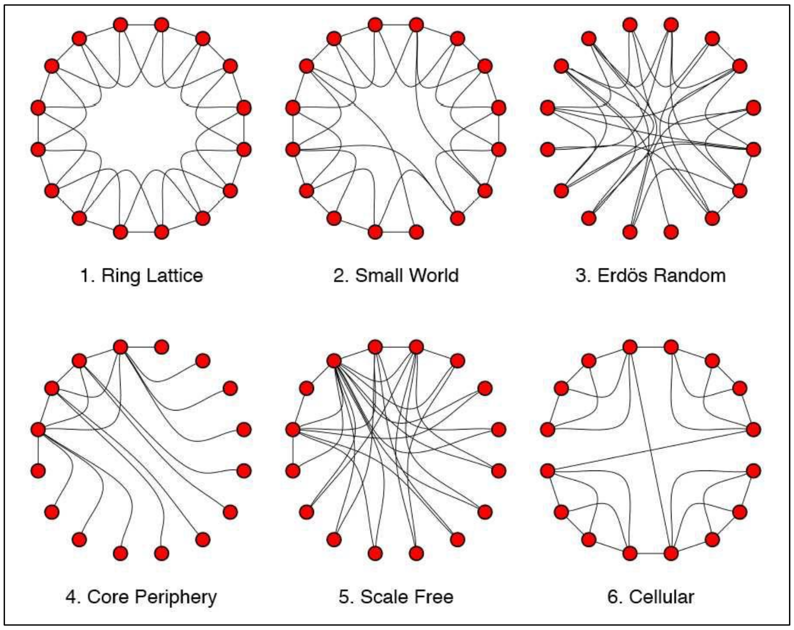


Figure 3.2. Pure Topology Types [5]

3.4 Data Sampling

Data sampling is the processing of selecting samples from task or problem specific domains with the objective of estimating the population parameters. Sampling and resampling are two different concepts. Data sampling focuses on what is the best way to represent the characteristics of the task or problem with the sample. Resampling focuses on what is the best way to quantify the uncertainty of the estimate.

In term of methodology, data sampling can be conducted through simple random sampling, systematic sampling, and stratified sampling. However, no matter which methodology is selected, we also need to define sample goal, population, selection criteria and sample size. Random sampling can be performed by selecting samples with a uniform probability from the domain. Systematic sampling can be performed by selecting samples with a specified pattern, such as intervals. Stratified sampling works on the population which can be partitioned into subpopulations.

Stratified sampling is also called stratified purposive sampling. It can ensure the characteristics of the study sample represent the characteristics of the task or problem specific population. For example, we conduct a survey in three towns with 1 million, 2 million and 3 million voters in population. when applying stratified sampling, 100 voters are selected from 1 million in town A, 200 voters are selected from 2 million in town B, and 300 voters are selected from 3 million in town C, instead of randomly selecting 600 from 6 million. The influence factors can be any demographic factors, other than town, such as gender, age, education, income, and so on. However, the selection of the influence factors requires prior knowledge about data collection and what variables need to be controlled. Moreover, stratified sampling can efficiently reduce sample errors when the partition can be influenced by domain specific factors.

3.5 Graph Sampling

To obtain a subset of the population, we apply data sampling to select individual members through probabilistic or non-probabilistic methodologies. The selected subset is transformed from the original data set, has smaller size, and represents the characteristics of the original data set. Since sampling is a time-convenient and cost-effective way to prepare, transform and research the entire population, it is the basis of research design.

Graph sampling is a simple and effective way to transform large graphs into a smaller one. If we consider the generation algorithm as a family of graphs, the procedure of graph sampling is to sample one graph from the family. Different from the individual and independent data points, graphs have both vertices and connections between vertices. When sampling comes into play, graph sampling needs to handle more variables than individual independent data points.

Graph sampling is needed for several reasons, such as lack of data, hidden population survey, graph sparsification, reduce test cost, and visualization.

Lack of data happens when the entire data set is related to large population and randomly partially selected sample sets cannot be used to represent all the properties of the entire population. In such a scenario, we can randomly start with a small set of seed vertices and crawl them. The quantity of the sample set is determined by the number vertices we start with and the number of hops we crawl. To ensure a good coverage, we can evaluate the sample sets by measuring graph properties, such as degree centrality, cosine centrality, coefficient centrality and so on.

Hidden populations can be explored through sampling algorithms. Hidden populations cannot be directly enumerated and sampled in the entire population, for example, drug abusers in the sociology study. To sample the hidden population, we can define a heuristic procedure, start with a small set of seed vertices, and expand the sample set along the edges. The quality of the sample set can be evaluated through both the graph properties and the usefulness of the sample set for a certain application.

Graph sparsification makes it difficult to load useful information into limited space by manipulating the entire graph. Graph sparsification is a classical problem. It can happen in vertices, edges and even both and can be solved by adding constraints on the transformation, such as preserving all cuts, etc.

To reduce test cost, the most important factor is the number of samples. In biochemical research, the tests in protein interaction network are very expensive. In clinical study, it is difficult to collect many patients to participant in research.

For graph visualization, the graphs must fit into the screen and to be readable for human. Graph sampling can be used to aggregate similar records, summarize the characteristics of the original graphs, and reduce the number of records in the sample set. After graph sampling, the sample set can be easily visualized, show all the preserved graph properties, and show the preserved characteristics of the original graphs.

Graph sampling methodology needs to satisfy two requirements. First, we need to get representation subset of vertices. When the valuation of the attribute in the population cannot be directly used, for example, the drug abusers, the sample set needs to be discovered through sampling algorithms. Normally, the hidden attributes can be represented with either the combination of several attributes or the output of the attribute filters. Secondly, the sample graph needs to preserve certain properties of the original graph, at least, to preserve certain properties within a certain error margin. After sampling, graph properties can be preserved within a certain confidence and graph algorithms can be conducted on the sample set.

3.6 Feature Extraction

Feature extraction is the process of reducing dimensions by evaluating the relevance between variables and targets. After removing non-informative and redundant variables, several advantages can be achieved in data modeling, for example, computation costs can be lowered, noise in inputs can be decreased, the contribution of relevant variables can be magnified, and the performance of the model can be improved.

Although feature representation is problem or domain specific, once all the variables potentially influencing the change of the target variables are collected, the evaluation and extraction process can be converted into machine learning problems, such as supervised feature extraction and unsupervised feature extraction. For supervised feature extraction, the training set is needed to build subsets of the features. On the other hand, unsupervised feature extraction does not require the training set. However, both supervised and unsupervised feature extraction can have the statistical theories behind to perform the hypothesis testing.

The methodology for feature extraction can be divided into several categories, based on data types. The data can be both numerical variables and categorical variables. For numerical input and numerical output, the most common correlation measures are Pearson's correlation coefficient, Spearman's rank coefficient and so on. For numerical input and categorical output, the most common correlation measures are ANOVA correlation coefficient, Kendall's rank coefficient, and so on. For categorical input and numerical output, the process can be defined as reversed "numerical input and categorical output". For categorical input and categorical output, the most common correlation measures are chi-squared test, mutual information and so on.

3.7 Prior Knowledge

In data analysis, prior knowledge in application domains can bring big impacts in each one of the steps. For problem formulation, domain knowledge can help us form precise and accurate problem definitions. For data preparation, domain knowledge can efficiently reduce the burden of data collection and improve the quality of the data sets. For data preprocessing, domain knowledge can help us have a high-level understanding of the valuation of the variables, the correlation between the variables, and the noise, which paves the way to the proper data processing methodology. For data modeling, domain knowledge can be used to evaluate the quality of the results. For result interpretation, domain knowledge can help understand insights, correctness, and flaws.

AlphaGo Zero is a perfect example of how beneficial the prior knowledge is. AlphaGo Zero was trained by Google by combining the rules of Go. AlphaGo Zero defeated both human players and AlphaGo which was trained with the expertise of expert Go players. In real-world applications, human, as experts in the field, may have partial but not all prior knowledge, such as all the rules of Go, so that it is impossible to train a perfect system. However, this example also shows that the more prior knowledge we apply to the model training, the more accurate the system can be.

However, it is difficult to combine prior knowledge to data modeling. When the data set size is too big to be handled by human, when rules and boundaries are ambiguous, especially when noise is involved, the question is which prior knowledge is really related and how complete the prior knowledge is.

3.8 Computation Graphs

Computational graphs are generally used in Neural Networks (NN) to represent mathematical expressions in network architecture and describe required computation provided by the network. Computational graphs have two directions: one is forward computation; the other is backward computation.

As shown in Figure 3.3, computation graphs include nodes and edges in which nodes can be used to represent variables, and edges can be used to represent data dependencies and function arguments. From one layer to next layer, operations are the way to combine variables and to build functions.

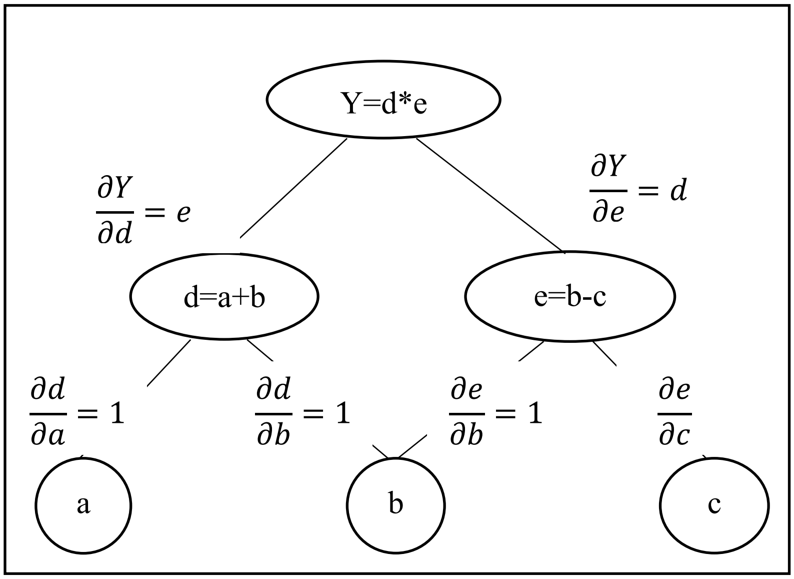

Y=d*e
$\frac{\partial Y}{\partial d} = e$
$\frac{\partial Y}{\partial e} = d$
d=a+b
e=b-c
$\frac{\partial d}{\partial a} = 1$
$\frac{\partial d}{\partial b} = 1$
$\frac{\partial e}{\partial b} = 1$
$\frac{\partial e}{\partial c}$
a
b
c

Figure 3.3. Computation Graphs

In Figure 3.3, we demonstrate how to use a computation graph to compute derivative. On the bottom layer are the input variables – a, b, c. On the middle layer are the functions to compute d and e: $d = a + b$, $e = b - c$. On the top layer is the function to compute Y: $Y = d * e$. Based on these functions, we can have the derivative: $\frac{\partial d}{\partial a} = 1$, $\frac{\partial d}{\partial b} = 1$, $\frac{\partial e}{\partial b} = 1$, $\frac{\partial e}{\partial c} = -1$, $\frac{\partial Y}{\partial d} = e$, and $\frac{\partial Y}{\partial e} = d$. Since $\frac{\partial Y}{\partial a} = \frac{\partial Y}{\partial d} * \frac{\partial d}{\partial a} = e * 1 = e$, $\frac{\partial Y}{\partial b} = \frac{\partial Y}{\partial d} * \frac{\partial d}{\partial b} = e * 1 = e$, $\frac{\partial Y}{\partial c} = \frac{\partial Y}{\partial e} * \frac{\partial e}{\partial c} = d * (-1) = -d$. This example simulates the computational unit in NN. The architecture of NN can be the combination of multiple computation units and even the combination of several sub networks. We will illustrate different architectures, such as Multi-Layer Perceptron Networks (MLP) [209], Recurrent Neural Networks (RNN) [87], Convolution Neural Networks (CNN) [144], Long Short Term Memory (LSTM) [79] and so on.

Computation graphs are adaptable to complex computations by allowing self-circles, interleave graph generation and evaluation. Forward and backward computation can be built into one process to repeatedly optimize parameters. The drawbacks are dealing with high dimensional data through multiple iterations can be time-consuming and untraceable.

3.9 Deep Neural Networks

Deep learning is a branch of machine learning, in which computation graph can be used to build hierarchical structures, so that deep learning is also called hierarchical learning. There are several advantages in hierarchical models, for example, the performance of deep learning models scales with the amount of data they are trained on so that the more data, the better the model; also, deep learning is capable to automatically promote significant features and suppress insignificant features. In other words, deep learning normally outperforms traditional machine learning.

Deep learning is generally used in image recognition, text mining, and graph analysis. We use Figure 3.4 to formulate the computation process in deep learning. As shown in Figure 3.4, for vertex 5, given output O, weights w, and bias θ from the previous layer, we can compute output Oj, Error Errj, the change of weights wij and the change of bias θ_j.

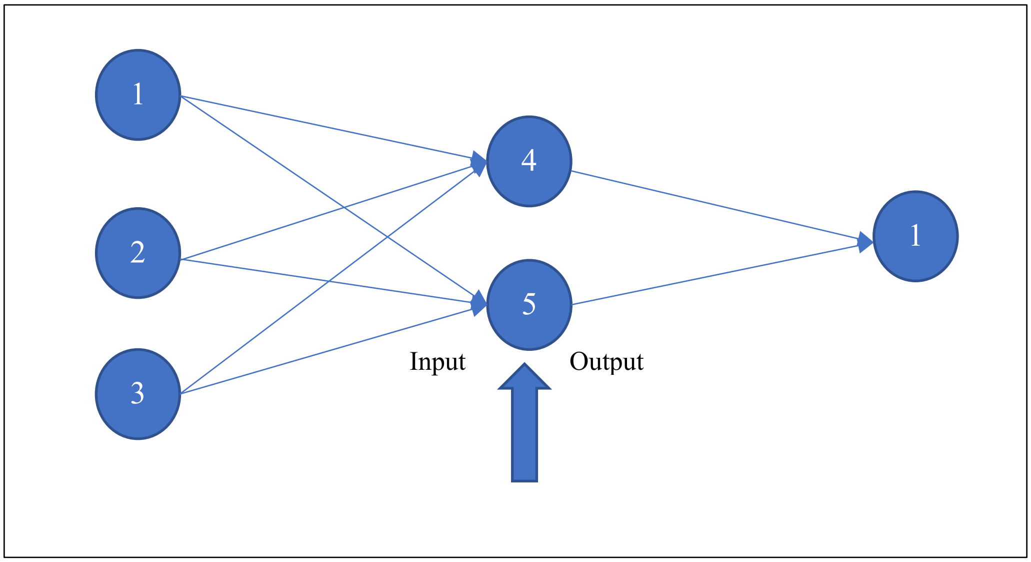


Figure 3.4. Deep Learning Prototype

Feature selection in deep learning can be included in modeling in several different ways which can be interpreted as the following.

$$I_j = \sum_i w_{ij} * O_i + \theta_j \qquad (3.13)$$

$$O_j = \frac{1}{1 + e^{-I_j}} \qquad (3.14)$$

$$Err_j = O_j * (1 - O_j) \sum_k Err_k w_{kj} \qquad (3.15)$$

$$\Delta w_{ij} = l * E_{rrj} O_i \qquad (3.16)$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \qquad (3.17)$$

$$\Delta \theta_j = l * Err_j \qquad (3.18)$$

$$\theta_j = \theta_j + \Delta \theta_j \qquad (3.19)$$

In which I_j is the input matrix of j-th layer, O_j is the output matrix of j-th layer, Err_j is the error matrix of j-th layer, w_{ij} is the weight on the i-th vertex, the j-th layer, Δw_{ij} is the change of weight on the i-th vertex and the j-th layer, θ_{ij} is the bias on the i-th vertex and the j-th layer, $\Delta\theta_{ij}$ is the change of bias on i-th vertex and the j-th layer.

As a convention, we use $x \in X$, $y \in Y$ to represent the input and the output, parameterized by θ, so that the model can be formulated as

$$y = F(x, \theta) \quad (3.20)$$

Each hidden layer h^l can be computed from the previous layer,

$$h^l = f_l(h^{l-1}, \theta) \qquad (3.21)$$

$$f_l(h^l, \theta) = \sigma(W^l h^l + b^l) \qquad (3.22)$$

In which, $h^l \in R^{H_l}$ for the l-th layer, weight matrix $W^l \in R^{H_l \times H_l}$ for the l-th layer, bias matrix $b^l \in R^{H_l}$ for the l-th layer. Typical choice for $\sigma(x)$ is the logistic sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.

Multi-layer models can be defined as

$$F(X, \theta) = f_L(f_{L-1}(\dots(f_1(x, \theta^1)\dots, \theta^{L-1}), \theta^L) \qquad (3.23)$$

Deep learning has a vast number of variants, differing mostly in the architecture. Moreover, the weight matrix and bias matrix can be tuned differently.

3.10 Graph Neural Networks

Graph neural networks (GNNs) extend recurrent neural networks, which can process most of the graphs and can be applied on both graphs and nodes focused problems. Graphs can be positional and non-positional. The intuitive idea underlying GNN is that nodes in a graph represent objects or concepts and edges represent their relationships.

The learning problem is always implemented by the minimization of the quadratic error function. Several optimization algorithms can be used: almost all of them are based on sub procedure that computes the error gradient w.r.t the weights. The possible optimization methods include, for instance, gradient descent, scaled conjugate gradient, Levenberg-Marquardt and resilient backpropagation.

The unfolding tree often contains most of the original information, in theory the pre-processing may cause a loss of information, because it may happen that two different graphs and nods are unfolded to the same tree. On the other hand, a theoretical condition ensuring the lossless unfolding is described in [33] that the generated tree contains the same information of the original graph.

3.11 Graph Kernel

In kernel methods, all computations are done via a kernel function which is the inner product of two vectors in a feature space [133], [221]. There are several kernel methods. Most of them are based on the idea of an object that can be decomposed into substructures and a feature vector that is composed of the counts of the substructures.

3.12 Graph Propagation

One interesting class of neural networks are dynamical systems with three salient properties [201]. First, they possess very many degrees of freedom, second, their dynamics are nonlinear and, third, their dynamics are dissipative. Systems with these properties can have complicated attractor structures and can exhibit computational abilities.

A learning algorithm is a rule or dynamical equation which changes the location of fixed points to encode information. One way of doing this is to minimize, by gradient descent, some function of the system parameters. The formalism is illustrated by deriving adaptive equations for a recurrent network with first order neurons, a recurrent network with higher order neurons and finally a recurrent first order associative memory.

Weisfeiler-Lehman [221] can be used to map the original graph to a sequence of graphs, whose node attributes capture topological and label information to measure the similarity of graphs. The kernel of WL is a function of two graphs that quantifies their similarity.

3.13 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) [79] is a modified edition of Recurrent Neural Network (RNN) which can be used to continuously go through text sequences and to build a word network. The network information is stored in memory cells and three adaptive, multiplicative gating units shared by all cells in the block as shown in Figure 3.5. Also, each memory cell has a recurrent self-connected linear unit - Constant Error Carousel (CEC), which is the core of the cell and stores the state of the cell, as shown in Figure 3.5. The Constant Error Carousel (CEC) overcomes a fundamental problem in RNNs and can prevent error signals from decaying quickly.

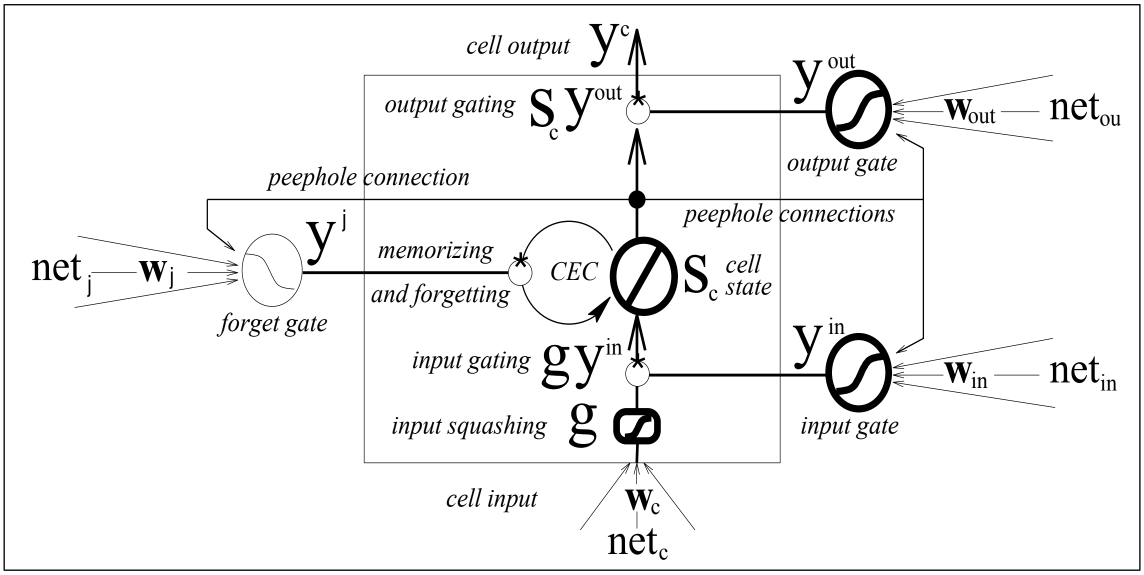


Figure 3.5. Long Short-Term Memory Prototype [80]

LSTM can learn the fine distinction between sequences of spikes spaced either 30- or 50-time steps apart without the help of any short training exemplars. Temporal distance between events conveys information essential for numerous sequential tasks, such as speech recognition, motor control and rhythm detection. This makes LSTM a promising approach for tasks that require the accurate measurement or generation of time intervals.

The cell output, y_c, is calculated based on the current cell state s_c and four sources of input: net_c is input to the cell itself while net_{in}, net_φ and net_{out} are inputs to the input gates, forget gates, and output gates. We consider discrete time steps t = 0,1,2,… A single step involves the update of all units (forward pass) and the computation of error signals for all weights (backward pass). For each discrete time step, we use a two-phase update scheme that computes as the following:

1. Phase one:
    a. Input gate activation y^{in},
    b. Forget gate activation y^{φ},
    c. Cell input and cell state s_c,
2. Phase two:
    a. The input gate activation y^{in} and the forget gate activation y^{φ} are computed as follows

$$net_{in_j}(t) = \sum_m w_{in_j m} y^m(t-1) + \sum_{v=1}^{s_j} w_{in_j c_j^v} s_{c_j^v}(t-1), y^{in_j}(t) = f_{in_j}(net_{in_j}(t)) \qquad (3.24)$$

$$net_{\varphi_j}(t) = \sum_m w_{\varphi_j m} y^m(t-1) + \sum_{v=1}^{s_j} w_{\varphi_j c_j^v} s_{c_j^v}(t-1), y^{\varphi_j}(t) = f_{\varphi_j}(net_{\varphi_j}(t)) \qquad (3.25)$$

and the state of memory cell s_c is calculated by adding the squashed, gated input of the cell to the state at the previous time step $s_c(t-1)(t>0)$, which is multiplied (gated) by the forget gate activation ($s_{c_j^v}(0) = 0$)

$$net_{c_j^v}(t) = \sum_m w_{c_j^v m} y^m(t-1), s_{c_j^v}(t) = y^{\varphi_j}(t) s_{c_j^v}(t-1) + y^{in_j}(t) g(net_{c_j^v}(t)) \quad (3.26)$$

b. The output gate activation y^{out} is computed as follows

$$net_{out_j}(t) = \sum_m w_{out_j m} y^m(t-1) + \sum_{v=1}^{s_j} w_{out_j c_j^v} s_{c_j^v}(t), y^{out_j}(t) = f_{out_j}(net_{out_j}(t)) \quad (3.27)$$

c. The cell output is computed as follows

$$y^{c_j^v}(t) = y^{out_j}(t) s_{c_j^v}(t) \quad (3.28)$$

Finally, assuming a layered network topology with a standard input layer, a hidden layer consisting of memory blocks, and a standard output layer, the equation for the output units k are:

$$net_k(t) = \sum_m w_{km} y^m(t), y^k(t) = f_k(net_k(t)) \quad (3.29)$$

where m ranges over all units feeding the output units and f_k is the output squashing function.

Generally used LSTM are Bidirectional LSTM which captures both the past and the future information. The standard LSTM architecture allows strictly sequential information propagation. Multilayer LSTM which let the higher layers capture longer term dependencies of the input sequence. Tree structured LSTM [235] allow for richer network topologies where each LSTM unit can incorporate information from multiple child units.

Order-insensitive models are insufficient to fully capture the semantics of natural language due to their inability to account for differences in meaning because of differences in word order or syntactic structure. Order-sensitive sequential or tree-structured models are a linguistically attractive option due to their relation to syntactic interpretation of sentence structure.

3.14 Problem Specific Neural Networks

Kernel methods represent the characteristics of the data set which can be based on common substructures, statistical model, and local transformation of data. [99]

Simple kernels defined on subparts of given structures can be extended by generic operations to complex and convolutional kernels. These simple kernels normally have self-similar property, such as sequences and graphs.

For sequences, the string kernels can represent a sequence of entities in a particular order. Long Short-Term Memory (LSTM) with forget gates [79] integrates sequential data into deep learning prototype by adding memory block into the prototype. The memory block contains one or more memory cells which have recurrently self-connected linear units, called Constant Error Carousel (CEC). By recirculating activation and error signals indefinitely, the CEC provides short-term memory storage for extended time periods. The input, forget, and output gate can be trained to learn what information to store in the memory, how long to store it, and when to read it out.

For graphs, the graph kernels can represent the networked data with vertices and edges: vertices represent entities and edges represent relationships. Images can be considered as a special type of networked data in which vertices are positional. Graph neural network (GNN) is based on neural network model and can fit graph data into it. In GNN, each node is attached with neighborhood information. Vertex information can be updated based on neighborhood information. For positional graphs, vertex propagation can be applied to entire graphs. For non-positional graphs, vertex propagation can be conducted along the neighbors of the vertices.

For kernels based on a statistical model, finding an appropriate kernel function for a particular problem can be difficult and sometimes may be largely unresolved issues. In graph analysis, several statistical kernels are already discovered. For example, the distribution of the number of vertices is Gaussian, the distribution of the number of edges given vertices is exponential, and the distribution of the number of vertices with degree k follows Poisson distribution. [5]

For kernels based on local transformation of data, local transformation of data can be used to highlight specific features. For example, in image processing, different transformation kernels can be used for blurring, sharpening, embossing, edge detection, and more. This can be done by doing a convolution between a kernel and an image, which adds each element of the image to its local neighbors, weighted by the kernel. For graph neural network, graph propagation can be conducted among neighbors and the operation can be specified to result in a particular selection, such as maximum, minimum, mean, median and customized regression.

Chapter 4: Graph Space Management and Parallel Planning

4.1 Introduction

In parallel planning, both space management and job scheduling need to be taken care of before parallel execution. Only limited space can be assigned to all the tasks which need to be scheduled simultaneously.

Data stream bin packing can be used to estimate both the upper bound and the lower bound of the space cost and time cost, which can be done through minimizing the number of bins we can use, and the maximum number of items (or the maximum duration of jobs) in each bin. By using bin packing to optimize both the space complexity and the task scheduling, we can efficiently manage data store, schedule data processing, and find a way to split data and to schedule job execution for the best of both world. This problem can be modelled as vector bin packing, as shown below,

Minimize Σy_j, (4.1)

Subject to $\Sigma a_i x_{ij} \leq C y_j,\ j \in J$, (4.2)

$\Sigma x_{ij} = 1,\ i \in I$, (4.3)

$x_{ij},\ y_j \in \{0, 1\},\ i \in I,\ j \in J.$ (4.4)

in which, $Y= \{y_j \mid y_j \in \{0, 1\},\ j = 1, \ldots, m\}$ is the set of boxes, $X = \{x_i \mid x_i \in \{0, 1\}, i = 1, \ldots, n\}$ is the set of items, a_i is the size of the item i. The constraint (4.1) can ensure that the number of boxes needs to be minimized. The constraint (4.2) can ensure that the total size of items in each box cannot exceed the capacity of the box. The constraint (4.3) can ensure that one item can only be in box.

There are several scenarios for bin packing. One is that we need to put a bunch of stuff into a knapsack, and we need to put as much stuff into the knapsack as possible. The other one is that we have a list of boxes and a bunch of items. We need to use the minimum number of boxes to hold all the items and the number of items in each box are as small as possible. The two scenarios look different. However, if we look at the two scenarios carefully, we will see both are about the same problem. In the first scenario, the problem is that each item has different size so that it needs enough space to hold it. If the space in knapsack is not big enough, the items cannot be put into the knapsack. In the second scenario, we have a big space and then divide the entire space into several small boxes, so, the space capacity is not critical anymore. What we need to do is to put each one of the items into different boxes which is easy to implement. Since this condition can be satisfied easily, the challenge in the second scenario is different, however, if we tweak the problem in the second scenario and each item have different sizes instead of the same size, the two scenarios become comparable. They are about the same problem, in which the size of each item is completely random in the range from 1 to n. In the second scenario, we need to fill up multiple knapsacks but the restriction to pick items and to fill up each one of the boxes is the same. In the two scenarios, there are several characteristics in common. First, the space capacity is limited. Second, we need to include as much stuff as possible.

The mathematical description of bin packing problem can be simplified as: given a positive space capacity C, a list of items $L= (p_1, p_2, \ldots, p_n)$, we need the number of bins m, such that $L = B_1 \cup B_2 \cup \ldots \cup B_m$, $sum(s(p_i)) < C$ and m is minimized.

The applications of bin packing problem can be derived into several categories, based on the variety of terminology and the variety of parameters. Based on terminology, we can interpret bin packing problem as a storage problem. In computer memory, we have limited space and need to load unlimited data for computation. We don't need to load all the data into memory at once which is also impossible. But, each time, we need to load as much data as possible and the total data size cannot exceed the size of the memory and the data size in each block cannot exceed the size of the block either. There is another example with different terminology. To have a long cable with a certain length, we need to connect small pieces of cables together: each segment cannot be longer than C and the total length of the segments cannot be longer than L.

These examples are only related to three parameters: the size of the entire space, the maximum size of segments in the space, and a list of items. Based on parameters, we can interpret bin packing problem as truck packing problem, in which we need to consider, the size of the items, the weight of the items, the maximum size of each unit, the size of the truck. The condition becomes that the maximum weight in each unit cannot exceed the upper limit w, the maximum size in each unit cannot exceed the upper limit of s, the total weight in the truck cannot exceed the upper limit W, the total size of the items cannot exceed the size of the truck S.

Another example is the commercial in station breaks on television. We need to assign commercials to each station break. The maximum length of commercials cannot exceed the length of the station break, the minimum income of commercials cannot be less than the budget of the station break, the maximum cost of the commercials cannot exceed the budget of the station break. In these examples, we have various parameters, such as total capacity, unit capacity, the maximum weight in each unit, the maximum capacity in each unit, the minimum capacity in each unit. These parameters are independent.

When we discuss the upper and lower bounds of the parameters, we can split the problem into several separate problems according to the different parameters and solve the upper and lower bounds of each parameter separately.

Scheduling problem is also a bin packing problem with the execution time as a parameter. In computer task scheduling, we need to load tasks into memory and execute the tasks. The conditions are that the maximum size of the tasks in each memory unit cannot exceed the size of the memory unit, the maximum execution time of each memory unit cannot exceed the execution limit of each unit, the total size of the units can be minimized. The execution time can be different for different operations, such as search, read, write, calculation and so on. In data processing, we have similar applications. We normally take a snapshot of the data which is also called data window. When the data window moves one slot forward, we update the snapshot. In a snapshot, we divide the space into several segments, and, in each segment, the data set size cannot exceed the size of the segment, the execution time in each unit cannot exceed the execution limit of each unit. The execution can be insertion, deletion, update, data join, search, and so on. This problem cannot be easily converted to parallel processing. Because the execution time in each unit has upper and lower bound, when multiple processors are applied, the finishing time can be within a range and the size of the range are determined by the upper bound and lower bound of the execution time. If tasks are scheduled very well, parallel computing can be efficient. The challenge is how to decide the upper bound and lower bound of the execution time in each unit.

Streaming data processing can be modeled as a bin packing problem in which we consider the size of the data window is the total space capacity. When the data stream is flowing through the data window, we take a snapshot of the data stream and update the data window we cached. Sliding window is a general solution for data stream processing. Although sliding window can provide a snapshot of the data stream, it cannot ensure the data processing result can be concise and windowed stream can be stored efficiently. The challenge is still how to arrange data items efficiently so that we can load as many data items into the space as possible and finish data processing as quick as possible. The way we handle the data model is that we divide the total space into equal units and, in each unit, the total data size cannot exceed the size of the unit and the total execution time cannot exceed the time limit for each unit so that the total time cost of all of the units is less than the speed of the window movement. For operation execution time cost, we need to consider several data operations, such as data join, data insertion, data deletion, data search, data copy and so on.

A good trade-off between the size of the space and the accuracy of the results is required when dealing with a data stream. For some aggregation operations, the more data we use, the more accurate the result is, such as maximum, minimum, sum, count and so on. For some operations, such as data join, select, group by and so on, if and only if the sample set is complete, the query result can be complete. However, given a certain time interval and memory space, we don't have random access to the entire data stream, neither store the entire streaming data set to the memory, we output partial results.

4.2 Space Management

We use I to indicate the set of items. N(I) is the number of items in set I, SIZE(I) the total size of all items in I, OPT(I) the total number of bins used in an optimal solution for set I.

Packing Properties are listed below. In property 1 and 2, the total number of bins used by the list of vectors $<I_{space}, I_{duration}>$ is greater than the total number of bins used by I_{space}, or $I_{duration}$, and less than the total number of bins used by $(1+\varepsilon)$-approximate of I_{space} and $(1+\varepsilon)$-approximate of $I_{duration}$. In property 3, the number of items in I, I_{space}, and $I_{duration}$ are the same. Because we basically use the same set of items with different variables: one indicates the space cost of the item set, the other indicates the time cost of processing the item set. Bin-packing of the vector set is still NP-Hard. We use greedy algorithm to pack items: we manually pick a starting point and develop the approximate solution. For the item set I, space capacity and processing time of each job are two independent variables so that two sets of items in I_{space} and $I_{duration}$ are independent.

1. $OPT(I_{space}) < OPT(<I_{space}, I_{duration}>) < (1+\varepsilon)OPT(I_{space})$
2. $OPT(I_{duration}) < OPT(<I_{space}, I_{duration}>) < (1+\varepsilon)OPT(I_{duration})$
3. $SIZE(I) = SIZE(I_{space}) = SIZE(I_{duration})$
4. Bin-packing of space capacity and duration is NP-Hard.
5. I_{space} and $I_{duration}$ are independent.

4.3 Space Boundary Estimation

Theorem 1. For $k \in \{1, 2, \dots, n\}$, and a batch I contains $<I_1, I_2, \dots, I_k>$, the decision problem k-BP(I, C, k+1) on input (I, C) is NP-Complete.

Proof.

1. Prove L is NP

Given a set $L(<I_1, I_2, \dots, I_k>)$, we can verify in polynomial time that all of the n items are packed into m bins, so that Bin-packing items $L<I_1, I_2, \dots, I_k>$)is a NP problem.

2. Select a known NP-Complete language L’

Bin-packing items L’(I) is NP-Complete.

3. Describe an algorithm that computes a function f mapping every instance of L’ to an instance f(x) of L

Bin-packing items $L<I_1, I_2, \dots, I_k>$)< Bin-packing items I , so that we can define a function f mapping every instance of Bin-packing items $L(<I_1, I_2, \dots, I_k>)$ to Bin-packing items L’(I).

4. Prove that the function f satisfies $x \in L'$ if and only if $f(x) \in L$ for all $x < \{0,1\}$.

If and only if the set I_{space} and the set $I_{duration}$ can be packed into L, items in set $I_1, I_2, \dots, I_k$ can be packed.

5. Prove that the algorithm computing f runs in polynomial time.

Bin-packing the set $I_{\text{~~space~~}}$ and I_{duration} can be done in between O(NlogN) and O(N). In other words, Bin-packing L(<I_1, I_2 , …, I_k>) can be done in polynomial time. ■

Theorem 2. Given a batch I, there is a deterministic algorithm that pack input into bins satisfying P_1 – P_5, with $\sigma = O(NlogN)$ for sorted batch I and $\sigma = O(N)$ for unsorted batch I.

Proof.

Bin-packing stream I can be done with a complete scan of the list of bins for each one of the items. The time complexity of the algorithm is $O(N)$ for unsorted array, and $O(NlogN)$ for sorted batch I with binary search. ■

Theorem 3. For any ε, there is no comparison-based data structure for item matching which stores $O(\varepsilon logN), 0 < \varepsilon < 1$ on any given input stream of length N.

Proof.

The space complexity to search one item is between $O(logN)$ and $O(N)$, logN < s < N. So, comparison-based data structure for item matching cannot be done in $O(\varepsilon logN), 0 < \varepsilon < 1$.■

Theorem 4. Given a batch I, we can derive a set of <I_1, I_2 , …, I_k> that satisfies P_1 – P_5, and the solution S of <I_1, I_2 , …, I_k> uses at least $O(logN)$ bins and at most OPT(I) bins. Let ALG<I_1, I_2 , …, I_k> be the number of bins that our algorithm outputs. Then, it holds that $\mathrm{O(logN)} \leq ALG(I) \leq (1 + \varepsilon)\mathrm{OPT(I)}$

Proof.

O(logN) < min(OPT(I_1), OPT(I_2), …, OPT(I_k)) < OPT(<I_1, I_2 , …, I_k>) < max(OPT(I_1), OPT(I_2), …, OPT(I_k)) < OPT(I). In other words, O(logN) < OPT(<I_1, I_2 , …, I_k>) < OPT(I). For the number of bins our algorithm outputs ALG(I), ALG(I) < OPT(I) < $(1 + \varepsilon)\mathrm{OPT(I)}$, and also, ALG(I) >OPT(I) > O(logN). In other words, $\mathrm{O(logN)} \leq ALG(I) \leq (1 + \varepsilon)\mathrm{OPT(I)}$■

Theorem 5. For multi-variable Bin-packing, given a stream I, we can derive a set of <I_1, I_2, … , I_k> and use at least max(SIZE(I_1), SIZE(I_2), SIZE(I_3), …, SIZE(I_k)) memory and at least max(OPT(I_1), OPT(I_2), … , OPT(I_k)) bins.

Proof.

For each bin occupying memory space m with items <a_1, a_2, …, a_k>, m > $\max_{1<i<k}$(SIZE(a_i)) so that SIZE(<I_1, I_2, … , I_m>)≥max(SIZE(I_1), SIZE(I_2), … , SIZE(I_k)).

For each bin of capacity c with items <a_1, a_2, …, a_k>, m > $\max_{1<i<k}$(OPT(a_i)) so that OPT(<I_1, I_2, … , I_m>)≥max(OPT(I_1), OPT(I_2), … , OPT(I_k)). ■

Theorem 6. If $a_1 - a_n \leq \tilde{C}$ in which $a_1 > a_2 > \ldots > a_n$, then, after bin packing, the capacity of the largest box C_{max} and the smallest box C_{min} satisfy $C_{max} - C_{min} \leq \tilde{C}$

Proof.

At step 1, $C_{max}^{(1)}$ = max(a_i, 0), $C_{min}^{(1)}$ = min(a_i, 0). $C_{max}^{(1)} - C_{min}^{(1)} = a_i \leq \tilde{C}$

Suppose at step k, $C_{max}^{(k)} - C_{min}^{(k)} \leq \tilde{C}$

At step k+1, $C_{max}^{(k+1)}$ = max($C_{min}^{(k)} + a_i$, $C_{max}^{(k)}$), $C_{min}^{(k+1)}$ = min($C_{min}^{(k)} + a_i$, $C_{max}^{(k)}$).

$C_{max}^{(k+1)} - C_{min}^{(k+1)} \leq C_{max}^{(k)} - C_{min}^{(k+1)}$. Also, since $C_{min}^{(k+1)} \leq C_{min}^{(k)}$, $C_{max}^{(k)} - C_{min}^{(k+1)} \leq C_{max}^{(k)} - C_{min}^{(k)}$. In other words, $C_{max}^{(k+1)} - C_{min}^{(k+1)} \leq C_{max}^{(k)} - C_{min}^{(k)} \leq \tilde{C}$. ■

Theorem 7. Give a multiset A = {I_1, I_2, …, I_n} of positive integers with total sum S, the problem of deciding whether there exists a subset I ⊂ A of sum $\frac{S}{m}$ is NP-Complete.

Proof.

1. Prove L is NP. Given a set L(<I_1, I_2 , …, I_k>), we can sum the Euclidian distance between each vector [a_1, a_2, …, a_k] and the bin ceiling vector [b_1, b_2, …, b_k] and verify in polynomial time that all of the n items are packed into m bins, so that Bin-packing items L<I_1, I_2 , …,

I_k>) is a NP problem.

2. Select a known NP-Complete L’

   Bin-packing problem L’(I) into m bins is NP-Complete.

3. Describe an algorithm that computes a function f mapping every instance of L’ to an instance f(x) of L.

   Bin-packing items L<I_1, I_2 , …, I_k>) < Bin-packing I, so that we can define a function f mapping every instance of Bin-packing items L(<I_1, I_2 , …, I_k>) to Bin-packing items L’(I).

4. Prove that the function f satisfies x ∈L’ if and only if f(x) ∈ L for all x < $\{0,1\}^*$

   If and only if the set L(I_1, I_2 , …, I_k) can be packed into m bins, each one of the items in set I_1, I_2 , …, I_k can be packed into m bins.

5. Prove that the algorithm computing f runs in polynomial time.

   Bin-packing the set I_1, I_2 , …, I_k can be done in between $O(NlogN)$ and $O(N)$. In other words, bin packing L(<I_1, I_2 , …, I_k>) can be done in polynomial time. ■

Theorem 8. Given a multiset A = {I_1, I_2, …, I_n}, OPT(I) < ALG(I) < OPT(I) logOPT(I)

Proof.

O(logN) < min(OPT(I_1), OPT(I_2), …, OPT(I_k))

< OPT(<I_1, I_2 , …, I_k>) < max(OPT(I_1), OPT(I_2), …, OPT(I_k)) < OPT(I). In other words, O(logN) < OPT(<I_1, I_2 , …, I_k>) < OPT(I).

For the number of bins our algorithm outputs ALG(I), OPT(I) < ALG(I) < $(1+\varepsilon)$OPT(I), and also, $(1+\varepsilon)$OPT(I) $< OPT(I)\log\big(OPT(I)\big), since\ O\big(OPT(I)\big) < O(OPT(I)\log(OPT(I))$, In other words, OPT(I) $\leq ALG(I) \leq OPT(I)\log(OPT(I)$ ■

Chapter 5: Graph Sampling

5.1 Introduction

The connectedness nature of the network links distributed entities together and expand the size of the connections to different forms of infinity from entire population to entire supply chain and entire work fields. For the purpose of data analysis, it highlights the importance of graph sampling. We need smaller representatives of the entire network which can fit into main memory easily and maintains all the graph properties and topology types of the original graphs. We proposed two methodologies to sample graphs: one is based on the self-similar nature of the graph; the other is based on the curvature of the graph.

Based on the self-similar nature, graphs can be generated through edges, as shown in [40], and subgraphs, as shown in [148]. We want to apply these ideas to graph sampling that, given a graph, instead of dropping subgraphs to data sets we define a recursive process to consistently select subgraphs from original graph and put them into sample sets. The prime graphs can be used to select subgraphs and to decompose the original graphs, such as R-MAT and Kronecker graphs. In comparison with graph generators, sampling process is based on original graphs so that the change of the graph properties and graph topology types can be evaluated during graph generation.

There are many different types of networks. In daily life, people communicate with each other through phone calls, emails, what’s app messages, Facebook, Twitter, LinkedIn, and other social media platforms. In business, because of materials, energy, labor fees, and market sizes, companies open branches in different locations, and cooperate with other businesses. In science, cross-edge projects and knowledge sharing can be seen everywhere. Scientists from different areas and different organizations build co-authorship, cite each other's papers, cite each other's patents, and also propose new ideas together. In other areas, such as film making, power grid, biology, and Internet, networks play the central role in activities, such as co-actors, electricity delivery, protein-protein interactions, and cyber security.

We are interested in extracting valuable patterns, trends, and knowledge from networks. For example, in citation network, we want to know which research is popular and is highly related to each other, and what are the trends of the development of the technology. In online news network, we want to know which topics people care about the most. In co-purchase network, we want to know how many products are highly related, and what are the trends of the purchase.

Network research has been conducted for many years, especially in the fields of physics and mathematics. The structure of the network is represented by graphs in data structure, and by adjacent matrix in mathematics.

For finite graphs and graph morphisms, the connectedness can be preserved during projection and join operations and be recovered under pullbacks. This property can ensure that graph merge and graph product can preserve the connectedness of the graph. In comparison with randomly sampling edges and vertices, it is much easier to maintain the graph properties and graph topologies by sampling through graph merging.

In Geometry, curvature is the force felt by observers moving along the curve. Curves evolve in two directions: one is tangent direction; the other is normal direction. Curvature is along the normal direction which can change the direction of the movement and eventually change the shape of the curves. During curve evolution, any free shape curves can evolve into circles. We use curvature model to simulate the dynamic of the community growth and to sample social graphs based on their topology.

Influence physically exists among people. We keep learning from the environment and other people. Human signals can be delivered during activity, communication, and any human interaction. The strength of the influence can be defined with several properties: mimicry, consistency, and communication. When we mimic other people's behaviors, we can be accepted easily. Most of time, when we deeply understand and accept other people, we can mimic other people very well. Consistency plays an important role when we try to convert other people with our opinions, behaviors, and styles. On the other hand, if we keep making changes when we try to connect to other people to directions, we can never build connections. Over time, people can understand, remember, and accept what we try to propose, such as opinions, behaviors, and styles. Communication is the key in relationship. A successful communication can draw people together. Resistance exists among people as well. People naturally want to keep a distance between each other for respect and for protection which was researched in sociology about one hundred years ago. In social network, we only consider the influence among people which strengthen the social ties and contribute to the construction of the communities.

Social analysis is based on connections and interactions among people to extract significant patterns and to interpret dynamics and trends of communities. For data entities, we are interested in significant individuals, significant connections, and significant subgroups. For insights, we are looking for social trends and social dynamics. The challenge normally comes from the size of the network which makes it impossible to conduct analysis and from the measurements which may not properly model the characteristics of the interactions within communities.

The speed of the community growth depends on the influence of the community, which is also well-known as rich get richer. Based on observations given in [150], the structure of the community shrinks and densifies over time. The bigger the community, the harder it is to attach to.

The stronger the influence of the community center, the more members it can attract. With the increase of the level of the community, the strength of its influence to the bottom level decreases exponentially. The influence of the community center is strong but cannot be as strong as that of their adjacent neighbors, which explain some members of the big communities can be separated from the community and attach to other communities. The influence of the community plays two roles: one is to draw their members closer and closer, the other one is to tear apart the connections of the members with other communities. A member can have connections with several different communities. Overtime, which connections can be kept, depends on the influence of the communities.

In this dissertation, we discuss related work in section 2, section 3 introduces related theory for interesting layer construction and network decomposition. In section 4, we discuss the construction of interesting layers. In section 5, we present experiments on several data sets. We conclude our work in section 6.

5.2 Related Work

5.2.1 Graph Theories and Operations

For all the graphs which are finite, no loops or multiple edges, the operations we need for graph sampling include graph reconstruction, graph decomposition, X-graph join, and Kronecker double covers. These operations have been generally discussed.

Indecomposable and decomposable graphs can be differentiated through conditions and can convert from one kind to another by removing a few edges [232].

In [107], graph automorphisms, connectedness and partition of joined graphs, especially X-join graphs, were provided. Graph X-join operation can be performed by replacing each x of X by graph Y_x. Conditions that G(X*Y) consists precisely of those automorphisms induced from

G(X) can indicate the topology of X-join graphs. In other words, when we apply graph X-join to symmetric graphs, graph and topology properties of jointed graphs can be verified.

Graph double cover is a graph projection operation given in [Waller, D. A. 1976]. Graph double can merge both the two vertices which results in merging edges. At this point, this operation is Similar to Cartesian product of two graphs on the two vertices instead of one. In terms of graph mapping, the difference between automorphism and double covers is that automorphism is one to one mapping, and double cover morphism is two to one mapping. Graph double cover can reduce the graph size, which is a two-fold projection onto G and preserves local structure as well. For graph sampling, in order to keep local structure, we choose to use graph double cover to repeatedly fold similar edges and vertices instead of removing them so that topology properties can be kept during graph sampling.

When graphs are categorical graphs, the local projection of graphs [70] can be performed though graph product, n-fold cover and other graph operations. Functional operations can perform graph merging which maps vertices to the same vertices, and edge generation which maps vertices to different vertices. Especially, X-join graph as a special case of local join is discussed in terms of graph properties, category graph properties, and the complexity of the X-join graphs.

Graph properties can be changed when graph operations are performed. The change can be quantitatively measured with additive combinatorics and extreme graph theory [Zhao. Y. 2019]. For example, the maximum number of edges, distinct distances, triangles, and other quantities of the graphs can be estimated when we conduct graph operations, such as edge and subgraph removal, and randomized and algebraic construction. We are especially interested in upper bounds and lower bounds of graph sizes, such as the maximum number of edges, during graph operations to ensure that the space complexity of the operations is within the upper limit of main memory.

The development of a purely metric, intrinsic geometry of networks can help the independent study of curvature flow without the vagaries associated to the embedding in an ambient space of certain dimension. Given the lack of smoothness of the graph structure, hence the full force of the differential geometric apparatus, in the network context, each discretization of curvature captures only an essential aspect of the classical notion, and as such, it can satisfy only a restricted set of the properties of the classical notion. Therefore, it is desirable to devise different notions of discrete curvature and explore their specific advantages for various task.

Ricci curvature and Forman curvature are theoretically developed in discrete differential geometry to provide geometric approach to explore curvature evolution. In [214], Haantjes-Ricci curvature, Menger-Ricci curvature were provided to formulate the curvature evolution on vertices and edges. Haantjes curvature has a clear advantage over Menger curvature as it is applicable to graphs without any assumption on the background geometry. Forman curvature for complex network [229] was developed in the new way to apply the discretization of the classical Ricci curvature proposed by R. Forman [74] in large-scale networks.

5.2.2 Graph Generators

Graphs can be generated recursively with individual graph patterns, such as R-MAT [40], and Kronecker Graphs [148]. This gives rise to the question if we can define a recursive process to sample graphs by factorizing the graph with prime graphs to find the meaningful portion of the graphs, meanwhile topology types and graph properties remain.

R-MAT [40] simulates the generation of social networks. Adjacency matrix represents the social network which can be partitioned into four regions: region a and region d contain separate groups of nodes that correspond to communities, region b and region c are the cross-links between region a and region b, that represent cross-link relations, such as friends of separate interests. The way it creates the sample set is that it randomly drops edges into different regions. It has simple structure and can capture the essence of the graphs in several parameters. R-MAT can be used to generate undirected graphs, directed graphs, and bipartite graphs.

Although R-MAT algorithm divided the adjacency matrix into four regions, when it recursively looks for the exact location of the connection, it is not quantitatively defined how to quantitatively find the location of edges in different regions. In the four regions, the distribution and correlation among edges are related to graph properties and topology properties of the graph. Based on topology types, in each region, vertices and edges can be divided into two categories: core and periphery so that, the properties of these vertices and edges follow different distributions. Given the definition of the four regions, for example, in friendship network, we can arrange friends of the same or different interests into the four regions, but we must find a way to define which ones can be arranged to the core and the periphery set, and which ones cannot be determined by randomly drop edges into different regions.

R-MAT generator [6] can cause combing problems that graph distribution can be significantly changed at regular geometric intervals. For example, the distribution of certain degrees can be very different from the distribution of the entire graph. Also, this change happens in regular basis and is a weakness of the R-MAT generator. R-MAT generator defines the meanings of the edges. However, it also needs to combine topology types into simulation process.

Like R-MAT, graph sampling can be done by randomly selecting edges. With no regard to regions, the methodology in [2] randomly select edges from graphs to make a seed set. After edge selection, connections between vertices in the seed set can be added. When we test this algorithm, we found that it requires a large seed set to generate large samples. When the seed set is large, there are only a few additional connections between vertices in the seed set.

Kronecker product [148] is based on the symmetric structure which can help produce self-similar graph for any size, obey the main static patterns and the temporal evolution patterns, and match the combination of properties. For the purpose of graph sampling, the symmetric nature of the Kronecker graphs can ensure that Kronecker graph pattern can be recursively used to join and project graphs. Also, this process can be done recursively, and the connectedness of the graphs can be preserved. Symmetric structure of the Kronecker graph simplifies graph operations. Kronecker graph structure defines a similar pattern and is also the smallest similar graph pattern which can be used to define prime graphs as the graph computation unit.

Kronecker product can also be used to manipulate graphs, such as graph factorization and expansion [26] [27]. Graph factorization is based on user/item incidence matrices and apply the Kronecker graph to reduce the dimensions of the graphs. Kronecker product can help expand data sets better than product setting in user base size, item vocabulary size, number of observations and other sample statistics, and the high order statistical properties are also preserved. One limitation of Kronecker product is that it creates similar graphs through block-wise low rank matrix product for graph expansions. But it will not affect graph factorization.

5.2.3 Graph Sampling

Snowball sampling [89] is based on vertices. Sampling process selects first k individual vertices. After that, each individual vertex reaches out to k more individual vertices. This process is repeated in s stages. When s = 0, snowball sampling becomes Bernoulli sampling in which we choose a particular portion from the population. However, it causes the variance of the estimation of mutual relationships (with individuals in the population are either in the sample or in the first stage) at least twice as large as the variance of the estimation with Binomial sampling, especially when the characteristic of the population is unknown. For networked data, multiple stage sampling is a better option. For our graph decomposition and reconstruction sampling, we adopt multiple stage sampling methodology, especially set the order of merging graph from low frequency to higher up, so that the difference between the sample sets and the original data can be observed.

Graph factorization can be used to reduce the size of the sample set. Sample quality and data clustering can be two separate steps. We are looking for a better way to systematically pick significant vertices and edges so that we can shrink the sample set size and reduce the workload for data clustering. Graph factorization can decompose graphs into smaller ones and the smaller graphs can be merged through projection if they are similar. When we keep factorizing graphs and projecting common factors, large networks can be converted into smaller ones with a few connections and a few vertices, which makes graph computation possible. In terms of data sampling, when we eliminate some of the data, the topology properties of the graphs need to be kept the same.

Graph decomposition can be conducted on each individual vertices to form low-dimensional embeddings [98]. Embeddings have both the vertex and neighborhood information. Both graph properties and topology types can be carried in neighborhood information. This methodology can convert global graph into local subgraphs and this process is reversible. In other words, the information loss is low. When the computation is based on local subgraphs, it requires low space complexity that makes it possible for most of workstations to perform.

Community Guided Attachment (CGA) model as shown in Figure 5.1: the bigger the community size, the harder to attach to it. We use a graph model in Figure 5.1 to model these observations. h(v, w) is the height of the least common ancestor. f(h) is difficulty function of h(v, w), $\mathbf{f(h) = c^{-h}}$**.** f is level independent. C is the difficulty constant $1<c<b$. D = log(b, n) = constant. D is the expected average out-degree of a node.

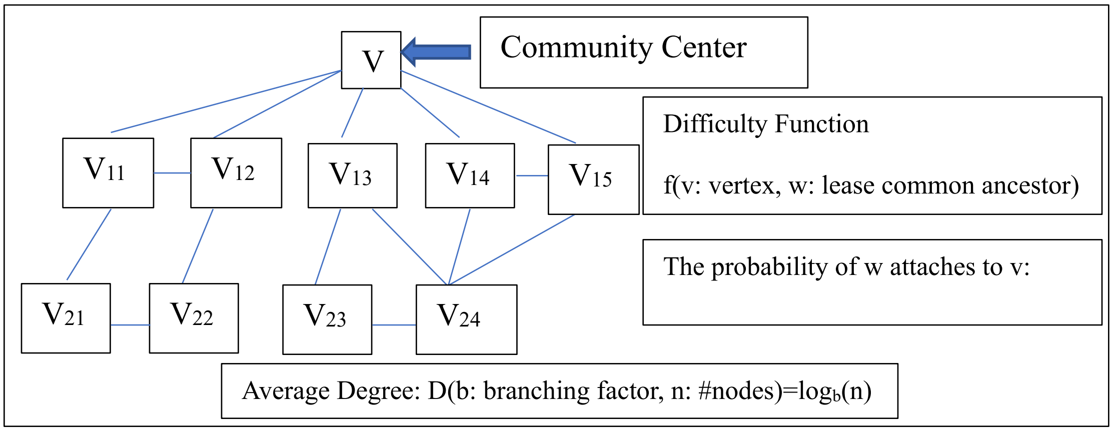

V
Community Center
V_{11}
V_{12}
V_{13}
V_{14}
V_{15}
Difficulty Function
f(v: vertex, w: lease common ancestor)
V_{21}
V_{22}
V_{23}
V_{24}
The probability of w attaches to v:
Average Degree: D(b: branching factor, n: #nodes)=$\log_b(n)$

Figure 5.1. A Community Influence Strength for Different Layers.

In each step t, the tree grows from t-1 to t by adding b new leaves to the current leaf as children leaves. New leaves can be new node of the graph. New nodes can also connect to internal nodes with probability $\mathbf{c}^{-d(v, w)/2}$

CGA model takes into consideration the association between the hierarchical structure and the probability of attaching new nodes. But the structure between the new node and existing nodes is not indicated. The difficulty constant C and the branching factor b are not quantitatively explained.

Forest fire model exhibits both densification and diameter decrease. Nodes with high out degree can serve as the bridge of the disparate parts in the network. Nodes arrive one at a time, start with node v and then attach a node w uniformly at random in the graph. Forest fire model starts burning links from w with forward burning probability p and backward burning ratio r, where v is the bridge.

Two random numbers x=p/(1-p) and y=rp/(1-rp). Vertex v selects x out-links and y in-links of w, $(w_1,w_2, \ldots, w_{(x+y)})$. Vertex v forms out-links to w1, w2, …, w(x+y), starts with w $(w_1,w_2, \ldots, w_{(x+y)})$ and then repeats this process.

Forest fire model can build a sample set based on the original topology. However, only local structure of the connections is recorded. The hierarchical structures of the communities are ignored. Also, the vertices with high degrees can be promoted. The forward burning probability p and the backward burning ratio r are neither proportional to the total number of edges nor the total number of nodes. Since we start with vertex V, vertex V is supposed to be the bridge of the two communities. If vertex w only has a few links, V cannot have many links either. In that case, V cannot be the bridge of two communities. The addition of vertex V does not show any specialty. In other words, we cannot oversee through forest fire model how many significant nodes and how many communities can be generated in the results.

The strength of the forest fire algorithm is that the way forest fire model grows the graph is based on breadth first search, which can keep the topology of the graph.

The degree distribution of the original set and the sample set needs to belong to the same family of probability distributions [230]. Based on the degree distribution, the probability that a node of degree i in the full net connecting to k ($k<i$) other nodes can be estimated with binomial distribution. The probability generating function is based on the degree distribution and can be used to uniformly build subnets.

5.3 Graph Theories

When we conduct graph sampling, graph topology types need to be kept and the graph properties cannot be changed. For this research, we reduce the sample set size through graph projection.

We illustrate symbols that are used through this chapter. V(X) is the vertex set of the graph X, and E(X) is the edge set of the graph X. Let vw be an edge {v,w} and $v \sim_X w$ be the adjacency of v and w vertices in graph X.

Centrality measures are generally used to evaluate how similar the vertices are. However, some measurements are based on the search of the entire graph, such as, betweenness centrality, eigenvector centrality, closeness centrality and so on. For computational efficiency, we intend to use local measurements to evaluate the similarity of the vertices instead of the global measurements. For this purpose, we derived several local measurements to evaluate the similarity of the vertices in each Kronecker graph.

Pattern Frequency measure the frequency of similar pairs measures if the similar pairs can represent the characteristics of the graphs. Let the frequency of the similar pairs $X \sim Poisson(\lambda)$

$$P(X = x|\lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!}, x = 0,1,2,\dots \qquad (5.1)$$

in which λ is the average frequency.

Density within a Hop is when we not only care about the frequency of the similar patterns but also care about the connectivity of the similar patterns. Hop plot studies the size of the neighborhood around the similar patterns. It is a local measurement which can be computed by loading a small portion of the network.

$$D = \frac{<k>}{E} \qquad (5.2)$$

in which <k> is the number of common neighbors of the two end vertices of the Kronecker graphs, E is the number of edges in the subgraph.

Effective Diameter Distribution measures distances between two vertices in a subgraph.

$$\delta = \frac{N^2}{N+2E} \qquad (5.3)$$

For a complete graph, E is equal to N(N-1) and then δ is equal to N/(2N-1). For Ring-lattice network, E is equal to $\binom{N}{2}$, δ is equal to N/(1+N-1)=1. For other topology types, after removing isolated vertices, δ is between N/(2N-1) and 1.

Automorphism is that, given a graph x, a permutation α of V(X) is an automorphism of X if for all $\mu, \nu \in V(X)$

$$\{\mu,\nu\} \in E(X) \Leftrightarrow \{\alpha(\mu), \alpha(\nu)\} \in E(X) \qquad (5.4)$$

With the composition function, automorphisms of a graph X can form the *automorphism group* of X, and it is denoted Aut(X). Aut(X) is a subset of the symmetric group on V(X).

From the definition of graph automorphism, we can derive these facts in automorphism groups, that, let the components of X be $X_1, \ldots, X_k$ then

$$Aut(X) = \prod_{i=1}^{k} Aut(X_i) \qquad (5.5)$$

Also, the following shows edge-complement $\overline{X}$ for a graph X, we have

$$Aut(X) = Aut(\overline{X}) \qquad (5.6)$$

A group X acts transitively or is transitive on the permutation set S of X if, for every x, y ∈ S, there exists $\alpha \in X$ such that $\alpha(x) = y$ is vertex-transitive. $Aut(X)$ is *transitive* on V(X), and is doubly transitive on S if ,for any two ordered pairs of distinct elements (x_1, x_2), (y_1, y_2) ∈ S*S, there exists $\alpha \in G$ such that $\alpha(x_1) = y_1$ and $\alpha(x_2) = y_2$.

A *graphon* (graph function) is symmetric measurable function W: $[0,1]^2$ -> [0,1]

A *graph homomorphism* from H to G is a map $\varphi: V(H) \rightarrow V(G)$ such that if $\mu\nu \in E(H)$.

Given graphs H and G, let $\hom(H, G)$ be the set of all such homomorphisms and let $\hom(H, G) = |\hom\,(H, G)|$. Define *homomorphism density* as

$$t(H,G) = \frac{\hom\,(H,G)}{|V(G)|^{|V(H)|}} \qquad (5.7)$$

It also defines the probability of the homomorphism for a given uniformly random map.

X and Y are two sets of vertices in a graph G and $e_G(X,Y)$ denotes the number of edges between X and Y; that is

$$e_G(X,Y) = |\{(X,Y) \in (X \times Y | xy \in E(G)\}| \qquad (5.8)$$

We can use the definition of $e_G(X,Y)$ to define the *edge density* between X and Y to be

$$d_G(X,Y) = \frac{e_G(X,Y)}{|X||Y|} \qquad (5.9)$$

Given a graph X, let the collection of its vertex-deleted subgraphs X-v for all v∈V(X) be the *deck* of X, denoted by D(X).

Convergent is that, given graphs H and X, the sequence converges to X if t(H, X_n) converges to t(H, X) for every graph H.

Existence of Limit is that, for graphs or graphons, every convergent sequence has a limit graphon.

Equivalence of Convergence is that sequence of graphs or graphons is convergent if and only if it is a Cauchy sequence with respect to the cut (distance) metric. (A Cauchy sequence with respect to metric d is a sequence $\{x_i\}$ that satisfies $sup_{m\geq 0} d(x_n, x_{n+m}) \rightarrow 0 \; as \; n \rightarrow \infty$).

In general, given graphs X and Y, a *graph product* is a graph that the vertex set is defined as V(X)*V(Y), and the edge set is defined by (and only by) the adjacency relations in both X and Y. The symbol & denotes a generic graph product X&Y for given graphs X and Y.

Kronecker product $X_1 \wedge X_2$ of the graphs X_1 and X_2, as shown in Figure 5.2, has vertex set $V(X_1 \wedge X_2) = V(X_1) \wedge V(X_2)$ with adjacency in $X_1 \wedge X_2$, given by $v_1 v_2 {\sim} w_1 w_2$ if and only if $v_1 \sim_{x_1} v_2$ and $v_2 \sim_{x_2} w_2$. It can be represented as

$$[\{x_1, x_2\} \in E(X) \wedge y_1 = y_2] or [x_1 = x_2 \wedge \{y_1, y_2\} \\ \in E(Y)] \; or \; [\{x_1, x_2\} \in E(X) \wedge \{y_1, y_2\} \in E(Y)]$$

Given graph X, Kronecker double cover $\tilde{X}$ has vertices $(v,1)$ and $(v,2)$ for each vertex v in X, with adjacency $v \sim_x$ w, if and only if $(v,1) \sim_x (w,2)$ and $(v,2) \sim_x (w,1)$ are in $\tilde{X}$. The equation for the vertex is in (5.10) and the equation for on edges is in (5.11).

$$\left.\begin{matrix}(v,1)\\(v,2)\end{matrix}\right\} \Rightarrow v \qquad (5.10)$$

$$\left.\begin{matrix}(v,1)(w,2)\\(v,2)(w,1)\end{matrix}\right\} \Rightarrow (v,w) \quad (5.11)$$

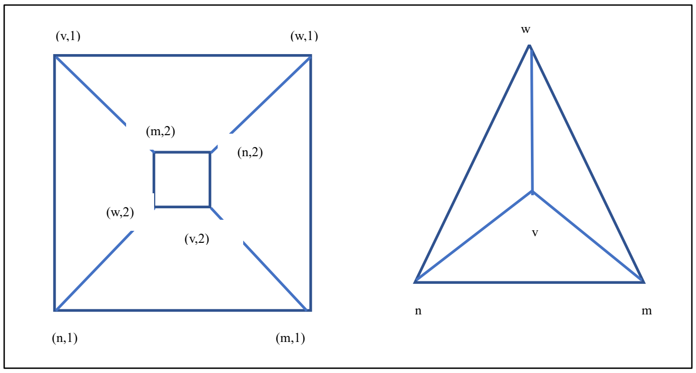


Figure 5.2. Kronecker Double Cover

5.3.1 Graph Decomposition and Reconstruction

Graphs are self-similar in terms of the structure of the graphs. Graphs can be formed by keep adding vertices and edges. Prime graphs as subsets of the graphs can be considered as the basic patterns on the graphs. It can carry more information and more characteristics of the graphs and can be specific to particular graphs. We use Kronecker graph as the primary graph to factorize graphs, as shown in Figure 5.3. There are some advantages in choosing Kronecker graph as the prime graph. For example, it is the smallest symmetric graph with no loops, and the symmetric nature of the Kronecker graph can make it possible for the graphs to expand and to shrink to any sizes through Kronecker operations.

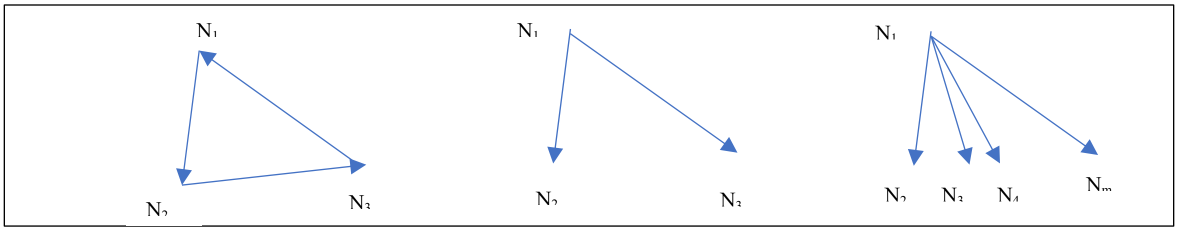
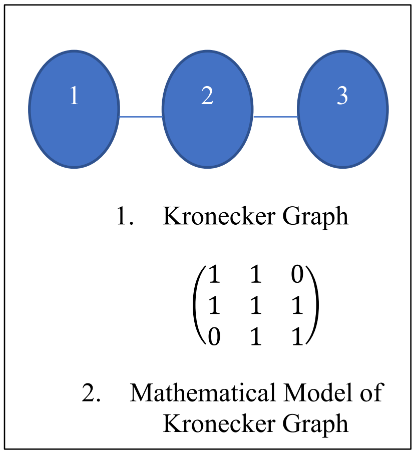


Figure 5.3. Kronecker Graphs



1. Kronecker Graph

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

2. Mathematical Model of Kronecker Graph

Figure 5.4. Prime Graphs in Social Network

5.3.2 Graph Factorization

Graph decomposition can be based on different prime graphs. For friendship network, graphs can be decomposed into a set of triangles, as shown in Figure 5.4(a). In each triangle, a pair of friends connect to each other and some of their friends are common friends for both. When a circle of three friends is defined, an interesting pattern of similar entities is formed. In the same way, we can also apply graph decomposition to other graph networks. For citation network, graphs can be decomposed into a set of Kronecker graphs which consists of one paper with two reference papers, as shown in Figure 5.4(b). For co-purchase network, graphs can be decomposed into a set of items, which consists of at least three items, as shown in Figure 5.4(c). To define similar vertices with neighbors, we can use as many of the common neighbors as possible in order to increase the strength of the similarity, if needed. Also, it is easier to compare adjacent neighbors than adjacent graphs.

To make it possible for graphs to decompose and to reconstruct, it requires that the graphs to be reconstructible, the homomorphism density can converge as n goes to infinity, and the upper and lower bounds of the number of the prime graphs needed for graph construction can be estimated. To satisfy these criterions, we list the following theorems.

Lemma 1. Every graph with at least three vertices is reconstructible [96].

Lemma 2. Every graph on at least four edges is edge-reconstructible [96].

Lemma 1 and Lemma 2 put restrictions on the lower bounds of both the vertices and the edges for graphs reconstruction. Other than the lower bounds of both the vertices and the edges, the upper bounds of both the vertices and the edges also need to be determined so that we can estimate if, after factorization, the size of the subgraphs can be enough to load into main memory. We prove the upper bounds in Theorem 4. Other than the upper/lower bounds of both edges and vertices, we are also interested in the number of subgraphs.

Lemma 3. Given a graph G-v in the deck of G, the degree of v and the degrees of the neighbors of v in G are reconstructible [96].

Lemma 4. Suppose G and F are graphs with |V(F)|<|V(G)|. Then

$$(|V(G)| - |V(F)|)\binom{G}{F} = \sum_{v \in V(G)} \binom{G-v}{F} \qquad (5.12)$$

Therefore, $\binom{G}{F}$ is reconstructible. [96]

In other words, given graphs G and F, the total number of subgraphs of G choosing F can be computed with equation 5.12. After quantitatively evaluating the upper bounds and the lower bounds of the space complexity, we want to show how to reconstruct subgraphs.

Let D(G) be the *deck* of G that can be formed through removing vertices from the graph G. Meanwhile, when vertices are removed from the graph, connected edges can also be removed, if any.

In other words, D(G) consists of subgraphs of G. The subgraph of G can also be generated through graph factorization based on prime graphs. The results of graph factorization can be the subset of the deck D(G).

Lemma 5. Let G be a graph without isolated vertices. The deck of G D(G) is edge-reconstructible, that is D(G) is uniquely determined from $\varepsilon D(G)$. Therefore, if G is reconstructible, then it is also edge-reconstructible [96].

Based on Lemma 5, we know graph factorization with prime graphs can be conducted because the deck of G is edge-reconstructible so that the subset of the deck of G is reconstructible which is the factorization results we are looking for.

Lemma 6. Let A be a n*n symmetric matrix with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \quad (5.13)$$

and matrix B can be derived from A by removing i-th row and i-th column from A, and if matrix B has eigenvalues

$$\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n \quad (5.14)$$

Then the eigenvalues of B interlace those of A, that is, [96]

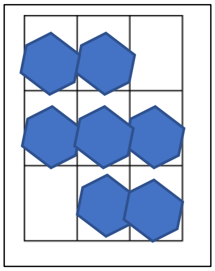
$$\lambda_i \geq \mu \geq \lambda_{i+1} \qquad (5.15)$$

Based on Lemma 6, after matrix reconstruction, the order of the eigenvalues stay the same.

As shown in Figure 5.5(a), given a Kronecker graph G1 and the number of G1 n, when n goes to infinity, the graphon W_{Gn}: $[0,1]^n$ -> [0,1] converges to a function which looks like Figure 5.5(c).

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

G_1	G_1	0
G_1	G_1	G_1
0	G_1	G_1



(a) Adjacent Matrix of G1 (b) Adjacent Matrix of G1*G1 (c) Graphon of G, as n→ ∞

Figure 5.5. A Kronecker Graph G1 and the Limit of W_{Gn}.

5.3.3 Clustering Coefficient

The clustering coefficient of a vertex u on unweighted graphs is the fraction of possible triangles through that vertex, that exist,

$$c_u = \frac{2T(u)}{\deg(u)(\deg(u)-1)} \quad (5.16)$$

in which T(u) is the count of triangles through the vertex u and deg(u) is degree of the vertex u.

For weighted graphs, let clustering coefficient be the geometric average of the edge weights on subgraphs,

$$c_u = \frac{1}{\deg(u)(\deg(u)-1)} \sum_{vw} (\widehat{w_{uv}} \widehat{w_{uw}} \widehat{w_{vw}})^{1/3} \quad (5.17)$$

The edge weights $\widehat{w_{uv}}$ are normalized through total edge weights over the maximum weight in the network $\widehat{w_{uv}} = w_{uv}/\max(w)$.

For directed graphs, let clustering coefficient be a fraction of all possible directed triangles or geometric average of the subgraph edge weights, respectively. This definition can be applied to both unweighted and weighted directed graphs.

$$c_u = \frac{T(u)}{deg^{tot}(u)(deg^{tot}(u)-1)-2deg^{\leftrightarrow}(u)} \quad (5.18)$$

where T(u) is the total number of directed triangles through vertex u, $deg^{tot}(u)$ is the total of both in and out degrees of vertex u and $deg^{\leftrightarrow}(u)$ is the reciprocal degree of vertex u.

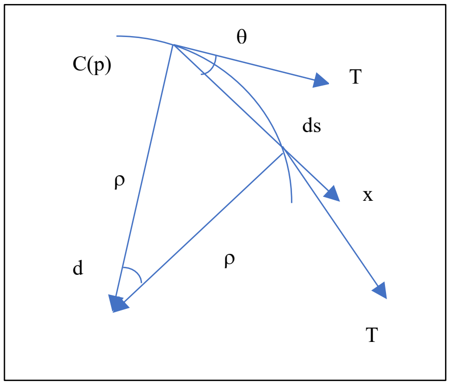


Figure 5.6. The Tangent T(p) and the Normal $\vec{N}(p)$ along the Parametric Curve C(p)

5.3.4 Differential Geometry in a Plane

Let the curve C(p) = {x(p), y(p)}, where p∈[0,1]. P is the parameter of the *curve* where for every value of p between 0 and 1, C(p) is the coordinate of one point along the curve.

Formally, we say the curve C maps the interval I=[0,1] to the Euclidean plane R^2 and write C: $I \rightarrow R^2$.

Unit Tangent is that we let T be the first derivative C_p of the curve C with respect to its parameter p, or the velocity.

The second derivative C_{pp} of the curve C is the derivative of the velocity according to the parameter p. The unit tangent is obtained by normalization

$$\vec{T} = \frac{T}{|T|} = \frac{\{x_p, y_p\}}{\sqrt{x_p^2 + y_p^2}} \quad (5.19)$$

Unit Normal is that, given a vector in the plane, the normal to this vector is obtained by changing the positions of its arguments and inverting the sign of one of them. The *unit normal* is obtained by normalization.

$$\vec{N} = \frac{\{-y_p, x_p\}}{\sqrt{x_p^2 + y_p^2}} \quad (5.20)$$

The inner product between the tangent and the normal is 0.

$$< T, \vec{N} >= x_p \frac{-y_p}{|C_p|} + y_p \frac{x_p}{|C_p|} = 0 \qquad (5.21)$$

Curvature is that the length from 0 to p can be measured by

$$s(p) = \int_0^p \left|C_{\tilde{p}}(\tilde{p})\right| d\tilde{p} = \int_0^p \sqrt{x^2{}_{\tilde{p}}(\tilde{p}) + y^2{}_{\tilde{p}}(\tilde{p})}\, d\tilde{p} \quad (5.22)$$

The first derivative C_s of the curve C is

$$C_s = C_p \partial_p p = C_p / |C_p| \qquad (5.23)$$

The second derivative C_{ss} of the curve C is

$$C_{ss} = \frac{(C_p, C_{pp})}{|C_p|^3} \vec{N} \qquad (5.24)$$

in which the quantity k=$\frac{(C_p, C_{pp})}{|C_p|^3}$ is called curvature. K =$\theta_s = \frac{d\theta}{ds} = 1/\rho$, where ρ is called the curvature radius.

5.3.5 Differential Geometry in a Network

Forman curvature for an edge is given in the following formula:

$$F(e) = w_e\left(\frac{w_{v_1}}{w_e} + \frac{w_{v_2}}{w_e} - \sum_{e_{v_1} \sim e, e_{v_2} \sim e}\left[\frac{w_{v_1}}{\sqrt{w_e w_{e_{v_1}}}} + \frac{w_{v_2}}{\sqrt{w_e w_{e_{v_2}}}}\right]\right) \quad (5.25)$$

in which e is the edge under consideration between vertex v_1 and vertex v_2, w_e is the weight of the edge e, w_{v1} and w_{v2} are the weights of vertex v_1 and vertex v_2, e_{v1}~e and e_{v2}~e are the set of edges incident on nodes v_1 and v_2 after excluding the edge e.

Forman curvature on vertices is

$$F(v) = \frac{1}{\deg(v)} \sum_{e_v \sim e} F(e_v) \quad (5.26)$$

in which e_v denotes the set of edges on the vertex v and deg(v) is the degree of the vertex v.

Menger-Ricci curvature of an edge is given by

$$K_{M,O}(e) = Ric_{M,O}(e) = \sum_{T_e \sim e} K_{M,O}(T_e) \quad (5.27)$$

$$K_{M,O}(T) = \varepsilon(T) * K_M(T) \quad (5.28)$$

where $T_e \sim e$ denotes the triangles connected to the edge e, $\varepsilon(T) \in \{-1,0,1\}$, $K_M(T)$ can be Euclidean, spherical, or hyperbolic version and the Menger-Scalar curvature of a vertex is given by

$$K_{M,O}(v) = scal_{M,O}(v) = \sum_{e_k \sim v} Ric_{M,O}(e_k) = \sum_{T \sim v} K_{M,O}(T) \quad (5.29)$$

where $e_k \sim v$ and $T \sim v$ denotes that vertex v connect to all the edges e_k and all the triangles T as a vertex.

Haantjes-Ricci curvature of an edge is given by

$$K_{H,O}(e) = Ric_{H,O}(e) = \sum_{\pi \sim e} K_{H,O}(\pi) \quad (5.30)$$

$$K_{H,O}(\pi) = \varepsilon(\pi) * K_H(\pi) \quad (5.31)$$

where p~e denotes the paths connect the vertices anchoring the edge e, $\varepsilon(\pi) \in \{-1,0,1\}$, $K_H(T)$ can be Euclidean, spherical, or hyperbolic version and the Menger-Scalar curvature of a vertex is given by

$$K_{H,O}(v) = scal_{H,O}(v) = \sum_{e_k \sim v} Ric_{H,O}(e_k) = \sum_{T \sim v} K_{M,O}(T) \quad (5.32)$$

where $e_k \sim v$ stands for all the edges e_k adjacent to the vertex.

Haantsjes-Ricci curvature for an edge in undirected and unweighted simplicial complexes can be obtained by counting the triangles T containing an edge e, that is

$$Ric_H(e) = \#\{T | T > e\} \quad (5.33)$$

5.4 Graph Sampling through Kronecker Graph Merging

Data sampling is the first step in data analysis. Through data sampling, we can not only shrink the size of the data set but also improve data quality. Network data set is different from the data set of independent samples. We need to control the quality of both vertices and edges which correspond to entities and entity relationship.

At this point, there are more data properties to check in networked data set than in independent samples. After adding connections between data points, all the data entries are organized in the form of different topology types, such as ring lattice, small world, Erdos random, core-periphery, scale free, and cellular network.

Data sampling needs to keep both the distribution of vertices and edges, and the original topology of the data set, which can be verified through topology properties, such as the distribution of degrees, degree frequencies, degree ranks, effective diameters, eigenvalues, and other centrality measures. For a data set with independent samples, we only need to think about the sample distribution which is equivalent to the distribution of vertices.

The data property we normally use in data sampling is the frequency of the data entries. Different from the data set with independent samples, network sampling needs to consider both the frequency and the stability of the samples.

First, selected items must be frequent. For example, normally, random vertices and edges are not frequent and cannot be used to analyze the characteristics of the network. Secondly, since connections play an important role in social network, the stability of the connectedness between vertices is also required in sample selection. On the other hand, random connections are not stable and cannot be used to represent the characteristics of the network. Other than the local properties of vertices and edges, we also care about the global properties of the networks, which is presented in network topology.

Except Erdos-Renyi random graphs, most of graphs can be divided into two regions based on the degree distribution: one is core region which has high degree vertices, and the other is periphery region which has low degree vertices. For the number of vertices, the different between the two regions is in an order of magnitude so that the vertex hierarchy is from core to periphery vertices. In social networks, the core region contains people with strong influences and the periphery region contains people with weak influences.

The idea of top-down sampling is that, if we group core and corresponding periphery into one region, which is equal to slice the graph from core to periphery, we can have many different regions each one of which represents the characteristics of the graph. When the sample set is large enough, a portion of the regions are enough to represent the graph and the rest of the regions can be ignored.

The idea of the bottom-up sampling is that, because of the huge difference in the number of vertices in core and periphery, if there are redundant samples, more redundant samples are in periphery and less redundant samples are in core. It can effectively shrink the sample size if we remove redundant samples from periphery.

Meanwhile, both top-down and bottom-up sampling remove some vertices and some edges, which gives rise to a question that, when the regions, no matter they are the entire slice of the hierarchy or the bottom layers of the periphery, are cut off from the graph, the methodology we choose to deal with the connectedness on the cut need to be in favor of maintaining the characteristics of the graph.

For vertices and edges on the borders of the regions, we define similarity among these vertices and edges and merge similar vertices and edges. Sampling process can be affected by two factors: one is the time complexity, and the other is the merging methodology. The time complexity of the sampling process depends on similarity comparison. Because, given vertices, the number of edges can go exponential so that it can be impossible to conduct the similarity comparison. The merging methodology can have big impacts to the topology properties of the graph. We provide proofs and experiments on complexities of graph merging methodology in the following sections.

5.4.1 Graph Merging

For different topology types, both the meanings and the significance of vertices and edges are different. We can divide most of graphs into two regions: core region (which can also be called bridge group for scale free topology, small world topology, and cellular topology), and periphery region (which can also be called ordinary group for small world topology, scale free topology, and cellular topology). When we conduct graph sampling through decomposition and reconstruction, graph properties and topology types need to be maintained. This can be illustrated in the following.

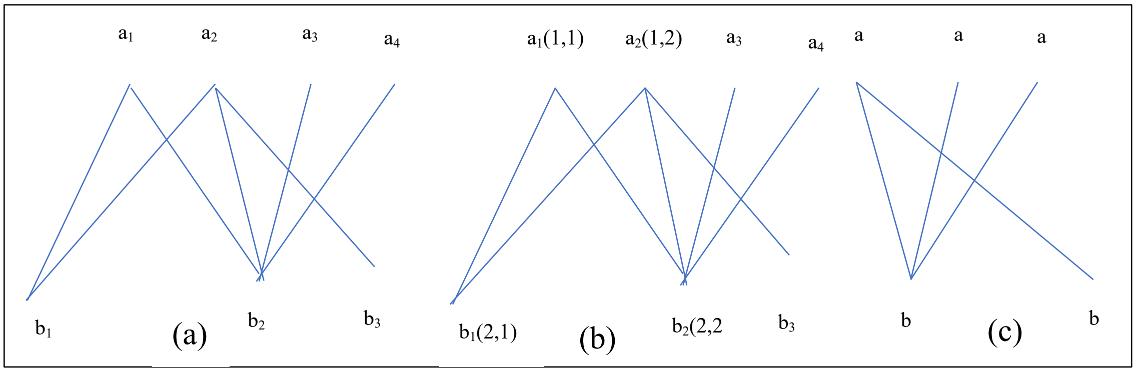


Figure 5.7. Graph Merge through Kronecker Double Cover. (a) a_1 and a_2 are similar because graphs $a_1b_1b_2$ and $a_2b_1b_2$ have two vertices, b_1 and b_2, in common. Vertices b_1 and b_2 are similar because they appear together in more than one graph (graph $a_1b_1b_2$ and graph $a_2b_1b_2$) (b) a_1b_1 and a_2b_1 are merged. (c) a_2b_1 and a_2b_2 are merged.

```
Algorithm 1. Kronecker Double Cover

Input: G: original Graph, V(G): vertex set of graph G, E(G): edge set of graph G, T: Cutoff Threshold, sim: Similarity Measure
Output: G': sample graph
    1. Periphery_set P = $\{(v,e)|\ v \in V(G), e \in E(G), degree(v) < T\}$
    2. Core_set C = G – P
    3. Kronecker set K = $\{(v, w_1, w_2)|\ v \in V(P), w_1 \in V(P), w_2 \in V(P), (v, w_1) \in E(P), (v, w_2) \in E(P)\}$
    4. Kronecker similar groups S = $\{(count, w_1, w_2)|count = len([v_s])\ for\ ([v_s], w_1, w_2)\ in\ K\}$
    5. For sim = S.count.min() to S.count.max()
    6. $Frequent\ set\ F = \{(v, w_1, w_2)|(v, w_1, w_2) \in K, count(w_1, w_2) \geq sim\}$
    7. Merged_set M=$\{(v,w)|if\ ([v_p], w_i, w_j) \in F, v\ replaces[v_p]$, w replaces $w_i\ and\ w_j\}$
    8. V' =$\{v|v \in V(M + C)\}$
    9. E' =$\{e|e \in E(M + C)\}$
    10. V = V'
    11. E = E'
```

Figure 5.8. Kronecker Double Cover Algorithm

As shown in Figure 5.7, Kronecker double cover methodology is used to merge edges. In Figure 5.7 (a), a_1 and a_2 have two neighbors b_1 and b_2 in common so that a_1 and a_2 are similar. b_1 and b_2 are in graphs, $a_1b_1a_2$ and $a_2a_1b_2$, respectively, and both b_1 and b_2 in graph $a_1b_1b_2$ and $a_2b_1b_2$, so that b_1 and b_2 are similar. New names (1,1), (1,2), (2,1), and (2,2) can be assigned to a_1, a_2, b_1, b_2. (1,1) and (1,2) can be merged, and (2,1) and (2,2) can be merged, as shown in Figure 5.7 (c).

On line 1, Periphery set P can be generated by selecting all the vertices with degrees less than T and also adding the edges between these vertices. On line 2, Core set C has the vertices and edges in the graph after excluding Periphery set P. On line 3, Kronecker graphs K can be generated by selecting two neighbors (w_1 and w_2) of a vertex v. On line 4, the graph frequencies of the Kronecker graphs with two neighbors (w_1 and w_2) in common are counted and the results are saved to the frequent set S(count, w_1, w_2). Kronecker similar groups are created based on the frequency of the isomorphic graphs. We assume that vertices and edges in periphery region are end vertices and edges. If they are by chance connected, those connections are considered as random connections. From line 5 to line 11, we conduct Kronecker double cover operation to merge edges. On line 6, we select Kronecker graphs with frequency greater than *sim* and save the results to the frequent set F. On line 7, Kronecker double cover operation can be performed to generate merge set M. On line 8 through line 11, we update Kronecker graph, vertex and edge sets with new names in the merge set M.

According to Theorem 1, vertices with at least two neighbors can be called constructible, otherwise, they are called *residuals*. After merging, some vertices become residuals, but they are not removed from the graph. Although Kronecker double cover sampling can ensure that the connectedness of the graph is not changed, however, when these vertices become residuals, they can be less significant.

5.4.2 Complexity of Kronecker Double Cover

Kronecker graphs can be denoted in matrix format. Therefore, Kronecker product and Kronecker double cover operations can be performed through matrix operations, such as matrix multiplication and factorization.

Theorem 1. Kronecker graph join has a limited graphon.

Proof. Kronecker graphs can be used to make a sequence of graphs through Kronecker graph join. Existence of Limit Theorem proves that, if a sequence of graphs or graphons are convergent, then they have a limited graphon. Therefore, Kronecker graph join can generate the sequence of graphs with a limited graphon. ■

Theorem 2. Any binary graphs can be generated from Kronecker graphs with no self-loops and multiple edges.

Proof.

We use the following equations to illustrate how to generate any binary graphs with no self-loops and multiple edges with Kronecker graphs.

Kronecker graphs can be defined with matrix M

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \qquad (5.34)$$

We want to multiply two Kronecker graphs M*M to generate a binary graph N, as shown below in 5.35. Figure 5.9 (a) shows that we can factorize graph N to form graph M. ■

$$N = M * M$$

$$= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{pmatrix} \qquad (5.35)$$

$$\begin{pmatrix} n_{11} & \cdots & n_{1i} \\ \vdots & \ddots & \vdots \\ n_{i1} & \cdots & n_{ii} \end{pmatrix} = \begin{pmatrix} M & M & \\ M & M & M \\ & M & M \end{pmatrix} = \begin{pmatrix} 1 & 1 & \\ 1 & 1 & 1 \\ & 1 & 1 \end{pmatrix} * M = M * M = M$$

(a) Mathematical Model of Kronecker Graph Factorization

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} * \ldots * \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}^{n} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

(b) Mathematical Model of Kronecker Graph Multiplication

Figure 5.9. Kronecker Graph Join and Factorization

Theorem 3. During projection, Kronecker graph properties remain.

Proof. a Kronecker graph sequence {G| G = G_i , i = 1, … , n} can be written in mathematical form, as shown below,

$$\prod_{i=1}^{n} G_i = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} * \dots * \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad (5.36)$$

As shown in Figure 5.9 (b), Kronecker graph multiplication can be used to generate Kronecker graphs. In other words, for a sequence of Kronecker graphs, Kronecker properties remain during projection. ■

To estimate the space complexity of a sequence of graphs, Homomorphism density can be useful. After decomposing Kronecker graphs, we can use Theorem 4 to estimate the space complexity, as shown in Equation 5.38. The complexities of both $\binom{G}{k_{1,2}}$ and $|V(G)|^3$ are O(n^3), that is, for Kronecker graphs, the homomorphism density is O(1).

Theorem 4. Homomorphism Density of Kronecker graph

Proof. Let a Kronecker graph $G_1 = k_{1,2}$ and let hom($k_{1,2}$, G) denotes the number of Kronecker graph G_1 in G. That is

$$\text{hom}(k_{1,2}, G) = \binom{2}{1} * \binom{1}{1} * \binom{G}{k_{1,2}} = 2 * \binom{G}{k_{1,2}} \qquad (5.37)$$

Let t($k_{1,2}$, G) denote the homomorphism density. That is

$$t(k_{1,2}, G) = \frac{\text{hom}\ (k_{1,2}, G)}{|V(G)|^{|V(k_{1,2})|}} = \frac{2 * \binom{G}{k_{1,2}}}{|V(G)|^3} \quad (5.38)$$

in which $k_{1,2}$ is a bipartite graph. This theorem can also be expanded to bipartite graphs $k_{1,n}$. ■

5.5 Graph Sampling through Curvature

When a community just starts growing, we only have the center of the community and randomly add other members, so that the influence of other members can be ignored, although, with the growth of the community, all the members gain more influence.

To keep the topology of the community, sampling process is based on edges instead of vertices. Each significant edge represents both significant vertices and the stable connections between significant vertices. For the rest of the edges, we have two options. One is we can project edges to vertices so that, the vertices can be connected if a path exists between them. The other is we can also remove the rest of the edges.

In short, if there is a path between two vertices, when the edges are removed, the connection between two vertices is cut and the two vertices become disconnected. Based on graph evolution, we want to simulate graph densification and shrinking so that vertices merge to other vertices along edges.

The sampling methodology we proposed can be illustrated in Algorithm 2. In this model, the influence of the community centers is based on the distance between the community center and the observed vertices. The influence of the community centers is taken into consideration. The simplified model is based on two assumptions:

Assumption 1 is that we consider clustering coefficient can indicate how much influence a vertex has. Clustering coefficient is the ratio of the number of triangles over the number of wedges. A triangle is the smallest community. The larger the clustering coefficients, the larger portion of neighbors are in triangles which are communities. The larger portion of the neighbors are in communities, the stronger influence the vertices have.

Forman-Ricci curvature can be represented by the total number of neighbors which the edge can have, and Haantjes-Ricci curvature can be represented by the number of triangles which the edge can form. Clustering coefficient evaluates not only how many stable connections a vertex can have, which is the number of triangles, but also the ratio between the number of stable connections over the total number of wedges, which indicates the stability of the connections, so that Clustering Coefficient can better represent the influence of the vertices.

Assumption 2 is that, at a particular moment, other than strong connections, whether the rest of the connections can be merged into strong connections randomly happen.

```
Algorithm 2.

input: p: the portion of sample set s/total set t, c: curvatures of edges, G: input graph
output: s: sample set
    1. $G_{moment}$ = {e: edges|$c_e$ > t, t is threshold at the moment and t > 0}.
    2. Find adjacent edges and add edges to sample set s with probability
       f(v, d, l)= $c(e) * p$
    3. If s.size() < t*p, decrease t and then go to step 1,
       else s = $G_{moment}$ and continue.
    4. Return s.
```

Figure 5.10. Curvature Algorithm for Community Detection

Community structure is based on the network structure which it embeds. Communities are subsets of the network. Sampling process needs to create a graph similar to the original network so that the properties of the sample set need to be similar to the original network.

5.6 Experiments

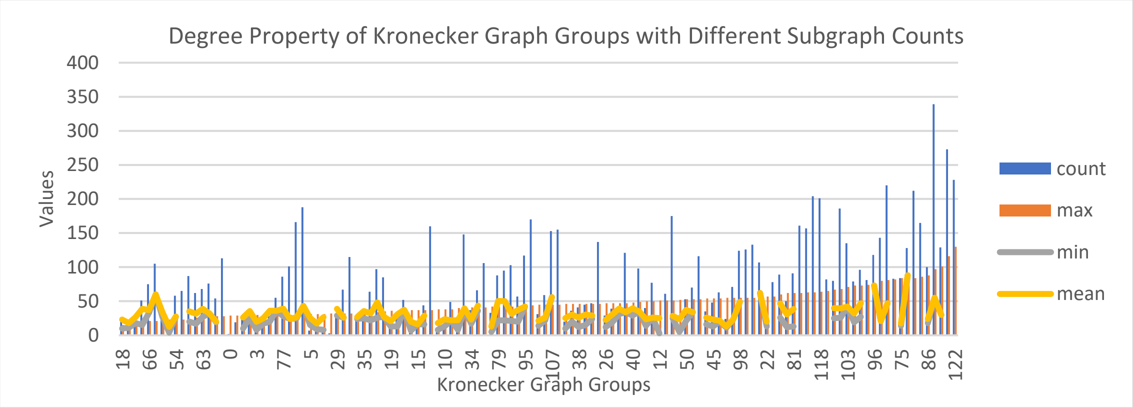
We use three data sets to prove the hypothesis: DBLP citation network, Amazon co-purchase network, Email-EU communication network. We further did preprocess on these data sets. Because we only need to merge edges with periphery vertices from the lowest degree to higher up, we separate core vertices and periphery vertices into two subsets and work on periphery vertex set only. In this way, we need to work on less vertices and edges. Degree distribution is exponential so that there is a clear gap between core and peripheral regions. We use three thresholds as cutoffs, such as the mean of the degrees (mean), the mean plus the standard deviation (mean+std), and the mean plus 2 times standard deviation (mean+2*std). Because degrees follow exponential distribution, the standard deviation can be greater than the mean so that we do not need to set the cutoff to the mean minus the standard deviation (mean-std).

For the three sample sets with cutoff equal to mean, mean+std, and mean+2*std, degree properties and Kronecker graph similarity counts are not related. In each sample set, for the core vertices group, we keep all the vertices and all the edges between core vertices and between core vertices and periphery vertices. The periphery vertices and edges between periphery vertices can be merged. In Figure 5.11, we group Kronecker graph based on similar graph frequency and visualized the similar graph counts, maximum, minimum and average degree of all the vertices in the subgroup. For the sample sets with cutoff equal to mean, mean+std, and mean+2std, the number of similar graph subgraphs are different. The sample set with cutoff mean has a smaller number of vertices and edges in periphery set so that it has less similar graph subgraphs. The sample set with cutoff equal to mean+2std has a greater number of vertices and edges in periphery set so that it has more similar graph subgraphs. In each similar graph subgraph, maximum, minimum and average degrees are random. Kronecker similarity frequency and vertex degree are two different measurements. During Kronecker graph merging, the number of edges is reduced but this operation does not directly change the degree distribution.

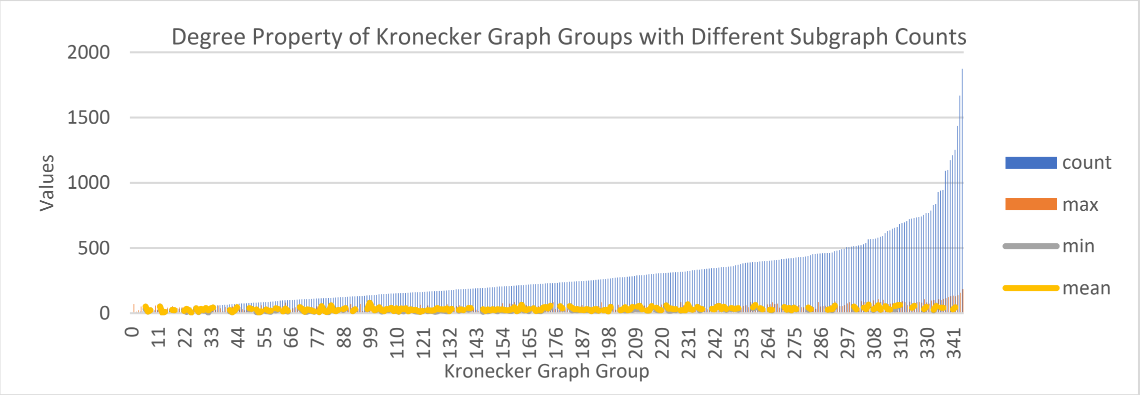
Based on graph similarity, we merge Kronecker graphs from low frequency groups to high frequency groups. This process converges within several iterations. Some high frequency groups cannot participate in graph merging because, after merging several vertices and edges, all of the

Kronecker graphs are already separated. We show in Figure 5.12 the number of vertices and edges in periphery subgraph when we merge similar Kronecker graphs from low frequency groups to high frequency groups. The number of edges and vertices are converged when graph merging stops. For sample set with cutoff equal to mean, the Kronecker similar groups are from 2 similar graph group to 339 similar graph group. The number of edges and vertices converge at 160 similar graph group. For sample set with cutoff equal to mean+std, the Kronecker similar groups are from 2 similar graph group to 1874 similar graph group. The number of edges and vertices converge at 160 similar graph group. For sample set with cutoff equal to mean+2*std, the Kronecker similar groups are from 2 similar graph group to 3461 similar graph group. The number of edges and vertices converge at 550 similar graph group.

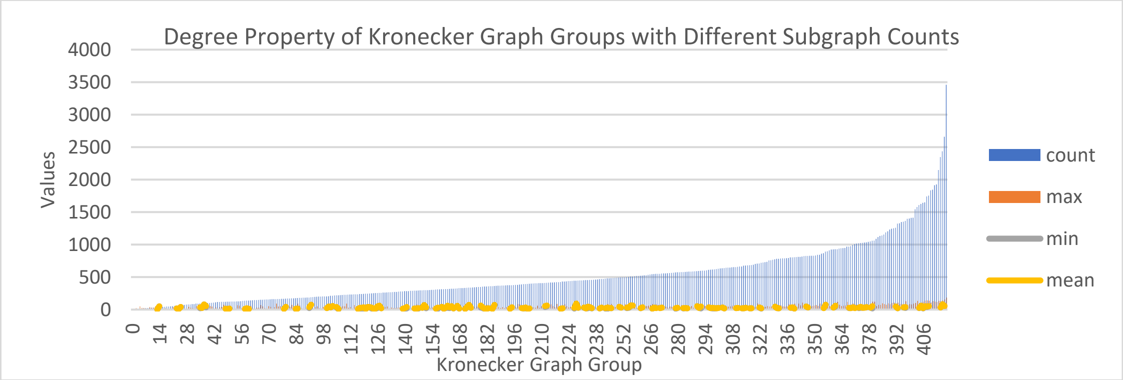
During Kronecker graph merging, the neighbors of similar vertices are merged as well. Since all the similar Kronecker graphs are merged, no two vertices share more than one neighbor, which means the connections between vertices on periphery region are merged. For topology types, Erdos-random graphs can have random connections, other than Erdos-random graphs, random connections play the role of the bridge between core region and periphery region and the connections between periphery region can be ignored. To maintain the connectedness of the graph, those connections can be merged.



(a) Sample Set with Cutoff = mean

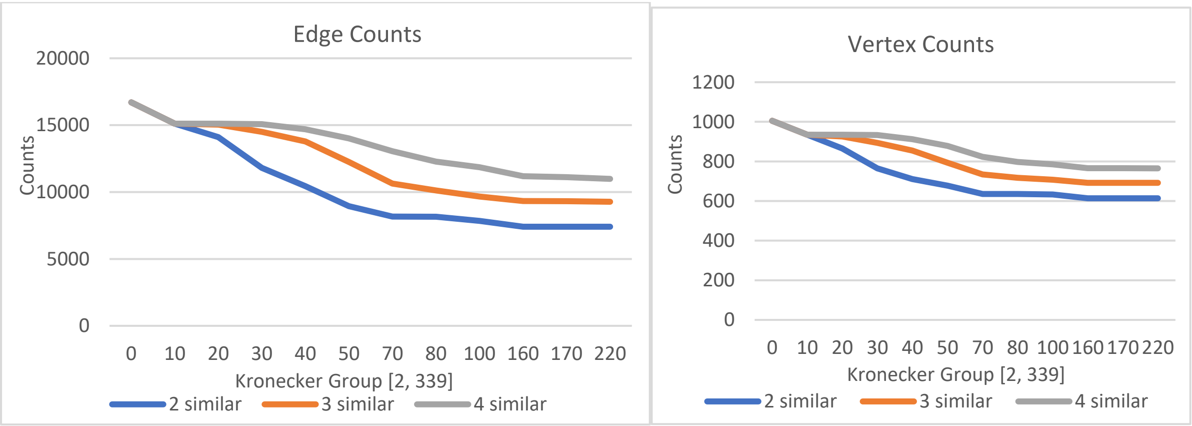


(b) Sample Set with Cutoff = mean + std

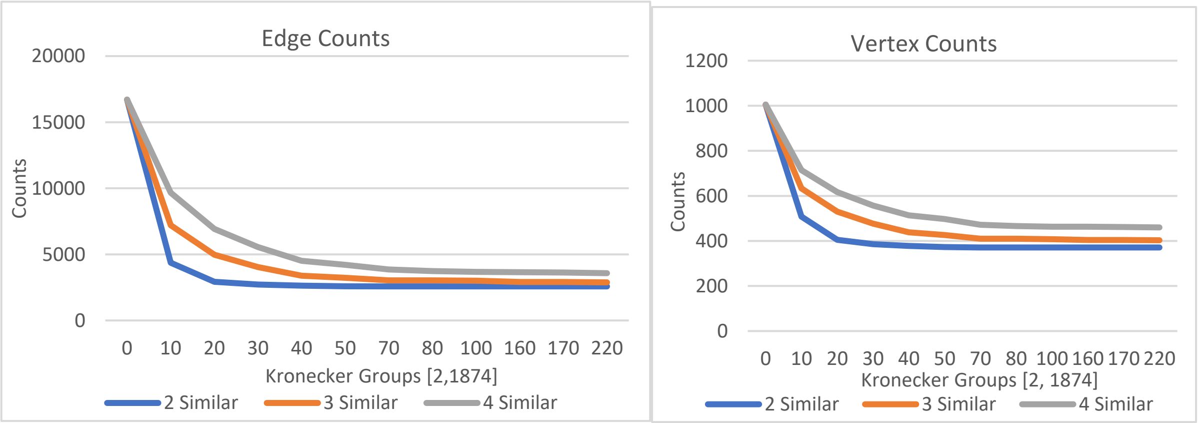


(c) Sample Set with Cutoff = mean+2*std

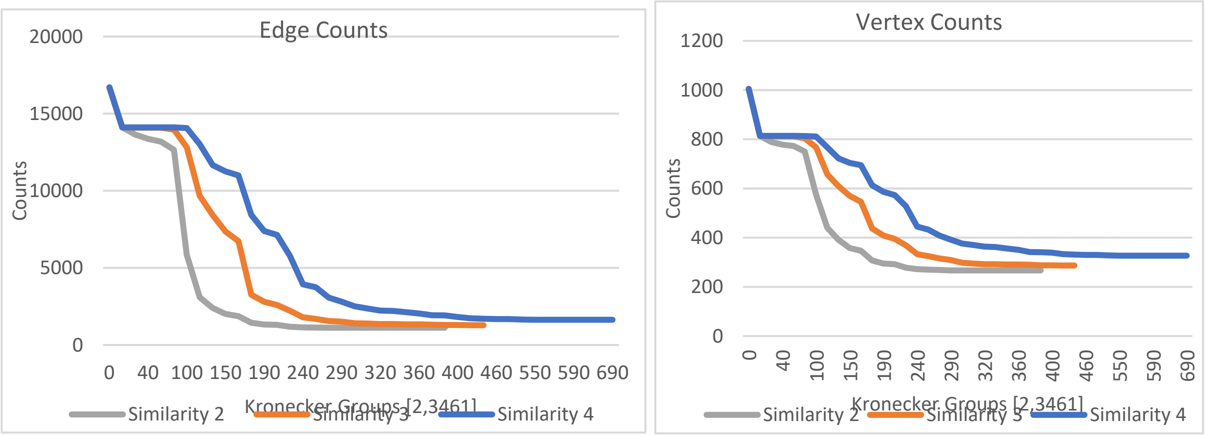
Figure 5.11. Degree Property of Different Sample Sets



(a) Edge and Vertex Counts for Sample Set with Core Periphery Threshold = mean

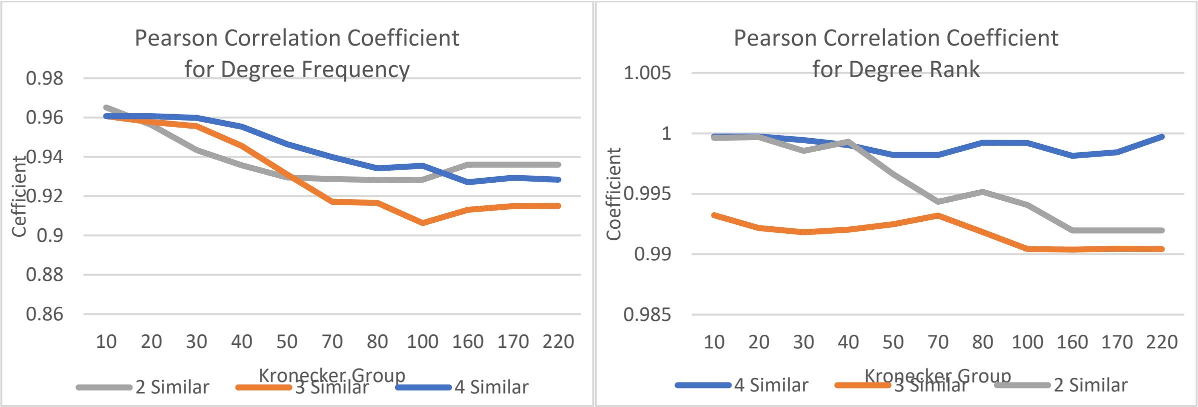


(b) Edge and Vertex Counts for Sample Set with Core Periphery Threshold = mean+std

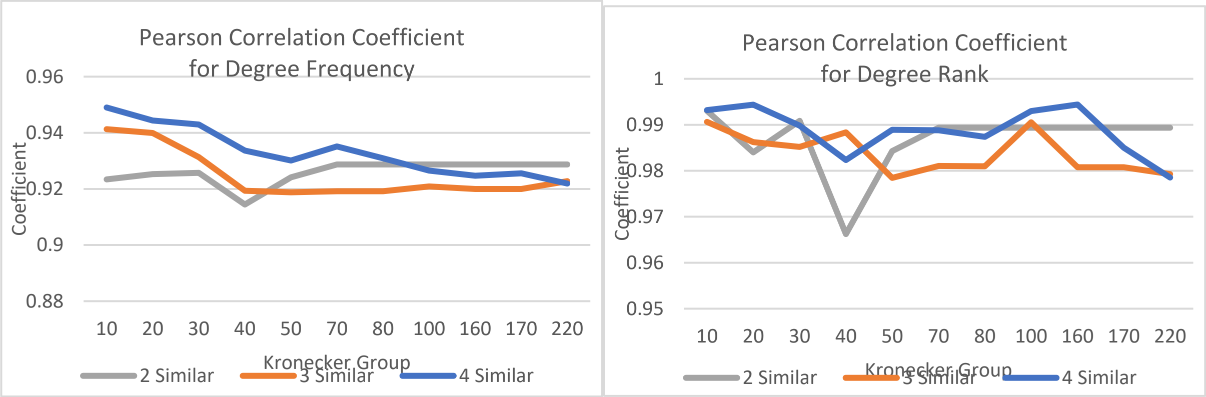


(c) Edge and Vertex Counts for Sample Set with Core Periphery Threshold = mean+2*std

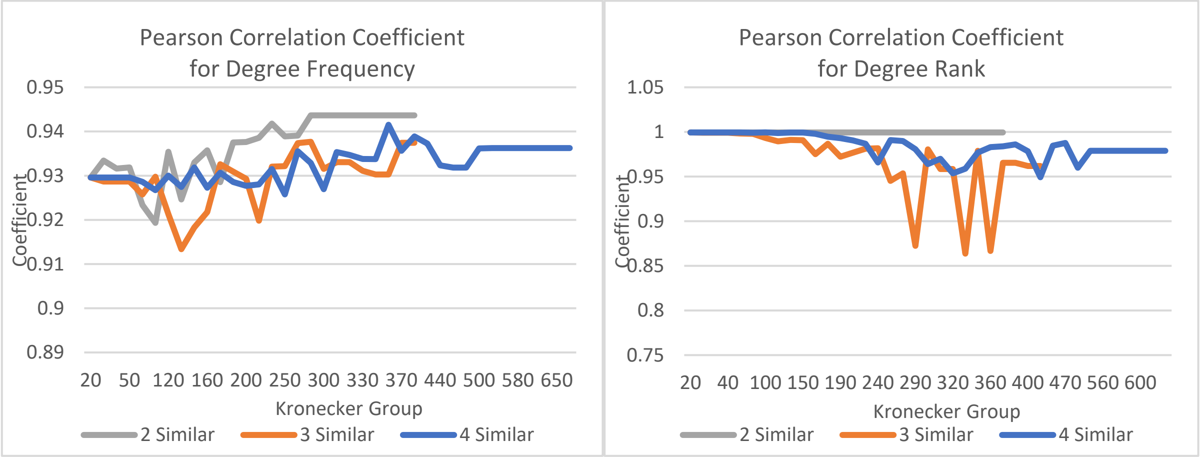
Figure 5.12. Edge and Vertex Counts During Merging Low to High Frequency Kronecker Groups



(a) Sample Set with Cutoff = mean



(b) Sample Set with Cutoff = mean + std



(c) Sample Set with Cutoff = mean+2*std

Figure 5.13. Comparison of Degree Frequency and Degree Rank Measurements between Original and Sample Sets

We use Pearson Correlation Coefficient to test the similarity of both degree frequency and degree rank between original and sample sets. The correlation coefficients for different sample sets are between 0.85 and 1. For degree frequency, when the cutoff is set to mean and mean+std, the Pearson correlation coefficient of degree frequency decrease when merging Kronecker groups from low similarity to high similarity and then converge to around 92%. For degree rank, when the similarity groups are from 0 to 220, the Pearson correlation coefficient of degree rank is within 97% to 99%. When the cutoff is set to mean and mean+std, the Pearson correlation coefficient is converged at similar group 220. When the cutoff is set to mean+2*std, the Pearson correlation coefficient is converged at similarity group 650. When the similarity groups between 220 and 650 are merged, the Pearson correlation coefficient of the degree rank does not have clear trends.

When the cutoff is set to mean, as shown in Figure 5.13(a), the trends of coefficients are consistent. When the cutoff increases from mean to mean+std and mean+2*std, the change of coefficients randomly go up and down within the range from 91% to 94%. That is because, when the cutoff is large, some core vertices are included into periphery set and can be merged during sampling. When high degree vertices are merged with the periphery vertices, the total number of degrees may increase. For periphery vertices, when vertices and edges are merged, the degree frequency and degree rank decrease. In this way, the change of the degree frequency and the degree rank can be inconsistent.

We use the generated sample sets to detect communities with Lou-vine clustering. We show the true positive rate in Figure 5.14(a) and false positive rate in Figure 5.14(b). When similarity thresholds are set between 15 and 30, the TPs are always greater than the ones of other similarity thresholds and the FPs are always less than the ones of other similarity thresholds. When similarity thresholds are between 35 and 40, the TPs are less than the ones of other similarity thresholds and the FPs are always greater than the ones of other similarity thresholds. In other words, the sample sets generated with similarity thresholds set between 15 and 30 have better quality and the sample sets generated with similarity thresholds set between 35 and 40 have worse quality.

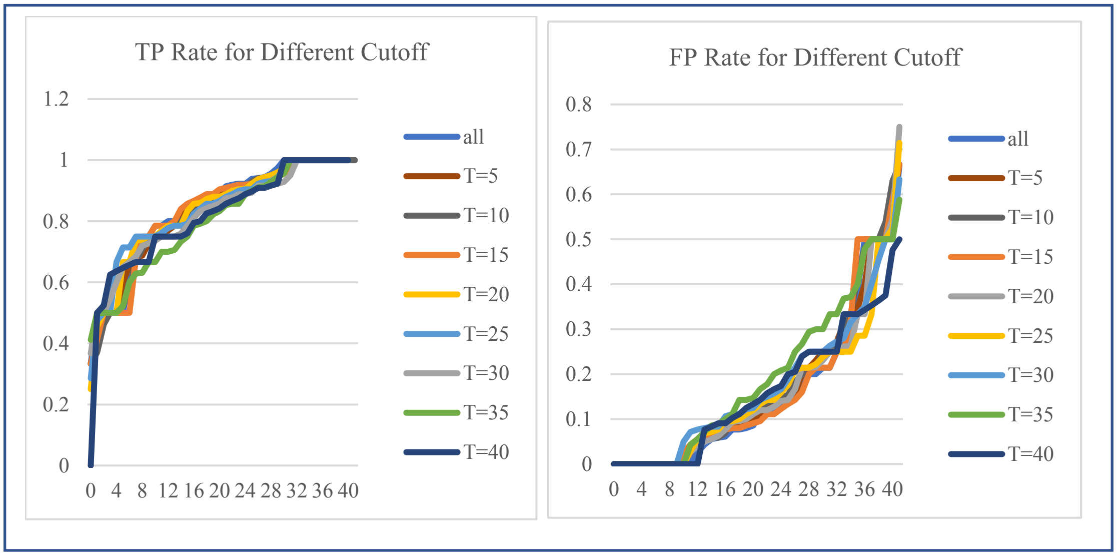


Figure 5.14. Clustering Conductance on Different Email Eu Data Sets

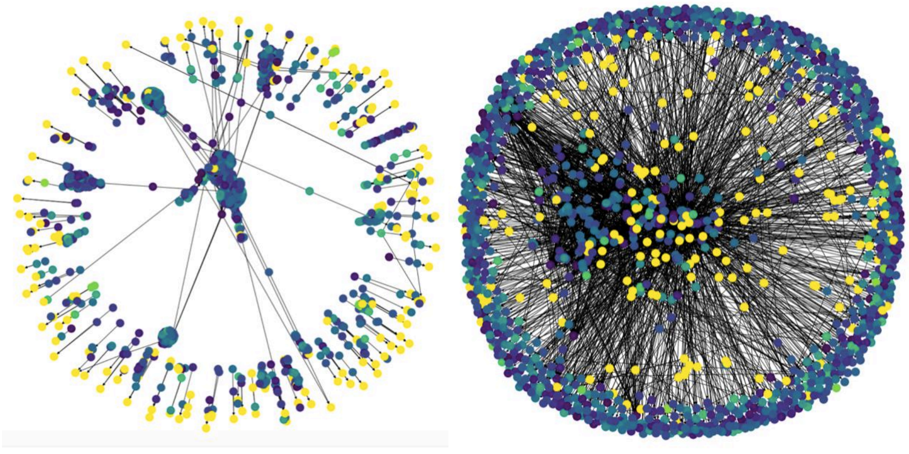
Based on the properties of the topology and the clustering conductance measurement, the sample sets generated with similarity thresholds set between 20 to 25 are like the original data set and also have good quality and the sample sets generated with similarity thresholds set between 35 and 40 are different from the original data set and also have bad quality. When the similarity thresholds are set between 5 and 20, the topology of the sample sets are different the original data set, but the sample quality increases when the similarity thresholds increase from 5 to 20.

We compare Forman-Ricci curvature, Haantjes-Ricci curvature, and Clustering Coefficient curvature methodology for graph sampling on Email-Eu data set. Clustering conductance is used to show the quality of the sample sets. In Table 5.1 are sample set sizes for all data, data after Forman-Ricci curvature sampling, data after Haantjes-Ricci curvature sampling, data after Clustering Coefficient curvature sampling.

Table 5.1. Sample Set Sizes

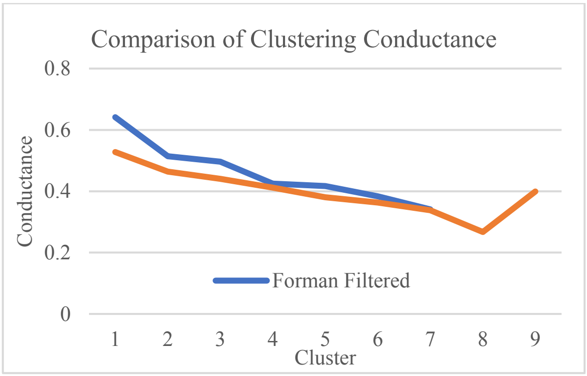
Data set	#Edges
All	16706
Forman Sampling	14850
Haantjes Sampling	14183
Coefficient Sampling	14648

In Figure 5.16, we show the clustering results for different sample sets. After Forman-Ricci curvature sample selection, the conductance of the sample sets is 15.18% greater than that of the original data. After Haantjes-Ricci curvature sample selection, the conductance of the sample sets is 7.57% less than that of the original data. After Clustering Coefficient curvature sample selection, the conductance of the sample sets is 15.035% less than that of the original data. Both Haantjes-Ricci and Clustering Coefficient curvatures are based triangle countering which provide more stable connections than the edge counting. That is why Haantjes-Ricci and Clustering Coefficient curvature sample selection provide better sample quality than Forman-Ricci curvature sample selection methodology.

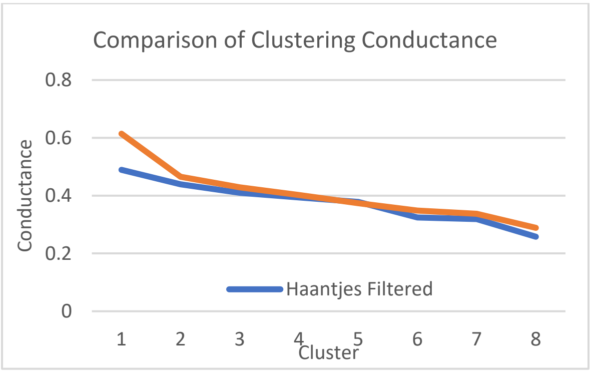


(a)References to Paper Network. (b)Paper to References Network

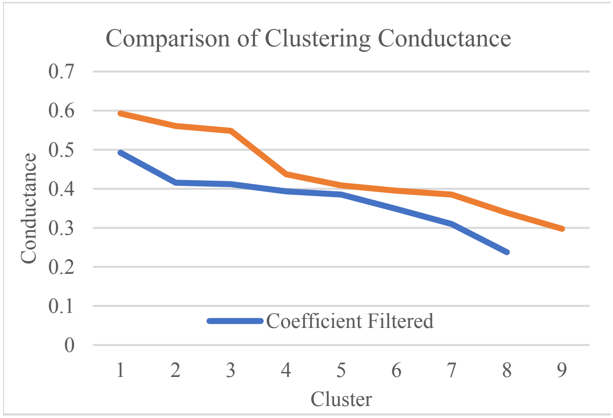
Figure 5.15. Citation Network Sampling



(a)Forman-Ricci Curvature



(b)Haantjes-Ricci Curvature



(c) Clustering Coefficient Curvature

Figure 5.16. Comparison of Sampling Methodology Based on Clustering Conductance

We use citation network to test our methodology. In the original data set, we had 27770 nodes and 352324 edges. After sampling, we kept 1308 nodes and 2688 edges. The sample set had 0.5% of the original data. However, community network structures were formed, and all of the network properties are kept. The citation network was visualized in Figure 5.15.

5.7 Conclusion

We presented two new network sampling methodology: Kronecker graph double cover and graph curvature.

By using Kronecker graph double cover methodology, graphs can be decomposed and reconstructed into subgraphs with both graph properties and topology types unchanged. Graph sampling is conducted through merging similar vertices and edges, in order to keep the connectedness of the network. Other than Erdos-random graphs that cannot be real-world graphs, graph topology types normally have hierarchical structures that can be divided into two regions: core and peripheral regions. To evaluate the performance of Kronecker double cover, we evaluate several different ways to separate core and periphery regions so that we can generate several sample sets. We evaluate the degree rank and degree frequency between the original and the sample sets, and use Pearson correlation coefficient to evaluate how much the sample sets are similar to the original. We also use Louvain clustering to evaluate the quality of the original and the sample sets.

For curvature methodology, we implemented Forman-Ricci curvature, Haantjes-Ricci curvature, and clustering coefficient curvature sampling on Email-Eu data set. Also, we compared the performance of the three curvature methods by performing data clustering on both sample sets and original data set. The evaluation of the performance was done by measuring the clustering conductance. Experimental results showed us that Haantjes-Ricci and Clustering Coefficient curvature sample selection can provide better sample quality than Forman-Ricci curvature sample selection methodology.

Both Kronecker double cover methodology and curvature methodology can maintain the graph properties and topology types and reduce the sample set sizes as well. After sample selection, the data quality of the sample sets remains.

Chapter 6: Entity Detection on Social Graph

6.1 Introduction

Web-scale knowledge bases build world knowledge online for question answering, recommendations, and personal agents. The generally used applications include Wikipedia, Google Knowledge Vault, Freebase, etc.

However, these knowledge bases are far from complete and are still growing. Named Entity Recognition (NER), is one of the important sub-tasks in information retrieval, also called Automatic Content Extraction (ACE), and has a big impact on the quality of the Natural Language Processing (NLP). With ACE, new entities can be constantly extracted and added to the knowledge bases for the purpose of indexing, searching, categorization, etc.

However, the entity representation is a challenge in natural language processing because of long-range dependencies, multiple interacting features, synonyms, etc. This issue motivates us to look at conditional probabilities of possible label sequences given observation sequencies. The conditional probability of the label sequences depends not only on the order of the word sequences but also the pre-defined arbitrary features. The context of the words can be used to uniquely define the semantic meanings of the words. The domain knowledge can be embedded in arbitrary features which can flexibly represent the characteristics of the observations at different levels of granularity.

The two techniques: (1) the context of words and (2) the domain knowledge, together can effectively improve the accuracy in labelling sequences. This idea has been implemented in machine learning based NER. Existing NER techniques can be based on rules, dictionaries, and machine learning.

Rule-based NER uses, other than part-of-speech (POS) tags, simple dependencies [76], grammatical features [204] and contextual features [101] to build rules for entity labelling. Rule-based NER systems can barely cover all the domain knowledge, be hard to expand and have the risk of overfitting.

Dictionary-based NER can be used to recognize both names and unique concepts. However, domain terms often have many variants which makes it difficult to have a complete dictionary so that fuzzy dictionary matching [46] [267] [239] appears to be a better approach when dictionary lookup is needed. In [46], protein functions were extracted from literature by identifying morphological, syntactic, and semantic variations. In [267], both bio-entity dictionary and abbreviation definition algorithm were developed to identify bio-entity names. Dictionary-based NER has two problems [239]: one is that short names can have a high probability to be predicted wrong, the other is variants of terms cannot be included in dictionaries. The first one was solved through entity classification. The second one was solved through string searching algorithm. However, both rule-based NER and dictionary-based NER are weak in adopting to new terminology.

Machine learning based NER, especially Conditional Random Fields (CRF) model, are generally used because they allow arbitrary features and use conditional probability instead of term frequency to estimate the labels of the entities. In [115], bi-directional CRF models were trained to tag gene names and gene product mentions in scientific text. The symmetric nature of the CRF was discussed and additional backward parsing makes CRF bi-directional. In [181], a semi-supervised integration of completely different classifiers for bio-name recognition was proposed to cover knowledge in unlabeled data. These technical advantages can ensure machine learning based NER is expandable to new terminology, flexible to different domain knowledge, and capable to long-distance dependencies.

The extraction of domain specific entities has been discussed in several fields, such as multi-language translation, clinical data analysis, pharmaceutical data analysis, customer profile analysis, online news analysis, etc. However, for different entities, the challenges in information extraction can be different. For this research, we focused on the product descriptions analysis in beverage industry and defined solutions to extract packaging information from product descriptions.

In the beverage industry, packaging information can be found in invoices and be used to verify transactions, retrieve sales performance, and evaluate market performance of a product. Although product descriptions are short but not composed of full sentences with proper grammar, they are written in a natural manner by salespeople and follow personal writing habits. Packing information provides detailed information about the quantities and units of the package, such as the number of packs, the number of cases, liquid volume, return or no return bottles, etc. The information needs to be extracted from the free text and to be put into structured tables for future query purposes. However, each year, billions of invoices can be created so that automatically extracting packaging information from invoices would become a big challenge. In this research, we defined the rich feature set for packaging information extraction by using (Conditional Random Fields) CRF.

6.2 Related Work

Statistical hidden state sequence model includes HMM (Hidden Markov Model) [147][95], CMMs (Conditional Markov Model) [75], CRF (Conditional Random Fields) model [141], are prominent recent approaches to information extraction tasks. These models are Markov-like models, and all encode the Markov properties: decisions about the state at a particular position in the sequence can depend only on a small local window. The mean idea is that: consider we have a hidden state sequence model which defines a probability distribution over state sequences conditioned on any given input. With such a model M we should be able to compute the conditional probability P of any state sequence S given some observed input sequence O.

Although Conditional Random Field model encompasses HMM-like models, the class of conditional random fields is much more expressive, because it allows arbitrary dependencies on the observation sequence. A conditional model specifies the probabilities of possible label sequences given an observation sequence. Furthermore, the conditional probability of the label sequence can depend on arbitrary, non-independent features of the observation sequence without forcing the model to account for the distribution of those dependencies. Conditional Random Fields (CRF) offer several advantages over Hidden Markov Models (HMM), including the ability to relax strong independence assumptions made in those models and avoiding the label bias problem. These advantages are derived from computing the conditional probability of an observation sequence.

Named Entity Recognition (NER) and Relation Extraction (RE) can be done in two stages [231]. NER was performed by combining a Bi-directional Long Short-Term Memory (Bi-LSTM) and a Conditional Random Fields (CRF) model. RE was implemented by a Convolutional Neural Network (CNN). Other than word embeddings, no domain specific knowledge was required to train the system so that it can be expanded to support different languages and clinical applications.

Rich features have strong impact on Named Entity Recognition (NER). In [1], rich features were defined in feature-based models (e.g. CRF) to build text mining systems for drug name recognition, Drug-Drug Interaction (DDI) extraction and DDI classification. The features can be based on bio-format, token, linguistic, binary, and semantic features. In [170], the feature sets were defined by Unicode tokenized, English tokenized, Part of Speech (POS) tags, Gazetteers, etc.

In health care industry, Named Entity Recognition (NER) can be applied and improved for different applications. In medical and biological documents, new proper nouns can be made frequently. PROPER [76] was proposed to extract material names, especially unknown words by using *surface clue* on the character strings. In [267], both CRF model and edit distance were used to extract bio-entities. CRF model was used with eight pre-defined features, such as surface word features, orthographic features, prefix/suffix features, word shape features, compound features, part of speech features, keyword features, and boundary word features. Edit distance algorithm was used to calculate the minimum number of operations on individual characters required to transform one string of symbols into another, so that the similarity of two strings can be measured. In [239], two issues in dictionary-based approach were solved for bio-entity recognition: one is false recognition caused by short terms which can be solved through string searching, the other is low recall due to spelling variations which can be solved by expanding dictionary with a probabilistic variant generator. In [204], one challenge in gene symbols and names recognition was that these names and symbols do not appear to follow construction rules. A program was used to distinguish the words of different natures, lexical, morphological, and semantic in sentences.

Deep learning can also be applied to Named Entity Recognition (NER). In [242], LSTM-CRF was used to recognize drug names and clinical concepts. The feature sets include morphological features, semantic features, and the clusters of the embeddings. In [231], drug-drug interactions from texts were detected by Bi-LSTM+CRF and relation extraction was done by CNN.

6.3 Definitions

The probability of label sequence y given observation sequence x can be written as the product of the appropriate elements of the n+1 matrices for that pair of sequences:

$$P(y \mid x, \lambda) = Z(x)^{-1} \prod_{i=1}^{n+1} M_i(y_{i-1}, y_i \mid x) \quad (6.1)$$

where $Z(x)^{-1}$ is a normalization factor and each $\{M_i(x) | i = 1, \ldots, n+1\}$ is a $| y * y |$ matrix with elements of the form

$$M_i(y^{'}, y_i | x) = \exp(\sum_j \lambda_j f_j(y^{'}, y, x, i)). \qquad (6.2)$$

where each $f_j(y^{'}, y, x, i)$ is either a state function $s_j(y^{'}, y, x, i)$ or a transition function $t_j(y^{'}, y, x, i)$.

Maximum likelihood training determines parameter λ values such that the logarithm of likelihood is maximized. For a CRF, the log-likelihood is given by

$$L(\lambda) = \sum_k [\log(Z(x)^{-1}) + (\sum_j \lambda_j f_j(y', y, x))]. \quad (6.3)$$

Differentiating the log-likelihood with respect to parameter λ_j gives

$$\partial L(\lambda) / \partial \lambda_j = E_{p'(Y, X)} [F_j(Y, X)] - \sum_k E_{p(Y|x^{(k)}, X)} [F_j(Y, x^{(k)})]. \quad (6.4)$$

where p' is the empirical distribution of training data and E_p [.] denotes expectation with respect to distribution p. Note that setting this derivative to zero yields the maximum entropy model constraint: the expectation of each feature with respect to the model distribution is equal to the expected value under the empirical distribution of the training data.

To identify the maximum-likelihood of parameter values, it must be possible to efficiently compute the expectation of each feature function with respect to the CRF model distribution for every observation sequence $x^{(k)}$ in the training data, given by

$$E_{p(Y|x^{(k)}, X)} [F_j(Y, x^{(k)})] \quad (6.5)$$

$$= \sum p(Y_{i-1} = y', Y_i = y | x^{(k)}, \lambda) F_j(Y, x^{(k)})$$

$$= \sum_{i=1}^{n} \sum_{y, y'} p(Y_{i-1} = y', Y_i = y | x^{(k)}, \lambda) f_j(y', y, x^{(k)})$$

We use a dynamic programming method to calculate $p(Y_{i-1} = y', Y_i = y | x^{(k)}, \lambda)$. Defining forward and backward vectors $\alpha_i(x)$ and $\beta_i(x)$ respectively by the base cases:

$$\alpha_0(y|x) = 1 \text{ if } y = \text{start; 0 otherwise} \quad (6.6)$$

$$\beta_{n+1}(y|x) = 1 \text{ if } y = \text{stop; 0 otherwise} \quad (6.7)$$

and the recurrence relations

$$\alpha_i(x) = \lambda_{i-1}(x)^T M_i(x) \quad (6.8)$$

$$\beta_i(x) = M_{i+1}(x)\, \lambda_{i+1}(x) \quad (6.9)$$

The probability of Y_i and Y_{i-1} taking on labels y' and y given observation sequence x(k) may be written as

$$p(Y_{i-1} = y', Y_i = y|x^{(k)}, \lambda) = \alpha_{i-1}(y'|x)\, M_i(y',y|x)\, \beta_i(y|x) / Z(x) \quad (6.10)$$

Another attractive property is the convexity of the loss function; indeed, CRFs share all the convexity properties of general maximum entropy models. From the HMM and for each of the trials trained, the most probable state sequence was calculated using the Viterbi algorithm.

6.4 Semantic Graph Smoothing

In a segment, the labels Y_i of the input text X_i are the vertices. The transaction between the adjacent labels, Y_{i-1} and Y_i, is the edge, E_i.

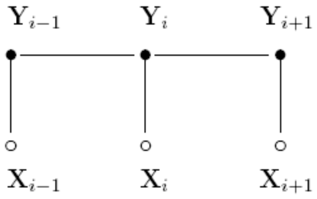


Figure 6.1. Chain-structured CRF Model for Sequences

As shown in Figure 6.1, and Figure 6.2, in our problem, X_{ij} (i = 1, ..., n and j=1, ..., n) is the set of texts, also called the observation sequence. Y_{ij} (i = 1, ..., n and j=1, ..., n) is the set of labels, also called the label sequence which is the unknown/hidden state variables. For this application, it includes non-boundaries, and boundaries of the objects.

In a CRF, given the observation sequence $\vec{X}$ in the graph, the distribution of each discrete random variable $\vec{Y}$ can be computed under the context. In input sequences, under the layout that $\vec{Y}$

is defined as "labels" and $\vec{X}$ as observations, efficient algorithms can be applied to optimize model training and conditional distributions learning to inference and determine the probability of individual labels and the most likely label sequences.

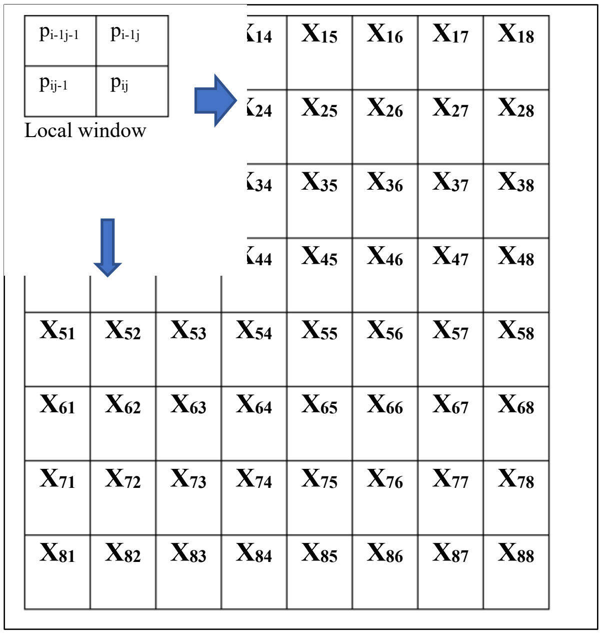

p_{i-1j-1}
p_{i-1j}
p_{ij-1}
p_{ij}
Local window
X_{14} X_{15} X_{16} X_{17} X_{18}
X_{24} X_{25} X_{26} X_{27} X_{28}
X_{34} X_{35} X_{36} X_{37} X_{38}
X_{44} X_{45} X_{46} X_{47} X_{48}
X_{51} X_{52} X_{53} X_{54} X_{55} X_{56} X_{57} X_{58}
X_{61} X_{62} X_{63} X_{64} X_{65} X_{66} X_{67} X_{68}
X_{71} X_{72} X_{73} X_{74} X_{75} X_{76} X_{77} X_{78}
X_{81} X_{82} X_{83} X_{84} X_{85} X_{86} X_{87} X_{88}

Figure 6.2. Computation of Local Probability of the Label for a Given Observation

A label sequence consists of several labels for the features. Each feature can be labeled as different symbols with different probabilities. We select the maximum probability of the feature as the feature probability. The combination of all the feature probabilities in a sequence is the probability of the given label sequence. The most probable label sequence was calculated using the Viterbi algorithm [247] to find the maximum probability of the feature sequence.

In CRF model, we can define artificial features based on context. Many papers [170], [56]. [1], discussed how to define features. There are several different kinds of features, such as, global features, local features, derived features, domain knowledge, etc. If the common patterns are defined correctly, CRF models continuously promote features and suppress non-features and eventually converge. But, if the common features are defined incorrectly so that CRF cannot systematically promote some data points and depress some data points, the model will never converge. In order to identify text segments, we consider gradients as the significant feature. Because, when there is a feature sequence in the texts, the gradients of their corresponding values are similar so that these gradients go to the same directions.

CRF models can be considered as Markov random fields models. The initial transition probability of each word is the same. During each iteration, the state function can average out the differences among neighbors, and the transition functions update the featured texts. Apparently, state functions make the local texts similar to each other. This operation is especially useful when some words are changed because of noise or system errors. When the significance of each word is computed not only with the value of the word itself but also the values of the adjacent neighbors, system errors and arbitrary noise can be averaged out. Feature functions are defined with the features we select so that they can consistently increase the significance of features and suppress the significance of non-features. As an undirected graphical model, the size of the local graph can be arbitrary, and the computation is based on the local graph we defined.

Given observations, the probability of label sequences can be computed with 6.11,

$$F(Y,X) = \exp\left(\sum_{j,i} \lambda_j t_j(y_{i-1}, y_i, x) + \sum_{k,i} \mu_k s_k(y_i, x)\right) \qquad (6.11)$$

in which $t_j(y_{i-1}, y_i, x)$ is the feature function, $s_k(y_i, x)$ is the state function, λ_j and μ_k need to be estimated with the training data. The reason CRF model is useful is because the feature function can be customized and can be combined with domain knowledge easily. In entity segment detection, we use gradients as features to define feature functions. For example,

$$t_j(y_{i-1}, y_i, x) = \begin{cases} b(x,i) & if\, g_{x(i-1),y(i)} = g_{x(i),y(i)}, or \\ & g_{x(i),y(i-1)} = g_{x(i),y(i)},\ or \\ & g_{x(i-1),y(i-1)} = g_{x(i),y(i)} \\ 0 & otherwise \end{cases} \qquad (6.12)$$

$$b(x,i) = \begin{cases} 1\ if\ 'WORD'\ IN\ LOCATION \\ \quad or\ ACTIVITY \\ \quad or\ PERSON \\ \\ 0\ otherwise \end{cases} \tag{6.13}$$

in which, $g_{x(i),y(i)}$ is the value of the gradient at (x_i, y_i).

The weakness of most of the entity segment detection methodologies is that, to determine the gradient similarity, a threshold must be selected in the first place and there is no way to quantitatively compute it. With CRF, the probability of each point is determined by not only the value of the data point but also its neighbors, which makes the algorithm false tolerant. Some words may not be picked as features, but, if those words are between two entities, there is a high probability that they can be defined as part of the entities. In other words, the decision on each data point is made based on not only the value of the data point, but also, its adjacent neighbors. Also, local probabilities can be adjusted globally through repeated training.

6.5 Feature Sets

In package descriptions, there are several challenges in packaging information extraction: ambiguous meanings of the words, inconsistent coding style and partial units. Ambiguous meanings of the words can happen to either numbers or the units. For example, digital numbers can be divided into several categories, such as units, vintages, proofs, dates, time, invoice series number, sales IDs, and company names, in which only units need to be extracted as packaging information. Acronyms can be divided into several categories, such as units, and brand names in which only units need to be extracted. Coding styles can be either: (1) quantity followed by units or (2) units followed by quantity, such as (1) “WHITE CLAW 2 VARIETY 12 P CAN”, (2) “WHITE CLW 2 VRTY P 12 CAN”. Partial units can occur as only quantities are in product descriptions but no units, such as “DOS EQUIS AMBER 2/12 LONG NECK”.

The meanings of units are based on the context, especially previous words and next words. For example, in “WHITE CLAW 2 VARIETY 12 P CAN”, *P* means *pack*. But, in “14 HANDS P GRIGIO 750 ML”, *P* means *Pinot*. P can be the short-hand of *proof* or the short-hand of *pack* in which we want to segment it if it is *pack* instead of *proof*, O can be the short-hand of OZ or the short-hand of proof in which we want to segment it if it is *OZ* instead of *proof*, LT can be the short-hand of *Liter* or the short-hand of *Miller Lite* in which we want to segment it if it is *Liter* instead of *Miller Lite*. Words can be synonyms. For example, quart can be written as *QUART|QT|QRT|QTS*, and gallon can be written as *GA|GL|GAL|GALS|GALLON*.

When there is a possibility that certain words may have multiple meanings, we need to create arbitrary rules to determine whether words are related to packaging, so that we can label packaging information based on the neighboring words in sentences. In packaging information extraction, we can define arbitrary rules based on coding formats, lexical formats, and part of speech tagging.

The feature sets can focus on semantic features of the words. In [170], the feature sets were defined by Unicode tokenizer, English tokenizer, POS tagger, Gazetteers, and also Gazetteer for degraded texts. For tokenized Unicode, the text can be split into simple tokens, such as numbers, punctuations, and words of different types, which distinguish words in upper case and lower case. For English tokenizer, we need to construct abbreviations into one token, such as “ ’30s ”, “ ‘Cause ”, “ ’s ”, “ ’d ”, “ ‘ll ”, “ ’ve ”, and do stemming to words in order to reduce the number of variations. For Part Of Speech (POS) tagger, we do part of speech tagging for words after the English text is processed. POS taggers are both domain-dependent and language-dependent. The tags are Penn TreeBank style and were defined in [56]. The taggers use default lexicon and ruleset but can be manually modified if necessary. For gazetteers, gazetteer lists are in plain text format and contain a set of names, such as city names, organization names, days of the week, etc. An additional index file is used to manage the lists. The index can be built into a finite state machine for term matching. For gazetteer for degraded texts, the alternative version of gazetteer lists is not case sensitive so that, if the parameter of the default gazetteer can be ambiguous because of the case insensitive, those words can be placed into specific lists.

In [1], the feature sets were defined by token, linguistic and semantic features. Token features include the original word and the lemma of the current word, and one token before and after the current word and their lemmas. Linguistic features include POS tags of the current word and its proceeding and following words; the length of the current, previous, and next words; and the suffix and prefix of the current word. Binary features can be defined as: (1) the current word is a number, (2) the current word has a decimal number, (3) the current is a unit, and (4) the current word is a symbol. Semantic features can be defined as a list of stop words, a list of abbreviations, a list of domain specific terms and a list of units of measurement. Semantic features can be utilized in binary functions to show the presence/absence of the word and in feature functions to indicate the name of the terms.

In beverage industry, packaging information has frequently used patterns. In packaging segments, we summarized all the patterns used frequently, such as “4/6/12 oz” which means 4 packs, 6 bottles per pack, and 12 oz per bottle, “750 ml”, etc. Other than quantity and units, separators are also frequently used, such as “/” and “-”. For the purpose of semantic analysis, part of speech tags plays an important role in identifying meaningful words. In other words, the feature sets include the list of units, quantities, number formats (such as “decimal”, “digital”, etc.), separators, and POS tags. The window we define include the previous, current, and next words. Feature functions directly return words instead of binary values. However, one drawback of the feature functions is that, for the beginning and the end of the sentences, there is no previous words or next words, that cause the words at the beginning and the end of the sentences short of enough features to tune the probability. We also add the beginning, and the end of the sentences features to feature sets to help adjust the probability of the words at the beginning and end of the sentences.

6.6 Experiments

In preprocessing, we built a tokenizer to scan and normalize the entire data set. Pre-formatting processes in the tokenizer split text into uniformed tokens. Upper case and lower case are not distinguished, and all text is converted into upper case. Special characters are filtered out. Special patterns can be uniquely formatted into predefined patterns. For example, 1/2, 1 / 2, 1-2, 1 - 2, 1X2 and 1 X 2, can be converted into 1 / 2. The combination of quantities and units are split into quantities and units. For example, 12A and A12 can be split into 12 A and A 12. The combination of units can be split into separated units. For example, Z/NR (in which Z means OZ for ounces, and NR means NO RETURN) can be converted into Z / NR. For abbreviations, we use a collection of abbreviations to build an abbreviation dictionary and search the dictionary to convert abbreviations to standardized units.

For tagger, we use a modified version of Natural Language Toolkit (NLTK) tagger, that produces part-of-speech (POS) tags for words and symbols. The tags used are Penn Treebank style. However, some tags need to be modified based on the domain knowledge. For example, *A* in beverage industry is the name of the unit - *pack* so that its POS tag is NN which means a noun, instead of DT which means a determiner, such as *a*, *the,* etc.

Packaging information can be extracted automatically through Natural Language Processing (NLP), especially by using Conditional Random Fields (CRF) model. CRF is a local algorithm that does not use the constraints found in the Hidden Markov Model (HMM), that makes it possible to use local information to make predictions. CRF can efficiently segment, and label sequences based on sentence context and the adjacent words within the sentence, similarly to how people naturally read and write. In Figure 6.3 is entity extraction process.

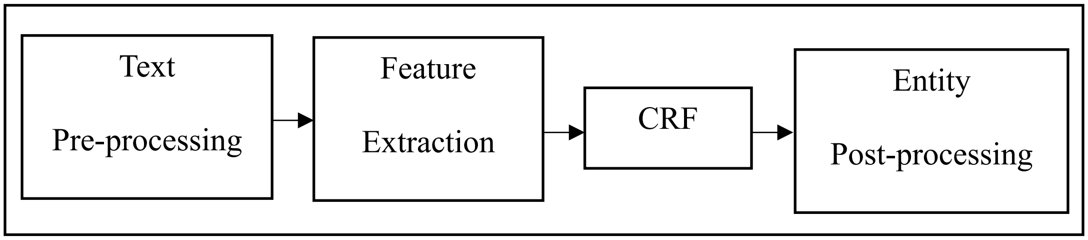


Figure 6.3. Entity Extraction Process

In our problem, X is the set of words, also called the observation sequences, and Y is the set of labels, also called the label sequence-the unknown/hidden state variables. In text, each label y_i is a vertex. The transaction between the adjacent labels, y_{i-1} and y_i, is the edge e_i. For example, X can be "Patron Silver TEQ 80 Proof 12 Pack 750 ml", and Y can be "O O PACK PACK O PACK PACK PACK PACK". For this application, it includes 'O' (background symbol), and 'PACK' (packing features).

In the CRF model, we can define artificial features based on context. Many papers discuss how to define features. There are several different kinds of features, such as global, local, Part Of Speech (POS) tags, bigram, acronyms, word shape, previous word, next word, and current word features.

For packaging information, we want to promote units, quantity, and the segments of the quantity and units. The arbitrary features we defined are based on the properties of current words and the properties of previous and next words. In other words, we use words and relationships between words to define arbitrary features. Some units can be spelled in several different ways that can cause synonym issues. Those words are not included in the arbitrary feature set. However, during training, the probability of those words can be promoted by their adjacent neighbors. We list selected features in Table 6.1.

Table 6.1. Usage of Features in Feature Windows in Packaging Information Extraction

Features	Usage		
	Current	Previous	Next
Units	Yes	Subset	Subset
Separators	Yes	Yes	Yes
Decimals	Yes	No	No
Numbers	Yes	No	No
POS (Part Of Speech) tags	Yes	Yes	Yes
POS tags (first two letters)	Yes	No	No
BOS (Beginning Of Sentence)	Yes	No	No
EOS (End Of Sentence)	Yes	No	No

In a CRF, given the observation sequence X, each discrete random variable Y can be conditionally computed under the context. In principle, because CRF is a Markov-like model so that the layout of the graph can be arbitrary; however, for text mining, the vertex y_i are constructed to form a chain with an edge between vertex y_{i-1} and vertex y_i. Given y_i as "label" for each observation x_i in the input sequence, we need to compute (1) the conditional distributions between y_i and its neighbors based on feature functions (2) the probability of a y_i given x_i, and (3) the maximum probability of the label sequence Y given X.

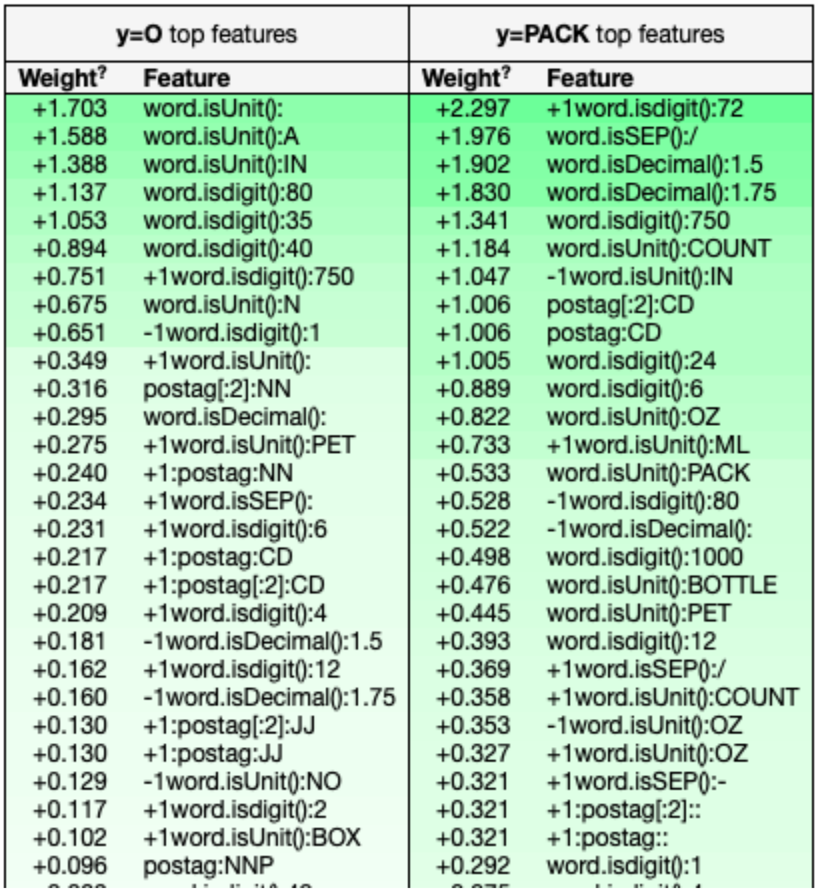
A label sequence consists of several labels for the features. Each feature can be labeled as different symbols with different probabilities. We select the maximum probability of the feature as the feature probability. The combination of all the feature's probabilities in a sequence is the probability of the given label sequence. The most possible sequence was calculated using the Viterbi algorithm [247], that can be used to find the maximum probability of the feature sequence. The weakness of this method is that adding feature sets makes the total data set size many times larger than the original data set. However, because CRF is a local algorithm, only local windows need to be loaded into memory so that the space cost for each computation is limited.

When we trained the CRF models, the feature set consisted of previous words, next words, current words, and word format. We used the CRF model to calculate the weights of all the features. For testing, we used the sum of all the features as the probability of the current word. If the current word is mislabeled in training, suppose the weight of the current word is 0, but the weights of the previous word, next word, and word type features may not be 0. In this way, even though some words are not trained, we can use CRF model to label them with a high probability.

We use alcohol product descriptions from both manual input (PRODUCT_PAYMENT_DATA) and master catalog input (PMASTER). Each set has 272 million records. However, PRODUCT_PAYMENT_DATA has more random information in it because it is real-world data. PMASTER data is manually cleaned and usually follows the same format: the number of packs/number of bottles per pack/volume per bottle, container information, package information. In terms of feature engineering, it is easier to promote features in PMASTER data. In terms of training samples, since PRODUCT_PAYMENT_DATA has more random descriptions, it requires a larger sample set to cover all the variants.

As previously mentioned, since the output results will be used to verify invoices, we require the results have a high accuracy so that the results can play the role of trusted evidence. In order to meet this requirement, we divide the data into three subsets based on randomness of the input:

1. There is no quantity information in it.
2. There is one number or “PACK/CASE/VOLUME” pattern in the description that is also followed by units with no ambiguous shorthand.
3. The remaining data will have both multiple quantities and multiple units in it. Quantities and units are loosely coupled. Some units may be spelled in different ways and some of the shorthand may have multiple meanings. Some numbers don’t need to be extracted because they are either part of the product names or unrelated information.



y=O top features		**y=PACK** top features	
Weight?	**Feature**	**Weight?**	**Feature**
+1.703	word.isUnit():	+2.297	+1word.isdigit():72
+1.588	word.isUnit():A	+1.976	word.isSEP():/
+1.388	word.isUnit():IN	+1.902	word.isDecimal():1.5
+1.137	word.isdigit():80	+1.830	word.isDecimal():1.75
+1.053	word.isdigit():35	+1.341	word.isdigit():750
+0.894	word.isdigit():40	+1.184	word.isUnit():COUNT
+0.751	+1word.isdigit():750	+1.047	-1word.isUnit():IN
+0.675	word.isUnit():N	+1.006	postag[:2]:CD
+0.651	-1word.isdigit():1	+1.006	postag:CD
+0.349	+1word.isUnit():	+1.005	word.isdigit():24
+0.316	postag[:2]:NN	+0.889	word.isdigit():6
+0.295	word.isDecimal():	+0.822	word.isUnit():OZ
+0.275	+1word.isUnit():PET	+0.733	+1word.isUnit():ML
+0.240	+1:postag:NN	+0.533	word.isUnit():PACK
+0.234	+1word.isSEP():	+0.528	-1word.isdigit():80
+0.231	+1word.isdigit():6	+0.522	-1word.isDecimal():
+0.217	+1:postag:CD	+0.498	word.isdigit():1000
+0.217	+1:postag[:2]:CD	+0.476	word.isUnit():BOTTLE
+0.209	+1word.isdigit():4	+0.445	word.isUnit():PET
+0.181	-1word.isDecimal():1.5	+0.393	word.isdigit():12
+0.162	+1word.isdigit():12	+0.369	+1word.isSEP():/
+0.160	-1word.isDecimal():1.75	+0.358	+1word.isUnit():COUNT
+0.130	+1:postag[:2]:JJ	+0.353	-1word.isUnit():OZ
+0.130	+1:postag:JJ	+0.327	+1word.isUnit():OZ
+0.129	-1word.isUnit():NO	+0.321	+1word.isSEP():-
+0.117	+1word.isdigit():2	+0.321	+1:postag[:2]::
+0.102	+1word.isUnit():BOX	+0.321	+1:postag::
+0.096	postag:NNP	+0.292	word.isdigit():1

Figure 6.4. Top 30 Features for PACK Class (Packing Information) and O Class (Others)

In Figure 6.4, we can see the top features for packing class, such as PACK, OZ, COUNT, BOTTLE, and some frequently used quantities for packing. In top features for background, we can see non-units, such as NN, N, A, IN, and proof related quantities. These weights of the features can efficiently promote the significance of the packing information and suppress the background information.

We build CRF models for both level 2 and level 3 and test them separately. The testing process is defined as the following:

1. Step 1. We use a CRF model to make predictions on level 2 data and level 3 packing data.
2. Step 2. We use regular expression to count, for each record, how many numbers are in the original description and how many numbers are extracted. If there is a difference between the two counts, we manually verify the record to make sure the quantity and the units are extracted correctly.

When all the quantities are extracted from the descriptions in both level 2 and level 3 data, the units are correspondingly extracted correctly.

The quantities that do not get extracted from the level 2 data descriptions are related to proofs and vintages. In 10,000 records, 1/10,000 is extracted incorrectly. The misclassified results from descriptions are because they have only one or two numbers, or do not have units, or are too short. For level 1 data, there are various reasons as to why we receive incorrect results. Some numbers can be directly excluded from descriptions, such as alcohol proofs, vintages, and most of the numbers in brand names. When the numbers are frequently used in both packaging information and background information, those numbers can have higher probability to be labeled as “PACK”. The frequency of this happening is 1/10,000.

The model was both trained and evaluated on 150,000 samples with measurements, such as precision, recall and F1-score. The overall cross-validation accuracy is 0.999931669 and other measurements can also reach 0.9999 for both "PACK" class and "O" class, as listed in Table 6.2.

Table 6.2. Performance of CRF on 150,000 Samples.

	Precision	Recall	F1-score
O	0.99998932	0.99993059	0.99995996
PACK	0.99996233	0.9999942	0.99997827
macro avg	0.99997583	0.9999624	0.99996911
weighted avg	0.99997183	0.99997183	0.99997183

We also compared the performance of our model with a deep learning model, called bi-directional Long Short-Term Memory (Bi-LSTM). We trained and evaluated Bi-LSTM on character levels with 150,000 samples. In Figure 6.5, after 50 iterations, the accuracy on testing set was 0.99, and on validation set was 0.98.

The CRF model with arbitrary features was more accurate than Bi-LSTM model. However, in experimental results, CRF model might fail at the beginning of the sentences and the end of the sentences, but Bi-LSTM might fail randomly anywhere in the sentences. In other words, the results of the CRF model were more meaningful and formed complete sentences, but Bi-LSTM can randomly break sentences. In terms of machine learning, the results of the CRF model were better than Bi-LSTM because the weaknesses of the model are more trackable and can be fixed in post-processing.

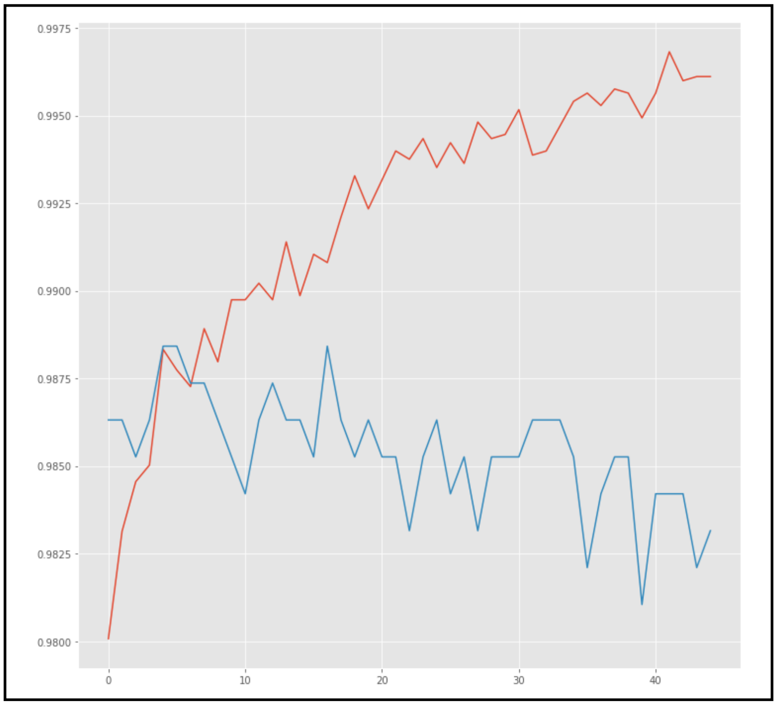


Figure 6.5. Performance of Bi-LSTM Model

6.7 Conclusion

We defined a feature extraction methodology for the CRF model to extract quantity and units from alcohol product descriptions. Based on the characteristics of the data set, such as short sentences, various formats, and many unlabeled features, we decided to use CRF to segment text sequences and to label units and quantities in the text sequences. Quantity and unit features can be divided into several categories, such as strong, ambiguous, and unrelated features. We design feature extraction functions to promote quantity- and unit- related features and use those features to identify ambiguous features based on the context. In experiments, in order to make it possible to verify each one of the results, we partition the data into three levels of subsets. For each subset, we focus on specific incorrect results and determine ways to solve them.

By using the feature extraction function defined for the CRF model, whenever quantities are extracted correctly, in the same records, the units can also be extracted correctly. The quantities extracted incorrectly are most likely because there is no unit in the description. In other words, the description is too short to provide enough strong features to promote the quantities, or most likely, the quantity is rarely used in other descriptions. This problem can be solved by adding more training samples, and especially by adding incorrectly labeled data into the sample set. After more samples are added to the training set, all quantities and units appear equally frequent, and the bias issue can be fixed.

We conducted experiments to test the performance of the CRF model with the arbitrary features we defined. We also compared the model with Bi-LSTM deep learning model. The experimental results showed that the CRF model performed better than Bi-LSTM. Also, in practice, the weakness of the CRF model can be fixed in post-processing, but the weakness of Bi-LSTM model was not trackable and cannot be fixed easily.

Chapter 7: Classification on Social Network

7.1 Introduction

Natural language is the most common ways to present information in different fields, in different perspectives and in different circumstances so that text becomes a rich source of information. However, with the development of information technology, the amount of documentation becomes impossible for human to handle. We need to take advantage of computers to help automatically understand documentation, group documentation into different categories, topics, and subjects until the total amount of information is capable for human to handle.

Text mining can be applied to question answering, spam detection, semantic analysis, news categorization, intent classification, to name a few. For different applications, text data can come from different sources, such as web pages, emails, chats, social media, tickets, product descriptions, invoices, insurance claims, user reviews and so on. Due to the unstructured nature, it is challenging and time-consuming to extract information, especially context-dependent terms from texts.

However, there are lots of open issues and challenges in text mining. For example, multilingual text documentation requires text refining algorithms to process the multiple languages and language-independent intermediate data. Also, personalized autonomous mining gives rise to new applications, such as user profiling, natural language query interpretation, and intelligent personal assistants. Another challenge is that the integration of domain knowledge could play an important role in text mining. Domain knowledge can help speed up text processing and increase the precision of the results. Domain-specific knowledge extraction requires semantic analysis to extract the association between the objects or concepts in the documentation.

Typical text classification uses machine learning technology to perform Natural Language Processing (NLP) and to assign labels or tags to textual units, such as terms, sentences, paragraphs, documents, and queries. Normally, machine learning-based methodology does classification in two steps: one is to select interesting features, the other is to feed features into classifiers to make predictions. Different from traditional machine learning, deep learning trains word embeddings as the starting point of the classification. This efficiently solves the issues in feature engineering and makes it possible to apply machine learning for many applications.

In terms of data modeling, text classification can be divided into two categories: one is based on maximum likelihood, the other is based on minimum energy. For maximum likelihood-based methodology, we have Naive Bayes, Support Vector Machines (SVM), etc. For energy-based methodology, we have Hidden Markov Model (HMM), Conditional Random Fields model (CRF), etc. The difference between the two categories is not only in the technical details but also in how many language patterns can be modeled. Normally, maximum likelihood-based methodologies consider words are independent tokens and use Bag of Words (BoW) to build sample sets. Minimum energy-based methodologies can not only fit models with individual words but also the associations between words, which make it possible to conduct semantic analysis. Deep learning is a separate architecture which trains word embeddings and the classification layer is the last layer in the architecture.

Deep Learning architecture is built upon neural networks. Neural approaches have the advantage of overcoming the limitations of feature engineering. Word embeddings convert input texts into an importance vector in which some words have higher significance, and some words have lower significance. In this way, the words with higher significance can contribute more to classification process and the words with lower significance can contribute less to classification process. It is optional to reduce the number of dimensions but, when the word embeddings are built, feature engineering is done.

Embedding models have a long history. The earliest embedding model is Latent Semantic Analysis (LSA) [64]. LSA is based on dimension reduction. The dimensions with larger Eigenvalues are considered more significant and can be kept. The dimensions with smaller Eigenvalues are considered less significant and can be removed. The neural network model was proposed in [28] and was based on a feed-forward neural network.

However, the early embedding models underperform classical models so that they were not generally used. In 2013, Google proposed word2vec [176][161] models that were trained on billions of words and can be applied to many NLP tasks. In 2017, a contextual embedding model based on a 3-layer bidirectional Long Short-Term Memory (LSTM) was trained on 1 billion words, that performs better than word2vec because it can capture the context and perform semantic analysis. The same year, Google developed Bidirectional Encoder Representation Transformer (BERT). BERT consists of 340 parameters, was trained on 3.3 billion words, and is the current state-of-the-art embedding model. The trend of using pre-trained larger models continues to be popular.

Although these large deep learning models show impressive performance on various NLP tasks, there are some challenges in data modeling. For example, it is hard to convert deep learning models into relational models. Linguistics as the domain knowledge in natural languages cannot be modeled in deep learning. For example, in entity resolution, we need to extract meaningful phrases from the text. The meanings of the words are defined by adjacent words and can be changed in different contexts. Some researchers argue that deep learning models do not really understand languages and are not robust enough for mission-critical domains.

Attention mechanism is a big breakthrough in deep learning. It becomes an increasingly popular concept and a useful tool in developing deep learning models for Natural Language Processing (NLP). Because it builds a shortcut of the context for input texts and promotes correlated words in one sentence. The importance vector in the attention layer can be updated through Markov-like updates and can be customized easily in classification. The prediction can be made by estimating how strongly the word is correlated with or attends to other words.

We propose a Featured Transformer Methodology (FTM) based on attention mechanism. It can efficiently add domain knowledge to deep learning architecture. The attention mechanism performs Markov-like updates. When the model converts, the associations between domain features and word embeddings can be extracted.

The remainder of the chapter is structured as follows: section 2 presents related work, section 3 introduces related definitions, section 4 explains the proposed methodology, section 5 evaluates the performance through experiments, and section 6 presents conclusions.

7.2 Related Work

7.2.1 Feed-Forward Networks (FFN)

Word representation can be considered as a multiple dimensional problem which requires high computation power. Originally, it can be solved by using Naive Bayesian Logarithm of Word count ratios as feature values for SVM to train a sentiment analysis model [254]. However, this methodology has been improved by deep learning by using FFN because of the high dimensional nature of the neural networks.

To compute the continuous vector representation of words, two models were proposed in [176] and were evaluated with word similarity task: one is bag of words model, the other is skip n-gram model. The idea is that we removed hidden layer for both of the two models and also defined the range for similar words for skip n-gram model. All these changes can efficiently decrease the computation complexity.

A global log-bilinear regression model was proposed in [176] that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. The model produces a vector space with meaningful substructure of 75% accuracy on a recent word analogy task, word similarity task, and entity recognition task.

Deep averaging neural network [166] added more averaging layers and applied dropout to improve performance.

Product quantization [126] was used to compress word embeddings and to reduce the size of the classification models. This method can lower the memory usage to two orders of magnitude and outperforms others by a good margin in memory usage and accuracy.

In [143], paragraph vector representation of the documents was learned to be fixed length feature representations from variable length pieces of texts, such as sentences, paragraphs, and documents. This way of constructing word vectors overcomes the weaknesses of Bag of Words (BoW) for not ignoring orders between words. It can achieve a new state-of-the-art performance on several text classification and sentiment analysis tasks.

7.2.2 Recurrent Neural Networks (RNN)

RNN based models view text as a sequence of words and are intended to capture word dependencies and text structures. RNN has issues in exploding and vanishing gradients when learning long distance correlations, which can be addressed in LSTM by using memory cell to preserve state over long periods of time. Other than the generally used linear chain structured LSTM models, a tree structured LSTM model was proposed in [235] which outperforms the linear chain structured LSTM models in semantic sentence similarity and sentiment classification.

The difference between the standard LSTM units and the Tree-LSTM units is that gating vectors and memory cell updates are dependent on the states of possibly many child units. Additionally, instead of a single forget gate, each Tree-LSTM unit (indexed by j) contains one forget gate f_{jk} for each child k, which allows the Tree-LSTM units to selectively incorporate information from each child. For example, a Tree-LSTM model can learn to emphasize semantic heads in a semantic relatedness task, or it can learn to preserve the representation of sentiment-rich children for sentiment classification. The input word at each node depends on the tree structure used for the network. In other words, each node in the tree takes the vector corresponding to the head word as input.

The forget gate controls the extent to which the previous memory cell is forgotten, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state. The hidden state vector in an LSTM unit is therefore a gated, partial view of the state of the unit's internal memory cell. Since the value of the gating variables vary for each vector element, the model can learn to represent information over multiple time scales.

TopicRNN [60] model integrates the merits of RNNs and latent topic models: it captures local dependencies using an RNN and global dependencies using latent topics. Unlike previous work on contextual RNN language modeling, this model is learned end-to-end.

LSTM was extended from chain structure to tree structure in which each memory cell can reflect the history memories of multiple child cells or multiple descendant cells in a recursive process. It can provide a principled way of considering long-distance interaction over hierarchies. It outperforms the state-of-the-art recursive model by replacing its composition layers with the S-LSTM [277] memory blocks and overcoming gradient vanishing in long-distance dependency. S-LSTM can be considered as a combination of recursive neural networks and recurrent neural networks. It wires memory blocks in a partial-order structures instead of in a full-order sequence as in a chain-structured LSTM.

A machine reading simulator to render sequence-level networks can process text incremental from left to right and perform shallow reasoning with memory and attention. The reader extends LSTM architecture by replacing the memory cell with a memory network which enables LSTM to reason about relation between tokens with a neural attention layer and then perform non-Markov state updates. Long Short-Term Memory-Network (LSTMN) [45] outperforms KN5, RNN and LSTM with less than 33, 21, and 7 in perplexity. In sentiment analysis, LSTMN outperforms Recursive Auto Encoder (RAE) [224], Recursive Neural Tensor Network (RNTN) [225], Dynamic Convolutional Neural Network (DCNN) [127], Deep Recursive Neural Networks (DRNN) [189], Convolution Neural Networks-Multichannel (CNN-MC) [Kim, Y. et.al. 2014], Tubelets with Convolutional Neural Networks (T-CNN) [129], Paragraph Vector (PV) [143], Constituency Tree-LSTM (CT-LSTM) [235], LSTM and 2-layer LSTM with accuracy 86.3% or 1-layer LSTMN and 87.0% for 2-layer LSTMN. In sentence inference, LSTMN deep fusion outperforms Bag of Words (BoW), LSTM, match LSTM (mLSTM) [255], with 86.3% accuracy.

A general framework jointly trains a feature generator and a linear model in which the feature generator consists of region embedding + pooling [159]. This framework can find an efficient way to explore a more sophisticated region embedding method using LSTM. The results show that embeddings of text regions representing complex concepts are more useful than embeddings of single words in isolation.

A region embedding method [125] [157] using LSTM can embed text regions of variable sizes, whereas the region size needs to be fixed in a CNN. Three RNN based architectures were introduced to model text sequence with multi-task learning. The differences among them are the mechanisms of sharing information among the several tasks. Experimental results show that our models can improve the performances of a group of related tasks by exploring common features.

RNN converts representation of texts into two-dimensional matrix [276]: the time step dimension and the feature vector dimension. 2D pooling operation over the two dimensions may sample more meaningful features for sequence modeling tasks.

By using Bi-LSTM, rich context of the whole sentence is leveraged to capture the contextualized local information in each positional sentence representation. As shown in [251], by matching with multiple positional sentence representations, it is flexible to aggregate different important contextualized local information in a sentence to support the matching. Experiments on different tasks such as question answering, and sentence completion demonstrate the superiority of our model. The model for semantic matching with multiple positional sentence representations, MV-LSTM, is defined as a sentence representation at one position. MV-LSTM can capture important local information by introducing multiple positional sentence representations. By using Bi-LSTM to generate each positional sentence representation, MV-LSTM has leveraged rich context to determine the importance of the local information. MV-LSTM performs better than ARC-I, CNTN, and LSTM-RNN. MV-LSTM obtains 11.1% improvement over LSTM-RNN and 5.6% relative improvement over MultiGranCNN. MV-LSTM works in three steps: (1) positional sentence representation through Bi-LSTM, (2) interactions between two sentences with cosine similarity measure, bilinear similarity measure and tensor layer similarity measure, and (3) interaction aggregation through k-max pooling and multi-layer perception.

7.2.3 Attention Mechanism

In [16], neural machine translation has encoder-decoder structure. It encodes a source sentence into a fixed-length vector from which a decoder can generate a translation. An automatic soft-search for parts of source sentences can solve the fixed-length vector issue and help improve the translation performance.

In [92], neural Turing machine was proposed to with attentional processes through which they can be coupled to external memory resources. The combined system allows it to be efficiently trained with gradient descent.

In [246], conditional probability of an output sequence can be learned through a neural architecture, named Pointer networks. Neural attention is different from the previous attention attempts in that it uses attention as a point to select a member of input sequences as the output, instead of aligning an encoder to a decoder through an importance vector. This methodology can be used to solve the problem of variable size output dictionaries.

7.2.4 Combination of Domain Knowledge

Text classification can be improved through domain specific knowledge [66], [3], [199], [78], in perspectives of tokenization [78], transfer learning [206], and domain specific features, such as the combination of both questions and answers [78], and the sarcasm identification [66].

7.3 Definitions

Attention mechanism works through a vector of importance weights in comparison to recurrent network architecture. Attention network architecture can discover connections with longer distances by using the context vector to provide shortcut connections between inputs and outputs. On the other hand, sequential models can remember a few states instead of the entire context.

For example, given a word in a sentence as the input, we use the attention vector to estimate how strongly it is related to or attends to other elements in the sentence/context and then take the sum of their values weighted by the attention vector as the approximation of the output.

The context vector consumes three pieces of information: encoder hidden states, decoder hidden states and alignment between source and target. Given a source sequence x = [x_1, x_2, …, x_n] to generate the target sequence y = [y_1, y_2, …, y_m], an attention mechanism is defined.

The encoder is a recurrent network, for example, bidirectional Recurrent Neural Network (RNN) with a forward hidden state $\overrightarrow{h_i}$and a backward one $\overleftarrow{h_i}$. The concatenation of the two can be used to represent the encoder state so that we can have both the preceding and the following words in the annotation of one word.

$$h_i = [\overrightarrow{h_i^T}, \overleftarrow{h_i^T}]^T, i = 1, 2, \dots, n \quad (7.1)$$

The decoder network has hidden state $s_t = f(s_{t-1}, y_{t-1}, c_t)$ for the output word at position t, t = 1, 2, …, m, where the context vector c_t is a sum of hidden states of the input sequence, weighted by alignment scores:

$$c_t = \sum_{i=1}^{n} a_{t,i} h_i \qquad (7.2)$$

$$a_{t,i} = align(y_t, x_i) = \frac{\exp(score(s_{t-1}, h_i))}{\sum_{i'=1}^{n} \exp(score(s_{t-1}, h_{i'}))} \qquad (7.3)$$

The alignment model assigns a score $a_{t,i}$ to the pair of input at position i and output at position t, (y_t, x_i), based on how well they match. The set of {$a_{t,i}$} are weights defining how much of each source hidden state should be considered for each input. In [16], the alignment score $a_{t,i}$ is parameterized by a feed-forward network with a single hidden layer and this network is jointly trained with other parts of the model. The score function is therefore in the following form, given that tanh is used as the non-linear activation function:

$$score(s_t, h_i) = v_a^T \tanh(W_a[s_t; h_i]) \quad (7.4)$$

where both v_a and W_a are weight matrices to be learned in the alignment model.

Attention mechanism can be divided into several categories: self-attention [45], global/soft attention [263], and local/hard attention [165]. Self-attention can be used to relate different positions of the same input sequence. The most popular self-attention architecture is LSTM. Global/soft attention can align attention weights over all patches, which can be done by using a window centered around the source position to compute a context vector after aligning position for the current target word. Local/hard attention can align attention weights one patch of the input at a time. The alignment of the attention weights is through alignment score functions, such as content-based attention, additive, location-base, general, dot-product, scaled dot-product score functions.

Transformer model [244] is built on self-attention mechanism. With a sequence transduction architecture, it has an encoder-decoder structure which is aligned by attention mechanism. Attention has three applications in transformer: attention layer for encoder, attention layer for decoder, and encoder-decoder attention layer for alignment. In other words, transformer follows neural sequence transduction architecture with both encoder and decoder which have stacked self-attention and pointwise, fully connected layers.

Attention functions are also called alignment score functions which can be divided into several categories according to the attention mechanism: content-based attention [92], additive attention [16], location-based attention [164], general attention [164], dot-product attention [164], and scaled dot-product attention [244]. Transformer in [244] uses scaled dot-product attention score function.

Attention mechanism simulates hidden Markov-like updates in which each attention point can be updated by all data points in previous layer. If the attention functions are selected properly, when the model is converged, long distance dependencies can be obtained.

Content-based attention was inspired by Neural Turing machine. As the first prototype of modern computers, Turing machine can conduct all kinds of computations with a controller and an unlimited tape as memory. Content-based attention focuses on content addressing based on the similarity between their current values and the values emitted by the controller. Content-based addressing can be a broad definition than location base addressing. Because contents can contain location information as well. A controller can take the values of variables and store them anywhere, retrieve them later and perform computation. In [92], the similarity measure is cosine similarity. The controller is central to the neural Turing architecture. The controller can be a feedforward network, a recurrent network and even a LSTM. Content base attention aligns system input to the controller output through similarity measure. In [92], the similarity measure is cosine similarity.

$$score(s_t, h_i) = cosine[s_t, h_i] \qquad (7.5)$$

Additive attention works by finding the maximum weighted sum of the output. This mechanism simulates sequence to sequence prediction problems. A typical application of additive attention is machine translation. Attention layer connects encoder layer and decoder layer on RNN or LSTM network architecture, in which conditional distribution of the sentence pairs can be learned and a corresponding translation for a given source sentence can be generated by searching for the sentence that maximize the conditional probability. Machine translation network architecture learns, aligns, and translates simultaneously. The attention alignment function is the weighted sum of the output of the encoder.

$$score(s_t, h_i) = \sum_{j=1}^{T_x} a_{ij} h_j \qquad (7.6)$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad (7.7)$$

$$e_{ij} = a(s_{i-1}, h_j) \qquad (7.8)$$

Dot-product attention can also be used to simulate sequence to sequence prediction problems in which long distance dependency can be preserved as well. A typical application of dot-product attention is transformer network architecture. Transformer network architecture is built upon convolution networks instead of recurrent networks, which can allow for more significant parallelization and efficiently reduce the total amount of computation. As shown below in equation (7.9), the transformer in [244] used a scaled dot-product score function which is like dot-product score function except a scaling factor.

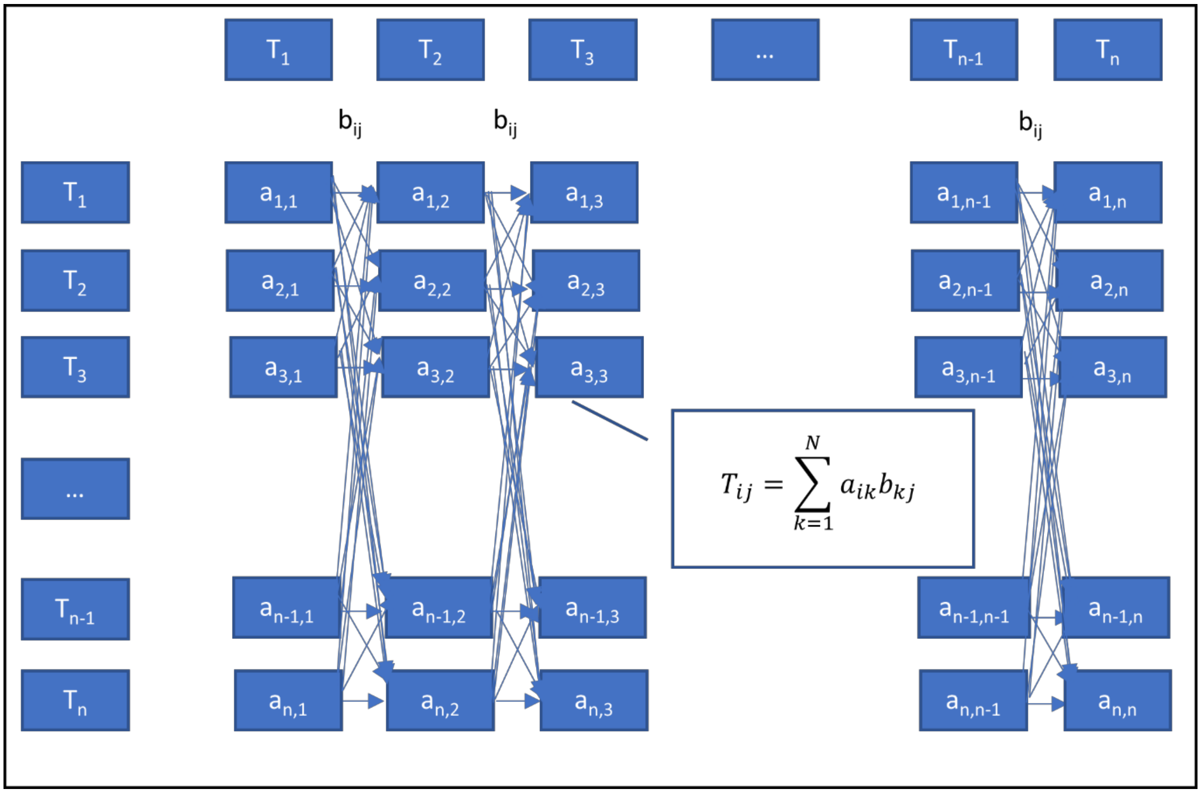
$$score(s_i, h_i) = \frac{s_i^T h_i}{\sqrt{n}} \quad (7.9)$$

7.4 Methodology

Transformer simplifies the neural network architecture and reduces computation cost. However, several open issues have not yet been solved, such as the combination of domain specific patterns, the combination of high-level associations, and unlabeled terms. Featured Transformer Methodology (FTM) can add arbitrary features to the network architecture can help solve feature combination, high level association combination, and unlabeled terms issues.

Bidirectional Encoder Representations from Transformers (BERT) has transformer network architecture. Attention layer plays the role of aligning encoder and decoder layers. Encoder and decoder layers are built upon convolution networks instead of recurrent networks to reduce computation costs. Attention layer can be updated through Markov-like process as shown in Figure 7.1.

Connections between two sequences are presented in two-dimensional matrix in which each dimension has a sequence of tokens. The more often the connections appear, the larger the values are. Attention mechanism is a shortcut of Markov Chain in which attention layer has less heads than the previous layer. As shown in Figure 7.2, in BERT network architecture, the importance vector of attention layer can align encoder to decoder and highly affect the final prediction. If we can properly adjust the importance vector, for example, to promote positive patterns and suppress negative/random patterns, we can efficiently improve model performance.



$$T_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj}$$

Figure 7.1. Markov Chain Updates

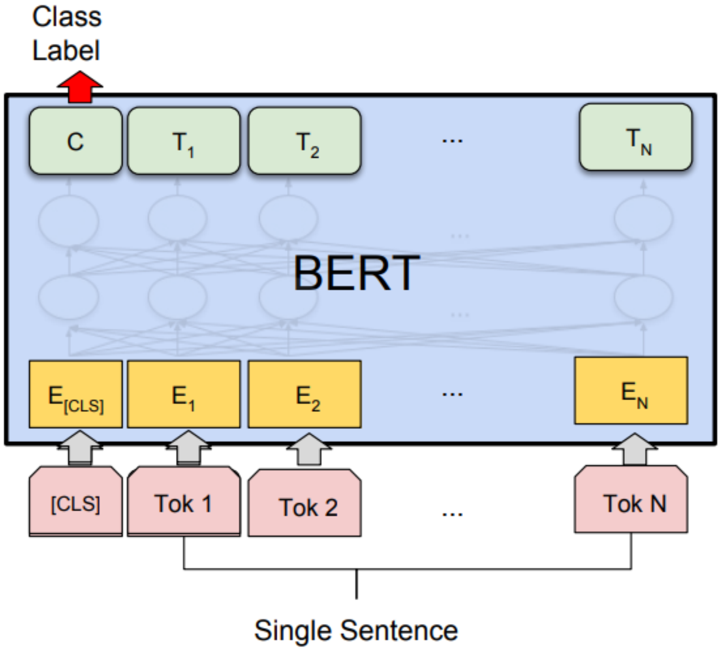


Figure 7.2. BERT Single Sentence Classification Network Architecture [59]

7.4.1 Model Structure

We propose Featured Transformer Methodology (FTM) to improve the performance of Transformer networks, especially on BERT network architecture. The idea is that we define patterns according to the domain knowledge, add patterns as arbitrary features to the input layer and use arbitrary features to promote the importance of corresponding terms on attention layer.

FTM works on four parts as shown below:

1. Input layer. We add domain specific patterns to the input so that these patterns can be used to present the high-level features which are defined according to domain knowledge. After adding these patterns to the input layer, they become part of the input. During model training, these patterns can be used to improve model performance. During prediction making, these patterns can contribute to the importance vector, especially for unlabeled terms that depend on these domain specific patterns to maximize the importance of the input sequences.

   Features can be defined in many ways, such as linguistic features and semantic features. For example, they can be linguistic features that include part of speech (POS) tags, suffix/prefix of the words and other. They can also be semantic features, such as stop words, a list of abbreviations, domain specific terms, and units of measurements.

   The selection of the features depends on the characteristics of the data. Additional features can efficiently fix the unlabeled data issue. However, too many additional features can cover up the characteristics of the data which may cause overfitting problem.
2. BERT Encoder layer. As shown in Figure 7.3, encoder layer consists of transformer blocks which include several layers. Each layer has two sub-layers with layer normalization: multi-head self-attention layer, pointwise and fully connected feed-forward network layer.

3. BERT Decoder layer. As shown in Figure 7.3, decoder has similar structure as the encoder. In addition to the two sub-layers, decoder inserts an attention layer to align encoder output to decoder.
4. Output layer. As shown in Figure 7.4, attention function maps a query and a set of key-value pairs to output. The outputs are computed as a weighted sum of the values where the weights are computed by compatibility function of the query with the corresponding keys.

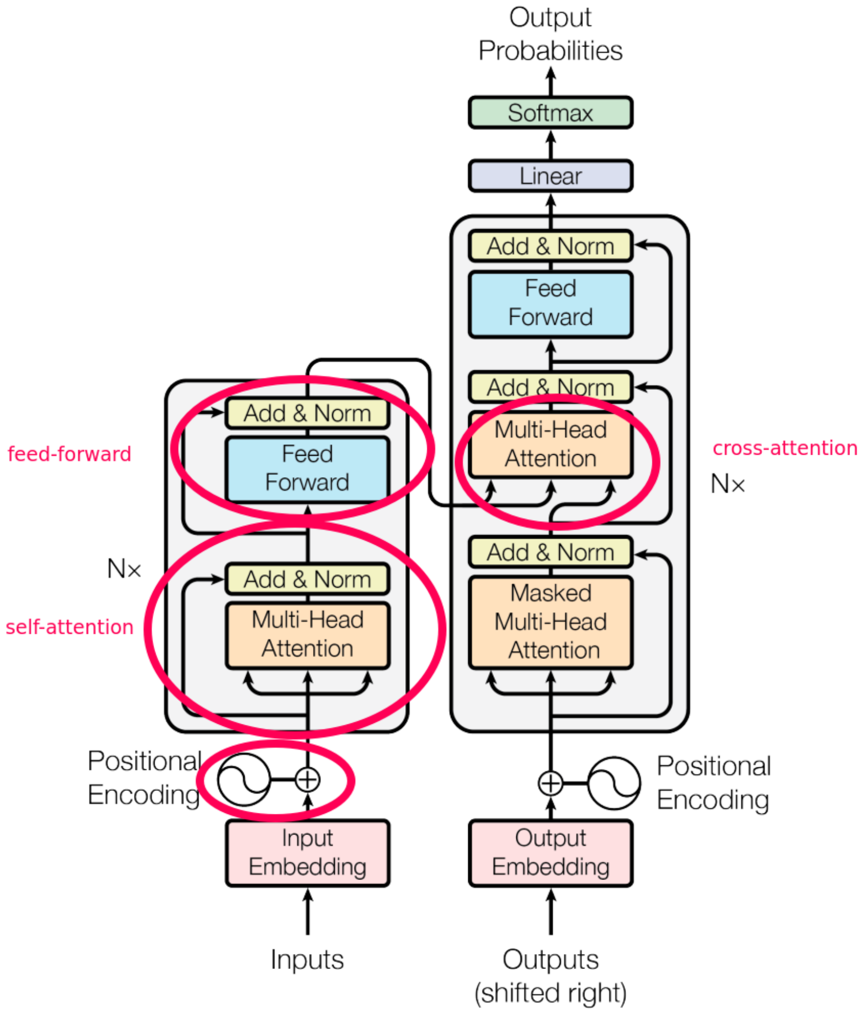


Figure 7.3. The Transformer Model Architecture. [244]

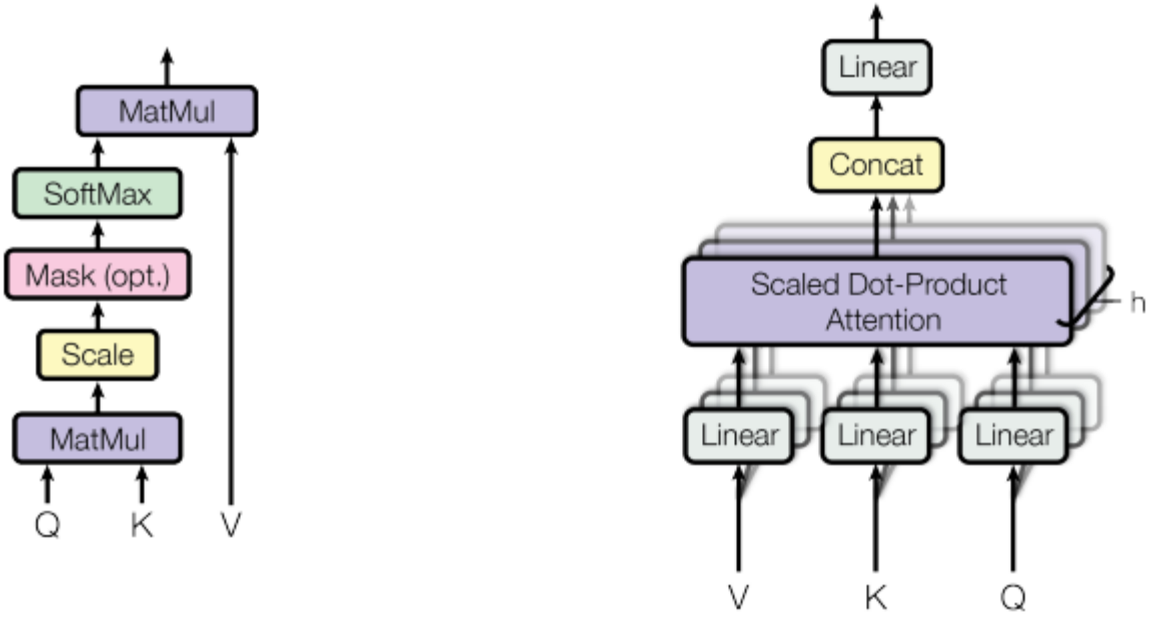


Figure 7.4. Attention Mechanism. [244]
Scaled Dot-Product Attention (left). Multi-Head Attention (right).

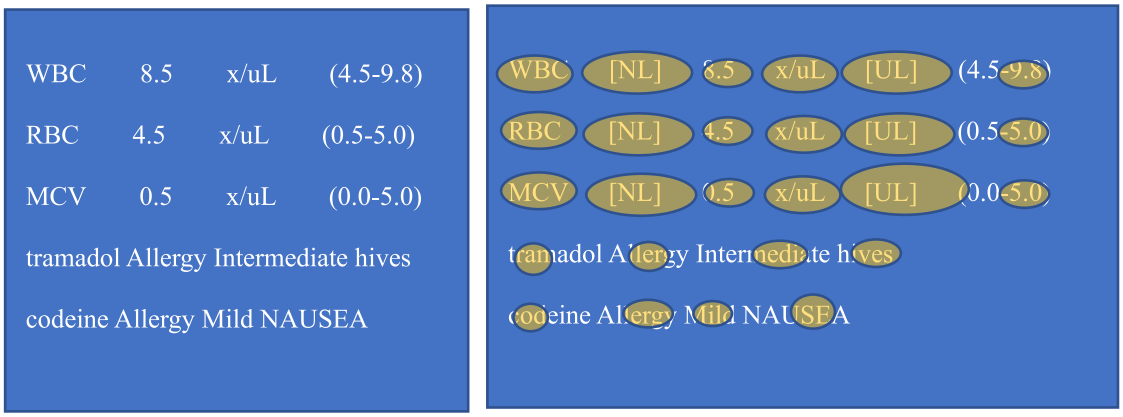
7.4.2 Input Layer

Let x be a input sequence of k words, denoted as $x_{1:k} = x_1, x_2, \ldots, x_k$. In BERT, an input sequence can be either one sequence or a pair of sequences separated by a special token [SEP]. We use one sequence input as an example to demonstrate how the arbitrary features are added.

In [270], auxiliary sentences are paired with the input to form sentence pairs so that the original classification can be converted into sentence-pair classification. Since the auxiliary sentences include higher level common features, the classification performance can be improved after the conversion.

However, our idea is not to increase the accuracy of pattern matching but to increase the importance of the feature dimensions so that we directly add arbitrary features to input sequences to maximize the importance of the positive samples.

We add unit entity feature ([UL]) and name entity feature ([NL]) after unit entities and name entities. After training, as shown in Figure 7.5, since unit entities and name entities appear very often, unit entities and name entities can draw large attention which can eventually promote the importance of corresponding samples. Especially, when some entities are missing during modeling training, in prediction process, since entity features of words have been trained and obtained large attention, the model can still count on name entity features ([NL]) and unit entity features ([UL]) to promote the importance of the input sequence.


WBC 8.5 x/uL (4.5-9.8)
RBC 4.5 x/uL (0.5-5.0)
MCV 0.5 x/uL (0.0-5.0)
tramadol Allergy Intermediate hives
codeine Allergy Mild NAUSEA
WBC [NL] 8.5 x/uL [UL] (4.5-9.8)
RBC [NL] 4.5 x/uL [UL] (0.5-5.0)
MCV [NL] 0.5 x/uL [UL] (0.0-5.0)
tramadol Allergy Intermediate hives
codeine Allergy Mild NAUSEA

(a) Original Text (b) After Adding Features

Figure 7.5. Importance of Attention Mechanism for Input Sequences.

7.4.3 BERT Encoder

After entity features are added to input sequences, for positive samples which have entities, entity features can gain large importance since they are common features and appear more often. For example, as shown in Figure 7.5(b), feature [NL] and feature [UL] are highly related, token *WBC/RBC/MCV* and feature [NL] are also highly related, token *x/uL* and feature [UL] are highly related.

7.4.4 BERT Decoder

Entity features can be added to decoder input sequences as well. These features can pick up the importance trained at encoder when attention layer aligns encoder to query sequences. When query sequences have entities unlabeled during encoding, the importance of the query can depend on entity features to get promoted. In other words, entity features can efficiently solve unlabeled feature issues.

7.4.5 Output Layer

The output layer is a softmax classifier on top of the attention layer. When the importance vector is tuned very well, the model can converge. During model training, the network architecture can be tuned through back-propagation and each layer can be tuned through auto regression.

7.5 Experiments

In all healthcare environments, documentation follows the Health Level Seven (HL7) Standard for electronic data exchange. For example, in the OBX segment, all the elements are defined to facilitate meeting the laboratory reporting requirements. There are limited number of fields in the OBX segment to support compliance. For unstructured lab project, we only use information in the fifth column which is called OBX.5.

In the field OBX.5, from different data sources, formats of text contents are coded in different ways. The formats can be related to section separations, paragraph separations, sentence separations, the formats of lists, the formats of interpretations, etc. For the purpose of building a sample set of OBX laboratory tests for machine learning, we need to convert different formats into one so that the sample set can be automatically parsed.

In the following, we use several examples to demonstrate how we format sentences, paragraphs, sections, lists, titles, and subtitles.

1. The number of columns: each document can have only one column.

   In Hutchinson raw messages, some documents have one column, but some have two columns separated by five or more spaces. We need to automatically read through the entire document to separate each line into two parts by using four or more spaces as the line-separator and then replace the line-separator with appropriate punctuation to convert the message into Comma Separated Values (csv).

2. Sections need to be in separate lines.

   In all sources, we can see a list of tests, notes, visits, etc. In the lists are separate items which are independent from each other in contents but can be related to one topic. For the lab test recognition task, we need to separate the list of tests into separate lines so that each test can be processed as one sample. In some sources, other than a list of tests, some lists can also have both similar contents and similar structures, such as a list of medication, notes, etc. So, for lists, we use a separate line for each item.

   For different sources, lists can be defined in different ways. For example, we can use indents, capital letters, numbers, different symbols as bullet points. We give several examples to show how the lists are defined.

3. Bullet points. Lists defined by bullet points can be structured well in raw messages. It may have sub-lists and several paragraphs in each item. When we parse the lists, we need to make sure each item and its contents are integrated together.

4. Numbers. When lists start with numbers, we also need to make sure each item and its contents are integrated together.

5. Capital letters. All capital letters. In Pratt Regional Medical Center, each message starts with the hospital header plus a list of contents which can be considered as separate sections.

Each section starts with the section topic in all capital letters, as shown in the following. In this example, there are two sections with the topics - ROS and IMPRESSION, respectively.

ROS:

I have reviewed all the other systems and they are negative.

IMPRESSION:

CARDIAC ARREST DUE TO OTHER UNDERLYING CONDITION - [I46.8]

In Kingman Healthcare Center, we can see the list starts with all capital letters followed by its contents with one space indent, as shown below,

FAMILY HISTORY

No significant family medical history.

ADDITIONAL NOTES

The nursing notes have been reviewed.

Figure 7.6. Example for Capital Letters as Headings

Reviewed Medications

Name: aspirin 81 mg tablet, delayed release Take 1 tablet(s) every day by oral route.

Date: 08/08/19 entered

Source: Aubry Boyce

Name: atorvastatin 80 mg tablet Take 1 tablet(s) every day by oral route for 90 days.

Date: 11/04/22 prescribed

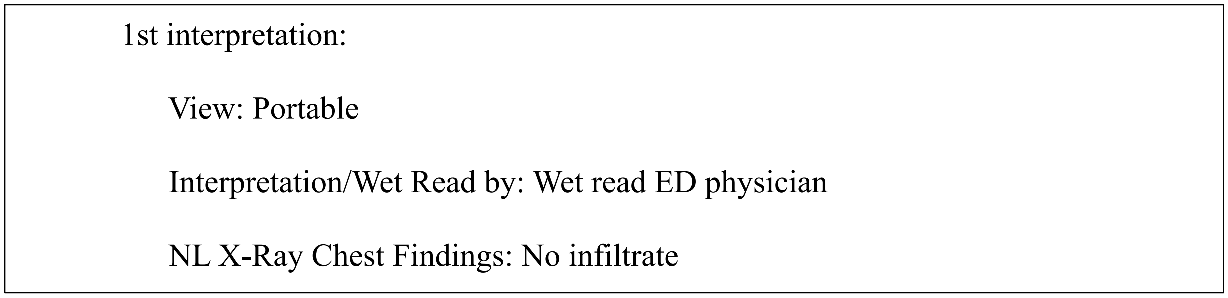
Source: Georges C. Elhomsy, MD

Figure 7.7. Example for First Letter of Word Upper as Headings

6. Text plus colon. This format is generally used in raw messages. Sometimes, the text can be all capital letters followed by colons, sometimes, the text can be in lower case. However, whenever we see text followed by colons, we know that is the start of the list. Several examples are listed below.

   As shown in figure 7.7, the first letter of word is upper case followed by colon. In WICHITA Diabetes, the list starts with words, each followed by colons in which the first letter of each word is uppercase, and the other letters are lowercase.

   As shown in figure 7.8, multiple levels. In Salina Regional Health Center, we have several levels of lists indicated by indents, as shown below. We convert multiple levels into one level because the items are independent from each other.



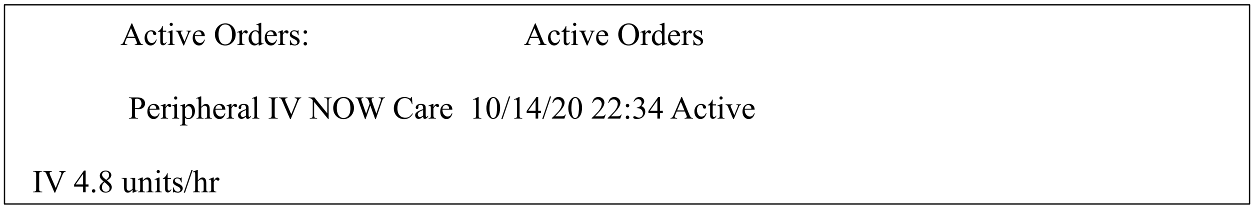
1st interpretation:

View: Portable

Interpretation/Wet Read by: Wet read ED physician

NL X-Ray Chest Findings: No infiltrate

Figure 7.8. Example for Multiple Level Headings



Active Orders: Active Orders

Peripheral IV NOW Care 10/14/20 22:34 Active

IV 4.8 units/hr

Figure 7.9. Example for Indents as Heading Indicators

7. Indents. Normally, indents and line spaces are used together to represent a list of items. For example, in Salina Regional Health Center, a list is defined as shown below. In this list, we can see sentence separators, such as comma, but, because there is a line space between two

items, we can then consider these items form a list.

8. Date/Time

   In Pratt Regional Medical Center, the lists can also start with a list of date/time. For example, in the following example, “VITALS HISTORY:” is defined as a list starting with date/time followed by test names, quantities and units. After parsing, the structure of the contents and the structures of the tests are similar, which can be useful in parameter tuning during model training, as shown in Figure 7.10.

VITALS HISTORY:

06/07/2022 17:33 Weight:63.6 kg(140 lb) Actual; Height:68 in(173 cm) BMI:21.25

06/07/2022 17:33 BP: 113/82 Arm (Automatic) MAP: 92.33 mmHG, Temp: 97.7 F Axillary, HR: 114 , Cardiac Rhythm: ST, RR: 42, O2 Sat: 98% 15 L/min via ETT - Bagged, Pain: 0/10 (Number scale) 17:40 BP: 121/83 Arm (Automatic) MAP: 95.67 mmHG, HR: 109 , Cardiac Rhythm: ST, RR: 36, O2 Sat: 100% via ETT - Ventilator

06/07/2022 17:45 GLASGOW COMA SCALE: E1V1M1 = GCS 3

06/8/2022 3:51 AM

Figure 7.10. Example for Date Time as Headings

9. Fields need to be separated by four or more spaces

   In laboratory tests, test results have a pre-defined structure which has several fields, such as test names, quantities, units, references, flags, etc. This structure can be used to define common patterns in data modeling to help extract lab sections. This structure normally forms an embedded table with several columns separated by multiple spaces. When the test results are not defined as embedded tables in the raw message, we need to re-arrange the test results to put them into this table structure so that each line can be one test, each column is an attribute, columns are separated by multiple spaces (four or more spaces). An example is given in Figure 7.11 to show before and after the structure is formed.

```
Before conversion,
Test
Result
Flag
Units
(Reference)
RBC
4.67
M/uL
(4.50 - 6.20)

After conversion,
Test    Result  Flag    Units   (Reference)
RBC     4.67            M/uL     (4.50 - 6.20)
```

Figure 7.11. Example for Line Spaces as Section Separators

10. Each sentence cannot be in two separate lines.

Sentences are the interpretations of the topic. For each topic, we put all the sentences under each topic into one line so that they can be in the same sample. Several indicators can be used to show us that some texts are interpretations, as shown in Figure 7.12.

Before conversion:

FINDINGS: Median sternotomy. Borderline cardiomegaly. Increased vascular prominence and bronchial wall thickening.

After conversion:

FINDINGS: Median sternotomy. Borderline cardiomegaly. Increased vascular prominence and bronchial wall thickening.

Figure 7.12. Example for Connecting Sentences

Dashes after the topic. in Figure 7.13, in Salina Regional Health Center, we join the lines,

Before conversion:

Free Text MDM Notes -

Patient continues to have severe DKA without any improvement in pH on arrival based on venous blood gas. Multiple attempts at peripheral IVs by staff, including ultrasound guidance were unsuccessful.

After conversion:

Free Text MDM Notes - Patient continues to have severe DKA without any improvement in pH on arrival based on venous blood gas. Multiple attempts at peripheral IVs by staff, including ultrasound guidance were unsuccessful.

Figure 7.13. Example for Dashes as Heading Indicators

11. Spaces at the beginning and the end of the sections need to be removed. After we code the raw message with the format we defined, each line can be one sample. The useful information for this project is the text so that we remove spaces before and after the text.

Before conversion:

Patient continues to have severe DKA without any improvement in pH on arrival based on venous blood gas.

After conversion:

Patient continues to have severe DKA without any improvement in pH on arrival based on venous blood gas.

Figure 7.14. Example for Removing Spaces at the Beginning and End of the Sections

Other than the format for the document structure, special symbols also need to be cleaned up in preprocessing.

System generated codes need to be removed, such as '\X0D\\X0A\','\R\\R\'.

Special symbols for HL7 format need to be replaced by spaces, such as '^','~','&'.

Special separators, such as '--','^Breakpoint','++++', need to be removed. If they are line separators, replace those with line spaces. If they are word separators, replace those with spaces.

Special formats for HL7 need to be removed. For example, for '20220612201600^YYYYMMDDHHMM', only keep '20220612201600'.

For sample selection, since we want to extract all lab tests (which are about 522 tests) from raw messages, we want to include as many lab tests as possible in the sample set. First, we built lab test dictionary that has all the terms we can extract from the raw messages. Sample selection stage can collect samples for each of the terms for model training. Each term needs between three to five samples to properly define features related to the term. If a term has less than three samples, the term cannot be accurately extracted by the trained model. If a term has more than five samples, the sample set is over prepared.

Term frequency can be used to represent how many samples each term has. Sample frequency is defined as how many terms have a certain number of samples. Samples are collected in two rounds. We combine the two rounds together for sample selection purposes. As shown below, we use Figures 7.15 to 7.18 to present the distribution of term frequency and sample frequency.

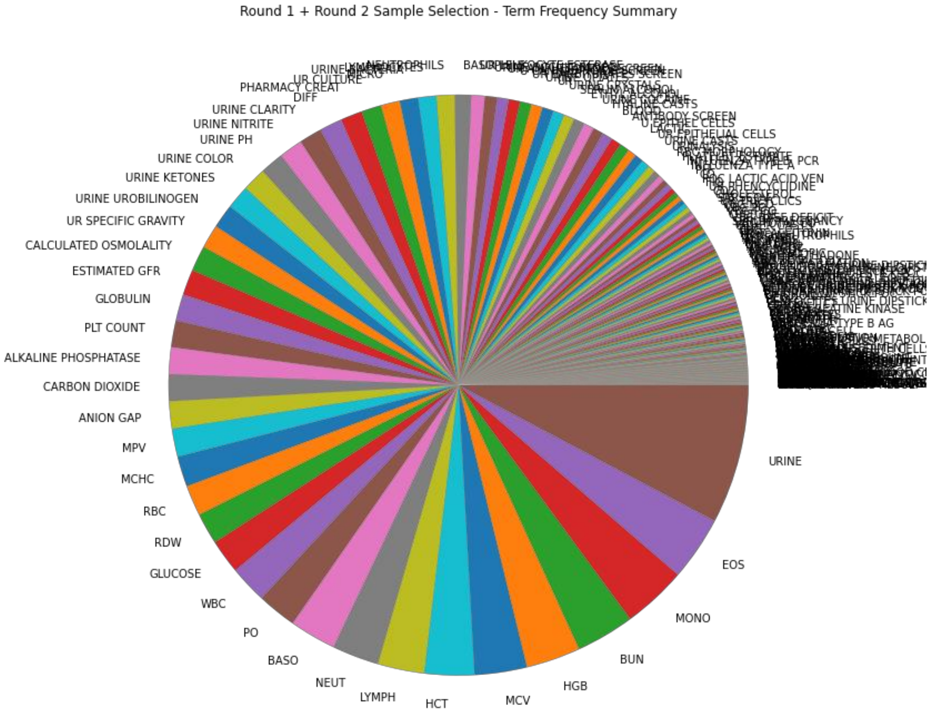


Figure 7.15. Term Frequency Pie Chart for Lab Test Extraction Project

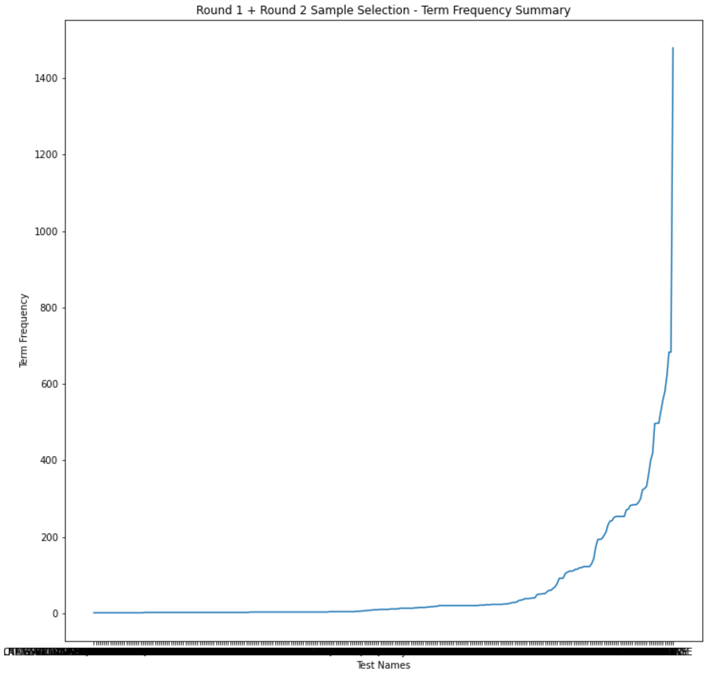


Figure 7.16. Term Frequency Line Chart for Lab Test Extraction Project

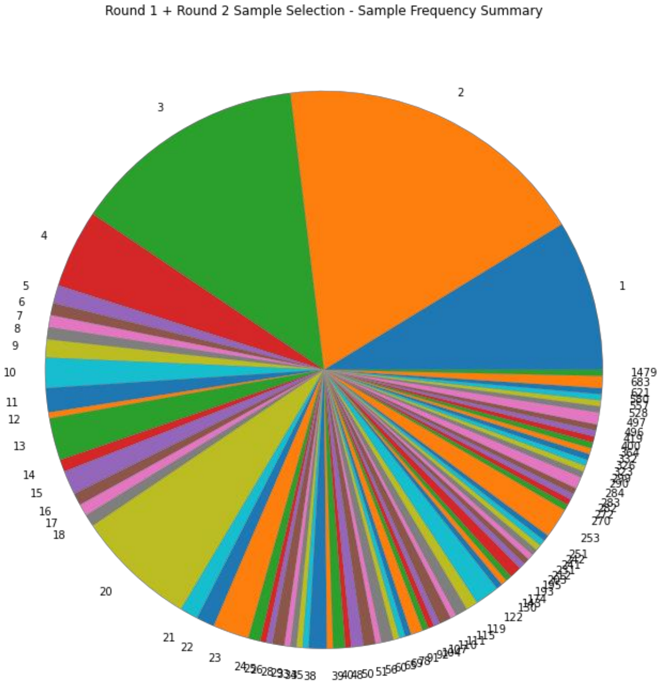


Figure 7.17. Sample Frequency Pie Chart for Lab Test Extraction Project

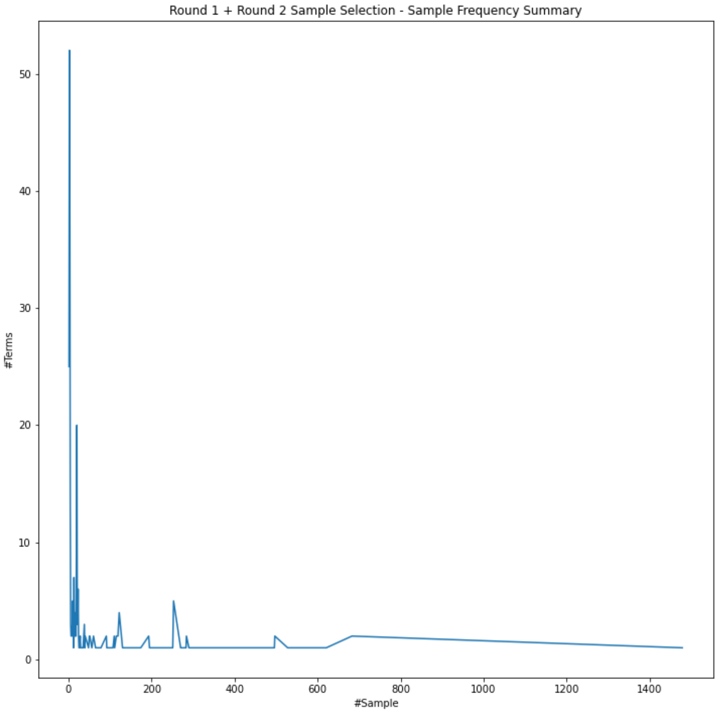


Figure 7.18. Sample Frequency Line Chart for Lab Test Extraction Project

The model was tested with 2871 samples in which 788 samples were labeled as 'lab' and 2083 samples were labeled as 'doc'. As shown in the following table, the experimental results showed that the performance measurements, such as precision, recall, and F1-score, indicated that the model can be used to identify lab tests with 100% in all measurements. In comparison with the Conditional Random Fields (CRF) model, Bi-directional Encoder Representation Transformer (BERT), Naïve Bayes Support Vector Machine (NBSVM), Logistic Regression (LOGREG), FASTTEXT, Standard Gated Recurrent Units (STANDARD GRU), Bi-directional Gated Recurrent Units (BiGRU), Featured Transformer Methodology (FTM) can perform much better than CRF, BERT, NBSVM, LOGREG, FASTTEXT, STANDARD GRU, and BiGRU in all measurements.

Table 7.1. Performance of BERT-Featured on 2871 Samples

	Precision	Recall	F1-score
O	1	1	1
LAB	1	1	1
Accuracy	1		
Macro Avg	1	1	1
Weighted Avg	1	1	1

Table 7.2. Performance of CRF on 2871 Samples

	Precision	Recall	F1-score
O	0.927	0.989	0.957
LAB	0.959	0.764	0.85
Accuracy	0.933		
Macro Avg	0.943	0.876	0.904
Weighted Avg	0.935	0.933	0.931

Table 7.3. Performance of BERT on 2871 Samples

	Precision	Recall	F1-score
O	0.99	0.99	0.99
LAB	0.99	0.99	0.99
Accuracy	0.99		
Macro Avg	0.99	0.99	0.99
Weighted Avg	0.99	0.99	0.99

Table 7.4. Performance of NBSVM on 2871 Samples

	Precision	Recall	F1-score
O	0.92	0.98	0.95
LAB	0.99	0.97	0.98
Accuracy	0.97		
Macro Avg	0.96	0.97	0.96
Weighted Avg	0.97	0.97	0.97

Table 7.5. Performance of LOGREG on 2871 Samples

	Precision	Recall	F1-score
O	0.92	0.96	0.94
LAB	0.98	0.97	0.97
Accuracy	0.96		
Macro Avg	0.95	0.96	0.96
Weighted Avg	0.96	0.96	0.96

Table 7.6. Performance of FASTTEXT on 2871 Samples

	Precision	Recall	F1-score
O	0.97	0.97	0.97
LAB	0.99	0.99	0.99
Accuracy	0.98		
Macro Avg	0.98	0.98	0.98
Weighted Avg	0.98	0.98	0.98

Table 7.7. Performance of STANDARD GRU on 2871 Samples

	Precision	Recall	F1-score
O	0.97	0.94	0.95
LAB	0.98	0.99	0.98
Accuracy	0.997		
Macro Avg	0.97	0.96	0.97
Weighted Avg	0.97	0.97	0.97

Table 7.8. Performance of Bi-GRU on 2871 Samples

	Precision	Recall	F1-score
O	0.98	0.98	0.98
LAB	0.99	0.99	0.99
Accuracy	0.99		
Macro Avg	0.99	0.99	0.99
Weighted Avg	0.99	0.99	0.99

7.6 Conclusion

We proposed a way to improve Transformer architecture by adding arbitrary features to input sequences. According to the Markov-like updates, we use arbitrary features to influence the importance vector of the attention layers. This methodology can help combine domain knowledge with the input sequences, efficiently promote the importance of the domain specific patterns, and fix the unlabeled data issue. We reviewed state-of-the-art solutions in Transformer architecture, present the association between Markov graph and attention mechanism, and introduced how to implement the methodology in BERT architecture on different layers, such as input layer, encoder and decoder layer, attention layer, and output layer. We compared the new methodology - Featured Transformer with several most recent research, such as CRF which also uses arbitrary features for classification, BERT, NBSVM, FASTTEXT, STANDARD GRU, and Bi-GRU. Experimental results showed us that, according to several performance measurements, such as precision, recall, F1-score, accuracy, and micro average and weighted average precision, recall and F1-score. Featured Transformer performs better than other methodologies in text classification. In future, we want to add lexical and semantic features and evaluate how much arbitrary features can influence data modeling.

Chapter 8: Conclusion

With the development of social networks, human society can be presented in virtual world through the Internet. A collection of work has been done during my Ph.D. studies with the goal of building expressive structured models that not only model individual data points but also connections between data points. The challenges we are facing are related to the size of data we need to use, the performance and limitation of modern computing machines for this study, the expressiveness of the models, and the potentials in improving state-of-the-art theories to solve new issues.

In Chapter 2, we reviewed the related work in Graphic Processing Units (GPU), GPU optimization, parallel computation and optimization, parallel join and query operation, graph processing with GPU, graph store task scheduling and parallel planning, performance ratio, bin packing algorithms, sampling, triangle computation, and community detection.

In Chapter 3, we defined symbols and explanations for the entire dissertation.

In Chapter 4, we discussed the upper bound and lower bound of the space cost. We used a snapshot of the data stream as a case study to show the boundaries of the bin packing. We proved that packing stream I into C bins is NP-Complete, the time complexity for bin packing is O(NlogN) for sorted data, and O(N) for unsorted data. For comparison-based data structure, the time complexity cannot be O(εlogN). The upper bound and lower bound of the bin packing can be O(logN) and (1+ε)OPT(I) respectively. The upper bound and the lower bound of the multi-variable bin packing are max(SIZE(I_1),.., SIZE(I_k)), and max(OPT(I_1), …, OPT(I_k)), respectively. The capacity of the largest bin C_{max} and the smallest bin C_{min} satisfy $C_{max} - C_{min} < C$ in which $a_1 - a_n <$

C and $a_1 > a_2 > \ldots > a_n$. For a multiset A of positive integers with total sum S, the problem of deciding whether there exists a subset of sum S/k is NP-Complete and the upper bound and lower bound of the time complexity is OPT(I) and OPT(I)logOPT(I).

To shrink the data set size, in Chapter 5, we worked on how to shrink the data set size through sampling technologies. Two methods were proposed to make this happen: one is based on Kronecker graph double cover, the other is based on graph curvature. Kronecker graph double cover theory and Curvature theory have been defined and proved in graph theories.

According to the symmetric structure of nodes and edges in graphs, Kronecker graph double cover can recursively decompose and reconstruct graphs and retain graph topology properties during these operations. To keep the connectedness during sampling processes, our methodology chooses to merge similar vertices and edges instead of removing or rebuilding vertices and edges. We theoretically explain the characteristics of the reconstructible graphs, how to factorize, decompose and reconstruct graphs with the topology properties unchanged, how to keep the connectedness during graph reconstruction, and how to conduct Kronecker double-cover operation and obtain the graph product. We also theoretically prove that Kronecker graph has limited graphons, Kronecker graph can be used to generate any binary graph with no self-loops and multiple edges, Kronecker graph properties hold during projection, and the Homomorphism density of Kronecker graphs. The sampling process can be through either top-down or bottom-up. We applied data clustering on three ground truth data sets to prove that, after sampling, graph data quality was improved.

Several curvature definitions, such as Forman curvature, Menger-Ricci curvature, and Haantjes-Ricci curvature, were discussed and applied to graph sampling. Curvature theory was defined in differential geometry in a network. Curvature describes how much the curve direction changed over a small distance. It demonstrated the evolution of the curves. In community detection, we can use curvature to simulate the evolution of the community, in which the influence of the community centers was based on the distance between the community and the observed vertices. The community can be considered as the snapshot of a particular moment during the evolution of the community. Different curvature formulas, such as Forman curvature, Menger-Ricci curvature, and Haantjes-Ricci curvature, were the driving force of the changes. We used three ground truth data sets to perform clustering and used results to evaluate the quality of the sample sets. Experimental results showed us that the sample sets can keep the topology properties and have better quality than the original data sets.

To conduct semantic entity analysis on social networks, we defined a feature extraction methodology for CRF models. We discussed different ways to define feature sets that can help extract domain specific patterns from texts. Conditional Random Fields models belonged to Markov models by losing the constraints in independence of the states. Associations between input sequences play important roles in data modeling. We summarized the ways to define feature sets, such as lexical features and semantic features. We also summarized the ways to define feature functions, such as binary functions and feature functions. We used packaging information extraction project as a case study to demonstrate the strength and weaknesses in feature sets and feature functions and discuss solutions to solve these issues. We compared the performance of our methodology with Bi-directional Long Short-Term Memory (Bi-LSTM) algorithm. The experimental results showed that, according to precision, recall, and F1-score, our methodology performed better than Bi-LSTM. Other than these measurements, when we checked the quality of the results, we can see that our methodology can ensure that the segments extracted from the input sequences are complete. On the other hand, the segments extracted by Bi-LSTM can be cut into disconnected pieces that cannot guarantee the completeness of the results and cannot be fixed during post-processing. The improved CRF with domain specific feature sets can efficiently extract domain specific segments from input sequences, and final results were meaningful and can be further improved through post-processing by using domain specific knowledge.

To conduct semantic classification on social networks, we improved Transformer architecture by adding arbitrary features to influence Markov-like updates on the importance vector in attention mechanism. This methodology can efficiently solve the unlabeled feature issues, combine domain knowledge with data models and promote the associations between featured terms and featured patterns. To propose the new idea, we surveyed related network architectures, such as Feed-Forward Network (FFN) architecture, Recurrent Neural Networks (RNN), attention mechanism, and the existing ways to combine domain knowledge with the neural network architecture. In terms of computation complexity, recurrent neural network architecture required more computation, but are efficient at sequence analysis because conditional probability can be computed through the encoder-decoder structure. Feedforward network architecture had low computation costs but cannot be able to discover the associations between entities. Attention mechanism can overcome the issue in feedforward network architecture by adding attention layer to the feedforward network architecture. The state-of-the-art solutions to combine domain knowledge with Transformer architecture were by converting single sequence analysis problems into sequence comparison problems. The performance can be improved by having more inputs. These solutions cannot influence the weights of the feature entities in the importance vector so that they cannot quantitively improve the performance of the classification models. Our methodology can add high level features to the input sequences and eventually influenced the results in the importance vector so that the accuracy in prediction can be improved. Experimental results also showed that our methodology outperformed popular classification models, such as Conditional Random Fields (CRF), Naive Bayesian Support Vector Machine (NBSVM), Standard Gated Recurrent Units (Standard GRU), Bi-directional Gated Recurrent Units (Bi-GRU), and Logistic regression.

In the future, we will continue to develop expressive models by combining domain knowledge to either machine learning models or deep learning models. Machine learning models can be more expressive by doing Markov-like updates. However, the definition of the feature set can efficiently influence the results. The feature sets can be defined with both lexical and semantic features. The big challenges are how to select those features and how many features we need to better fine tune the parameters. For deep learning, attention mechanisms can convert auto-regression updates into Markov-like updates and the importance vector can be considered as the snapshot of the context. However, it is even more difficult to measure how much domain specific knowledge can be added to the inputs and how to use domain specific patterns to fine tune the importance vector, but not to make changes to the distribution of the original data.

References

[1] Abacha, A.B., Chowdhury, M.F.M., Karanasiou, A., Mrabet, Y., Lavelli, A., & Zweigenbaum, P. (2015), Text Mining for Pharmacovigilance: Using Machine Learning for Drug Name Recognition and Drug–Drug Interaction Extraction and Classification, Journal of Biomedical Informatics, Volume 58, 122-132.

[2] Ahmed, N.K., Neville, J., & Kompella, R. (2013). Network Sampling: from Static to Streaming Graphs. ACM Transactions on Knowledge Discovery from Data, Volume 8, Issue 2, Article 7, 1–56. https://doi.org/10.1145/2601438.

[3] Akhtar, S., Basile, V., & Patti, V. (2020). Modeling Annotator Perspective and Polarized Opinions to Improve Hate Speech Detection. Proceedings of the AAAI Conference on Human Computation and Crowdsourcing, Volume 8, Issue 1, 151-154.

[4] Akilan, A. (2015). Text Mining: Challenges and Future Directions, 2015 2nd International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, 1679-1684. doi: 10.1109/ECS.2015.7124872.

[5] Albert, R. & Barabasi, A. (2002). Statistical Mechanics of Complex Networks. Reviews of Modern Physics. Volume 74. 47-97.

[6] Anand, V., Mehrotra, P., Margo, D.W., & Seltzer, M.I. (2020). Smooth Kronecker: Solving the Combing Problem in Kronecker Graphs. Proceedings of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA).

[7] Andersen, R., Chung, F., & Lang, K. (2006). Local Graph Partitioning Using PageRank Vectors, 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), Berkeley, CA, USA, 2006. 475-486, doi: 10.1109/FOCS.2006.44.

[8] Andersen, R., & Peres, Y. (2009). Finding Sparse Cuts Locally Using Evolving Sets. Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. 235–244. https://doi.org/10.1145/1536414.1536449.

[9] Appelt, D.E., Hobbs, J.R., Bear, J., Israel, D., Kameyama, M., Martin, D., Myers, K., & Tyson, M. (1995). SRI International FASTUS System: MUC-6 Test Results and Analysis. Proceedings of the 6th Conference on Message Understanding. 237-248. https://doi.org/10.3115/1072399.1072420.

[10] Applegate, D.L., Buriol, L.S., Dillard, B., Johnson, D.S., & Shor, P.W. (2003). The Cutting-Stock Approach to Bin Packing: Theory and Experiments. Workshop on Algorithm Engineering and Experimentation. http://bernarddillard.com/wp-content/uploads/2015/10/Cutting-Stock-Approach.pdf.

[11] Arifuzzaman, S., Khan, M., & Marathe, M. (2013). PATRIC: A Parallel Algorithm for Counting Triangles in Massive Networks. Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13). Association for Computing Machinery, New York, NY, USA, 529–538. https://doi.org/10.1145/2505515.2505545.

[12] Armant, V., De Cauwer, M. Brown, K. N. & O'Sullivan, B. (2018). Semi-Online Task Assignment Policies for Workload Consolidation in Cloud Computing Systems. Future Generation Computer Systems, Volume 82, 89-103. doi:10.1016/J.Future.2017.12.035.

[13] Avery, C. (2011). Giraph: Large-Scale Graph Processing Infrastructure on Hadoop. Proceedings of the Hadoop Summit. Santa Clara, Volume 11, Issue 3, 5-9.

[14] Azadegan, S. & Tripathi, A.R. (1997). A parallel join algorithm for SIMD architectures. Journal of Systems and Software Volume 39, Issue 3, 265-280.

[15] Azar, Y., Cohen, I.R., Kamara, S., & Shepherd, B. (2013). Tight Bounds for Online Vector Bin Packing. Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13). Association for Computing Machinery, New York, NY, USA, 961–970. https://doi.org/10.1145/2488608.2488730.

[16] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473.

[17] Balogh, J., Békési, J., Dósa, G., Epstein, L., & Levin, A. (2017). A New and Improved Algorithm for Online Bin Packing. Embedded Systems and Applications. arXiv:1707.01728.

[18] Bansal, N., Eliás, M., & Khan, A. (2016). Improved Approximation for Vector Bin Packing. Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). 1561-1579. doi: 10.1137/1.9781611974331.ch106.

[19] Bansal, N., Vredeveld, T., & Van Der Zwaan, R. (2014). Approximating Vector Scheduling: Almost Matching Upper and Lower Bounds. In: Pardo, A., Viola, A. (Eds) LATIN 2014: Theoretical Informatics. LATIN 2014. Lecture Notes in Computer Science, Volume 8392. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-54423-1_5.

[20] Batu, T., Berenbrink, & P., Sohler, C. (2009). A Sublinear-time Approximation Scheme for Bin Packing. Theoretical Computer Science. Volume 410. 47-49, 6 November 2009, 5082–5092. https://doi.org/10.1016/j.tcs.2009.08.006.

[21] Bar-Yossef, Z., Kumar, R., & Sivakumar, D. (2002). Reductions in Streaming Algorithms with an Application to Counting Triangles in Graphs. Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms. 623-632.

[22] Beamer, S., Asanovic, K., & Patterson, D. (2012). Direction-Optimizing Breadth-First Search, SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, 1-10, doi: 10.1109/SC.2012.50.

[23] Becchetti, L., Boldi, P., Castillo, C., & Gionis, A. (2008). Efficient Semi-Streaming Algorithms for Local Triangle Counting in Massive Graphs. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 16–24. https://doi.org/10.1145/1401890.1401898.

[24] Beigel, R. & Fu, B. (2012). A Dense Hierarchy of Sublinear Time Approximation Schemes for Bin Packing. Frontiers in Algorithmics and Algorithmic Aspects in Information and Management, 172–181. Springer, New York.

[25] Bell, N., & Garland, M. (2009). Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors. Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Portland, OR, USA, 1-11, doi: 10.1145/1654059.1654078.

[26] Belletti, F., Anderson, J., Lakshmanan, K., Mayoraz, N., Kanwar, P., Robie, T., Oguntebi, T., Krichene, W., & Chen, Y. (2019). Randomized Fractal Expansions for Production-Scale Public Collaborative-Filtering Data Sets. arXiv:1905.09874.

[27] Belletti, F., Lakshmanan, K., Krichene, W., Chen, Y., & Anderson, J.R. (2019). Scalable Realistic Recommendation Datasets through Fractal Expansions. arXiv, Abs/1901.08910.

[28] Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. Journal of Machine Learning Research, Volume 3. 1137–1155.

[29] Berry, J.W., Fostvedt, L.K., Nordman, J.D., Phillips, C.A., Seshadhri, C., & Wilson, A.G. (2014). Why Do Simple Algorithms for Triangle Enumeration Work in the Real World? Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS '14). Association for Computing Machinery, New York, NY, USA, 225–234. https://doi.org/10.1145/2554797.2554819

[30] Besta, M., Podstawski, M., Groner, L., Solomonik, E., & Hoefler, T. (2017). To Push or to Pull: on Reducing Communication and Synchronization in Graph Computations. Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '17). Association for Computing Machinery, New York, NY, USA, 93–104. https://doi.org/10.1145/3078597.3078616

[31] Betzler, N., Van Bevern, R., Fellows, M.R., Komusiewicz, C., & Niedermeier, R. (2011). Parameterized Algorithmics for Finding Connected Motifs in Biological Networks. IEEE/ACM Transactions on Computational Biology Bioinformatics. Volume 8, Issue 5, 1296-1308. doi: 10.1109/TCBB.2011.19. PMID: 21282862.

[32] Bhuiyan, M.A., Rahman, M., Rahman, M., & Hasan, M.A. (2012). GUISE: Uniform Sampling of Graphlets for Large Graph Analysis, IEEE 12th International Conference on Data Mining, Brussels, Belgium, 2012, 91-100, doi: 10.1109/ICDM.2012.87.

[33] Bianchini, M., Gori, M., & Scarselli, F. (2002). Recursive Processing of Cyclic Graphs. Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290), Honolulu, HI, USA, 2002, Volume 1, 154-159. doi: 10.1109/IJCNN.2002.1005461.

[34] Bikel, D.M., Schwartz, R.M., & Weischedel, R.M. (1999). An Algorithm that Learns What's in a Name. Machine Learning, Volume 34, 211-231.

[35] Brandes, U. (2001). A Faster Algorithm for Betweenness Centrality. Journal of Mathematical Sociology, Volume 25, 163 - 177.

[36] Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., & Shah, R. (1993). Signature Verification Using A "Siamese" Time Delay Neural Network. Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS'93). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 737–744.

[37] Bullmore, E., & Sporns, O. (2009). Complex Brain Networks: Graph Theoretical Analysis of Structural and Functional Systems. Nature Reviews. Neuroscience, Volume 10, Issue 3, 186–198. https://doi.org/10.1038/Nrn2575

[38] Burt, R. S. (2004). Structural Holes and Good Ideas. American Journal of Sociology, Volume 110, Issue 2, 349–399. https://doi.org/10.1086/421787

[39] Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., & Vigna, S. (2006). A Reference Collection for Web Spam. SIGIR Forum, Volume 40, Issue 2, 11–24. https://doi.org/10.1145/1189702.1189703

[40] Chakrabarti, D., Zhan, Y., & Faloutsos, C. (2004). R-MAT: A Recursive Model for Graph Mining. SDM 2004. 442-446

[41] Che, S., Sheaffer, J.W., & Skadron, K. (2011). Dymaxion: Optimizing Memory Access Patterns for Heterogeneous Systems, SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, Seattle, WA, USA, 1-11.

[42] Chekuri, C., & Khanna, S. (2004). On Multidimensional Packing Problems. SIAM Journal of Computing. Volume 33, Issue 4, 837–851.

[43] Chen, L., Jansen, K., & Zhang, G. (2014). On the Optimality of Approximation Schemes for the Classical Scheduling Problem. Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics. 657-668.

[44] Chen, R., Shi, J., Chen, Y., Zang, B., Guan, H., & Chen, H. (2019). PowerLyra: Differentiated Graph Computation and Partitioning on Skewed Graphs. ACM Transactions on Parallel Computing, Volume 5, Issue 3, Article 13, 1-39. https://doi.org/10.1145/3298989.

[45] Cheng, J., Dong, L., & Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 551–561, Austin, Texas. Association for Computational Linguistics.

[46] Chiang, J.H. & Yu, H.C. (2005). Literature Extraction of Protein Functions Using Sentence Pattern Mining. IEEE Transactions on Knowledge and Data Engineering, Volume 17, Issue 8, 1088-1098, doi: 10.1109/TKDE.2005.132.

[47] Chiba, N., & Nishizeki, T. (1985). Arboricity and Subgraph Listing Algorithms. SIAM Journal of Computing, Volume 14, 210-223.

[48] Christensen, H.I., Khan, A., Pokutta, S., Tetali, P. (2017). Approximation and Online Algorithms for Multidimensional Bin Packing: A Survey. Computer Science Review, Volume 24, 63-79, ISSN 1574-0137, https://doi.org/10.1016/J.Cosrev.2016.12.001.

[49] Chu, S. & Cheng, J. (2011). Triangle Listing in Massive Networks and Its Applications. Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11). Association for Computing Machinery, New York, NY, USA, 672–680. https://doi.org/10.1145/2020408.2020513.

[50] Cieslewicz, J., Berry, J., Hendrickson, B. & Ross, K.A. (2006). Realizing Parallelism in Database Operations: Insights from a Massively Multithreaded Architecture. Proceedings of the 2nd International Workshop on Data Management on New Hardware. (DaMoN ’06). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1140402.1140408.

[51] Coffman, E.G., Csirik, J.A., Galambos, G., Martello, S., & Vigo, D. (2013). Bin Packing Approximation Algorithms: Survey and Classification. Handbook of Combinatorial Optimization. 455-531. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7997-1_35.

[52] Cohen, J. (2009). Graph Twiddling in a MapReduce World. Computing in Science & Engineering, Volume 11, Issue 4, 29-41, doi: 10.1109/MCSE.2009.120.

[53] Cormode, G., & Veselý, P. (2019). Streaming Algorithms for Bin Packing and Vector Scheduling. Theory of Computing Systems, Volume 65, 916 - 942.

[54] Crauser, A., Mehlhorn, K., Meyer, U., Sanders, P. (1998). A Parallelization of Dijkstra's Shortest Path Algorithm. Brim, L., Gruska, J., Zlatuška, J. (Eds) Mathematical Foundations of Computer Science 1998. MFCS 1998. Lecture Notes in Computer Science, Volume 1450. 722-731. Springer, Berlin, Heidelberg. https://doi.org/10.1007/Bfb0055823

[55] Csirik, J., Johnson, D.S., Kenyon, C., Orlin, J.B., Shor, P.W., & Weber, R.R. (2006). On the Sum-Of-Squares Algorithm for Bin Packing. Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing. 208-217.

[56] Cunningham, H. (2002). GATE, a General Architecture for Text Engineering. Computers and the Humanities, Volume 36, Issue 2, 223-254. http://www.jstor.org/stable/30204529.

[57] Davidson, A., Baxter, S., Garland, M. & Owens, J.D. (2014). Work-Efficient Parallel GPU Methods for Single-Source Shortest Paths. Parallel and Distributed Processing Symposium, 2014 IEEE 28th International. 349-359.

[58] Dementiev, R. (2006). Algorithm Engineering for Large Data Sets, PhD Thesis, Saarland University. http://algo2.iti.kit.edu/documents/DementievDiss.pdf

[59] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv, abs/1810.04805.

[60] Dieng, A.B., Wang, C., Gao, J., & Paisley, J.W. (2016). TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency. arXiv, Abs/1611.01702.

[61] Domingos, P., and Richardson, M. (2001). Mining the Network Value of Customers. in Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01). Association for Computing Machinery, New York, NY, USA, 57-66. https://doi.org/10.1145/502512.502525

[62] Dósa, G., & Sgall, J. (2013). First Fit Bin Packing: A Tight Analysis. STACS 2013. 538-549.

[63] Dreyfus, S.E. (1990). Artificial Neural Networks, Back Propagation, and the Kelley-Bryson Gradient Procedure. Journal of Guidance Control and Dynamics, Volume 13, 926-928.

[64] Dumais, S. T., Furnas, G. W., Landauer, T. K. & Deerwester, S. (1988). Using Latent Semantic Analysis to Improve Information Retrieval. Proceedings of CHI'88: Conference on Human Factors in Computing, New York: ACM, 281-285.

[65] Eckmann, J.P., & Moses, E. (2002). Curvature of Co-Links Uncovers Hidden Thematic Layers. in the World Wide Web, PNAS, Volume 99, Issue 9, 5825-5829. https://doi.org/10.1073/pnas.03209339

[66] Eke, C.I., Norman, A.A., & Shuib, L. (2021). Context-Based Feature Technique for Sarcasm Identification in Benchmark Datasets Using Deep Learning and BERT Model. IEEE Access, Volume 9, 48501-48518.

[67] Elenberg, E.R., Shanmugam, K., Borokhovich, M., & Dimakis, A.G. (2015). Beyond Triangles: A Distributed Framework for Estimating 3-Profiles of Large Graphs. Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15). Association for Computing Machinery, New York, NY, USA, 229-238. https://doi.org/10.1145/2783258.2783413

[68] Elenberg, E.R., Shanmugam, K., Borokhovich, M., & Dimakis, A.G. (2016). Distributed Estimation of Graph 4-Profiles. Proceedings of the 25th International Conference on World Wide Web (WWW '16). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 483-493.

[69] Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On Power-Law Relationships of the Internet Topology. SIGCOMM Computer Communication Review. Volume 29, No. 4, 251-262. https://doi.org/10.1145/316194.316229.

[70] Farzan, M., & Waller, D.A. (1977). Kronecker Products and Local Joins of Graphs. Canadian Journal of Mathematics, Volume 29, 255-269.

[71] Faust, K. (2010). A Puzzle Concerning Triads in Social Networks: Graph Constraints and the Triad Census. Social Networks, Volume 32, Issue 3, 221-233.

[72] Fernandez De La Vega, W., & Lueker, G.S. (1981). Bin Packing Can Be Solved Within 1 + E in Linear Time. Combinatorica, Volume 1, 349-355.

[73] Fleischer, R., & Wahl, M.G. (2000). Online Scheduling Revisited. Embedded Systems and Applications. https://www.cs.toronto.edu/~bor/2421f19/papers/fleischer-best-makespan.pdf.

[74] Forman, R. (2003). Bochner's Method for Cell Complexes and Combinatorial Ricci Curvature. Discrete & Computational Geometry, Volume 29, 323-374.

[75] Freitag, D., & McCallum, A. (1999). Information Extraction with HMMs and Shrinking. Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction. 31-16.

[76] Fukuda, K., Tamura, A., Tsunoda, T., & Takagi, T. (1998). Toward Information Extraction: Identifying Protein Names from Biological Papers. Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, 707-718.

[77] Garey, M.R., Graham, R.L., Johnson, D.S., & Yao, A.C. (1976). Resource Constrained Scheduling as Generalized Bin Packing. Journal of Combinatorial Theory, Series A, Volume 21, Issue 3, 257-298.

[78] Geetha, M.P., Renuka, D.K. (2021). Improving the Performance of Aspect Based Sentiment Analysis Using Fine-Tuned BERT Base Uncased Model. International Journal of Intelligent Networks, Volume 2, 64-69.

[79] Gers, F.A., Schmidhuber, J. & Cummins, F. (1999). Learning to Forget: Continual Prediction with LSTM. 9th International Conference on Artificial Neural Networks ICANN 99. Edinburgh, UK, Volume 2, 850-855, doi: 10.1049/cp:19991218.

[80] Gers, F.A., & Schmidhuber, E. (2001). LSTM Recurrent Networks Learn Simple Context-Free and Context-Sensitive Languages, IEEE Transactions on Neural Networks, Volume 12, No. 6, 1333-1340, doi: 10.1109/72.963769.

[81] Gharaibeh, A., Costa, L., Santos-Neto, E., & Ripeanu, M. (2012). A Yoke of Oxen and a Thousand Chickens for Heavy Lifting Graph Processing. 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), 345-354.

[82] Gharaibeh, A., Santos-Neto, E., Costa, L., & Ripeanu, M. (2013). Efficient Large-Scale Graph Processing on Hybrid CPU and GPU Systems. arXiv:1312.3018.

[83] Gharan, S.O., & Trevisan, L. (2012). Approximating the Expansion Profile and Almost Optimal Local Graph Clustering. IEEE 53rd Annual Symposium on Foundations of Computer Science, 187-196.

[84] Gilmore, P. C., & Gomory, R. E. (1963). A Linear Programming Approach to the Cutting Stock Problem-Part II. Operations Research, Volume 11, Issue 6, 863-888.

[85] Gleich, D.F. and Seshadhri, C. (2012). Vertex Neighborhoods, Low Conductance Cuts, and Good Seeds for Local Community Methods. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12). Association for Computing Machinery, New York, NY, USA, 597-605. https://doi.org/10.1145/2339530.2339628.

[86] Goemans, M.X., & Rothvoss, T. (2013). Polynomiality for Bin Packing with A Constant Number of Item Types. Journal of the ACM (JACM), Volume 67, 1-21.

[87] Goller, C. & Küchler, A. (1996). Learning Task-Dependent Distributed Representations by Backpropagation through Structure. IEEE International Conference on Neural Networks. Volume 1, Issue 347. CiteSeerX 10.1.1.52.4759. doi:10.1109/ICNN.1996.548916. ISBN 978-0-7803-3210-2.

[88] Gonzalez, J.E., Low, L., Gu, H., Bickson, D., & Guestrin, C. (2012). PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI'12). USENIX Association, USA, 17-30.

[89] Goodman, L.A. (1961) Snowball Sampling. The Annals of Mathematical Statistics, 148-170.

[90] Govindaraju, N.K., Raghuvanshi, N., & Manocha, D. (2005). Fast and Approximate Stream Mining of Quantiles and Frequencies Using Graphics Processors. Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05). Association for Computing Machinery, New York, NY, USA, 611-622.

[91] Govindaraju, N., Gray, J., Kumar, R., & Manocha, D. (2006). GPUTeraSort: High Performance Graphics Co-Processor Sorting for Large Database Management. Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD '06). Association for Computing Machinery, New York, NY, USA, 325-336. https://doi.org/10.1145/1142473.1142511

[92] Graves, A., Wayne, G., Danihelka, I. (2014). Neural Turing Machines. arXiv:1410.5401v2. https://doi.org/10.48550/arXiv.1410.5401

[93] Gregor, D. & Lumsdaine, A. (2005). The Parallel BGL: A Generic Library for Distributed Graph Computations. Parallel Object-Oriented Scientific Computing (POOSC). 1-18.

[94] Greiner, J. (1994). A Comparison of Parallel Algorithms for Connected Components. Proceedings of 6th Annual ACM Symposium on Parallel Algorithms and Architectures. ACM, 16-25.

[95] Grishman, R., & Borthwick, A. (1999). A Maximum Entropy Approach to Named Entity Recognition. https://cs.nyu.edu/media/publications/borthwick_andrew.pdf

[96] Gross, J. (2010). Topics in Graph Theory. http://www.cs.columbia.edu/~cs6204/files/lec5-automorphisms.pdf

[97] Gross, J. L., Yellen, J. & Zhang, P. (2013). Handbook of Graph Theory. Chapman and Hall/CRC. ISBN 978-1-4398-8018-0.

[98] Hamilton, W.L., Ying, R., & Leskovec, J. (2018). Inductive Representation Learning on Large Graphs. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[99] Hammer, B. & Jain, B.J. (2004). Neural Methods for Non-Standard Data, ESANN'2004 in Proceedings of European Symposium on Artificial Neural Networks Bruges (Belgium), D-Side Publishing, ISBN 2-930307-04-8. 281-292

[100] Han, W., Mawhirter, D., Wu, B. & Buland, M. (2017). Graphie: Large-Scale Asynchronous Graph Traversals on Just A GPU. Parallel Architectures and Compilation Techniques (PACT). 233-245.

[101] Hanisch, D., Fundel, K., Mevissen, H. T., Zimmer, R., & Fluck, J. (2005). ProMiner: Rule-Based Protein and Gene Entity Recognition. BMC Bioinformatics, 6 Suppl 1(Suppl 1), S14. https://doi.org/10.1186/1471-2105-6-S1-S14

[102] Harish, P., & Narayanan, P.J. (2007). Accelerating Large Graph Algorithms on the GPU Using CUDA. Aluru, S., Parashar, M., Badrinath, R., Prasanna, V.K. (eds) High Performance Computing, HiPC 2007. Lecture Notes in Computer Science, Volume 4873, 197-208. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-77220-0_21

[103] Harshvardhan, Fidel, A., Amato, N.M., Rauchwerger, L. (2013). The STAPL Parallel Graph Library. in: Kasahara, H., Kimura, K. (Eds) Languages and Compilers for Parallel Computing. LCPC 2012. Lecture Notes in Computer Science, Volume 7760, 46-60. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37658-0_4.

[104] Harris, D.G. & Srinivasan, A. (2013). The Moser-Tardos Framework with Partial Resampling. in: IEEE 54th Annual Symposium on Foundations of Computer Science FOCS ’13, 469-478.

[105] Harshvardhan, West, B., Fidel, A., Amato, N.M., & L. Rauchwerger, L. (2015). A Hybrid Approach to Processing Big Data Graphs on Memory-Restricted Systems, 2015 IEEE International Parallel and Distributed Processing Symposium, Hyderabad, India, 2015, 799-808, doi: 10.1109/IPDPS.2015.28.

[106] He, B., Govindaraju, N.K., Luo, Q., & Smith, B.J. (2007). Efficient Gather and Scatter Operations on Graphics Processors. Proceedings of the 2007 ACM/IEEE Conference on Supercomputing (SC '07), 1-12.

[107] Hemminger, R.L. (1968). The Group of an X-Join Graphs. Journal of Combinatorial Theory. Volume 5, Issue 4, 408-418.

[108] Hoberg, R., & Rothvoss, T. (2017). A Logarithmic Additive Integrality Gap for Bin Packing. Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '17). Society for Industrial and Applied Mathematics, USA, 2616-2625.

[109] Hočevar, T., & Demšar, J. (2017). Combinatorial Algorithm for Counting Small Induced Graphs and Orbits. PLoS One, Volume 12, Issue 2, E0171428.

[110] Hočevar, T., & Demšar, J. (2014). A Combinatorial Approach to Graphlet Counting. Bioinformatics, Volume 30, Issue 4, 559-565.

[111] Holland, P.W., & Leinhardt, S. (1977). A Method for Detecting Structure in Sociometric Data, Editor(S): Samuel Leinhardt, Social Networks, Academic Press, 1977, 411-432, ISBN 978-0-1244-2450-0, https://doi.org/10.1016/B978-0-12-442450-0.50028-6.

[112] Hong, S., Oguntebi, T., & Olukotun, K. (2011). Efficient Parallel Graph Exploration on Multi-Core CPU and GPU. 2011 International Conference on Parallel Architectures and Compilation Techniques, Galveston, TX, USA, 2011, 78-88, doi: 10.1109/PACT.2011.14.

[113] Hong, S., Kim, S.K., Oguntebi, T., & Olukotun, K. (2011). Accelerating CUDA Graph Algorithms at Maximum Warp. ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming. Volume 46. ACM, 267-276.

[114] Hormozdiari, F., Berenbrink, P., Przulj, N., & Sahinalp, S.C. (2006). Not All Scale-Free Networks Are Born Equal: The Role of the Seed Graph in PPI Network Evolution. PLoS Computational Biology, Volume 3, Issue 7. https://doi.org/10.1371/journal.pcbi.0030118.

[115] Hsu, C., Chang, Y., Kuo, C., Lin, Y., Huang, H., & Chung, I. (2008). Integrating High Dimensional Bi-Directional Parsing Models for Gene Mention Tagging. Bioinformatics, Volume 24, 286-294.

[116] Hu, X., Tao, Y., & Chung, C.W. (2013). Massive Graph Triangulation. in Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13). Association for Computing Machinery, New York, NY, USA, 325-336. https://doi.org/10.1145/2463676.2463704.

[117] Hua, K.A. & Lee, C. (1991). Handling Data Skew in Multiprocessor Database Computers Using Partition Tuning. Proceedings of the 17th International Conference on Very Large Data Bases (VLDB '91). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 525-535.

[118] Huang, L., Ma, D., Li, S., Zhang, X., & Wang, H. (2019). Text Level Graph Neural Network for Text Classification. arXiv:1910.02356.

[119] Huang, P., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. Proceedings of the 22nd ACM International Conference on Information and Knowledge Management. 2333-2338.

[120] Im, S., Kell, N., Kulkarni, J. & Panigrahi, D. (2019). Tight Bounds for Online Vector Scheduling. SIAM Journal on Computing. Volume 48, No. 1, 93-121. https://epubs.siam.org/doi/abs/10.1137/17M111835X

[121] Jeub, L.G., Balachandran, P., Porter, M.A., Mucha, P.J., & Mahoney, M.W. (2014). Think Locally, Act Locally: The Detection of Small, Medium-Sized, and Large Communities in Large Networks. Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics, Volume 91, Issue 1, 012821. arXiv:1403.3795.

[122] Jha, M., Seshadhri, C., & Pinar, A. (2015). Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts. Proceedings of the 24th International Conference on World Wide Web (WWW ‘15). 495-505.

[123] Jia, Y., Lu, V., Hoberock, J., Garland, M., & Hart, J.C. (2011). Edge v. Node Parallelism for Graph Centrality Metrics. GPU Computing Gems. Volume 2, 15-30.

[124] Johnson, D.S. Fast Algorithms for Bin Packing. Journal of Computer and System Sciences, Volume 8, Issue 3, 272-314. https://doi.org/10.1016/S0022-0000(74)80026-7.

[125] Johnson, R. & Zhang, T. (2016). Supervised and Semi-Supervised Text Categorization Using LSTM for Region Embeddings. https://arXiv:1602.02373.

[126] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jogou, H., & Mikolov, T. (2016). Fasttext. Zip: Compressing Text Classification Models. https://arXiv:1612.03651.

[127] Kalchbrenner, N., Grefenstette, & E., Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. arXiv:1404.2188v1.

[128] Kale, L.V., & Bhatele, A. (2019). Parallel Science and Engineering Applications: The Charm++ Approach. CRC Press. Published October 23, 2019. ISBN 978-0-3673-7928-5.

[129] Kang, K., Ouyang, W., Li, H., & Wang, X. (2017). Object Detection from Video Tubelets with Convolutional Neural Networks. arXiv:1604.04053.

[130] Karmarkar, N., & Karp, R.M. (1982). an Efficient Approximation Scheme for the One-Dimensional Bin-Packing Problem. Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), 312-320.

[131] Karypis, G., & Kumar, V. (1998). A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. Proceedings of SIAM Journal on Scientific Computing, Volume 20, Issue 1, 359-392. doi:10.1137/S1064827595287997.

[132] Khorasani, F., Vora, K., Gupta, R. & Nbhuyan, L. (2014). Cusha: Vertex-Centric Graph Processing on GPUs. Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing. ACM, 239-252.

[133] Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized Kernels between Labeled Graphs. Proceedings of International Conference on Machine Learning. (ICML-2003), Washington DC, 2003. AAAI Press, 321-328.

[134] Kim, J., Han, W., Lee, S., Park, K., & Yu, H. (2014). OPT: A New Framework for Overlapped and Parallel Triangulation in Large-Scale Graphs. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14). Association for Computing Machinery, New York, NY, USA, 637-648. https://doi.org/10.1145/2588555.2588563.

[135] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP'14). Association for Computational Linguistics, Doha, Qatar. 1746-1751. doi:10.3115/V1/D14-1181.

[136] Kleinberg, J.M. (1999). Authoritative Sources in a Hyperlinked Environment. Journal of ACM, Volume 46, 604-632.

[137] Kloster, K., & Gleich, D.F. (2014). Heat Kernel Based Community Detection. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (KDD'14), 1386-1395.

[138] Kloumann, I.M., & Kleinberg, J.M. (2014). Community Membership Identification from Small Seed Sets. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 1366-1375.

[139] Kwok, T.C., & Lau, L.C. (2012). Finding Small Sparse Cuts by Random Walk. Proceedings of International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. APPROX RANDOM 2012 2012. Lecture Notes in Computer Science. Volume 7408, 615-626. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32512-0_52

[140] Kyrola, A., Blelloch, G.E., & Guestrin, C. (2012). Graphchi: Large-Scale Graph Computation on Just a PC. Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. (OSDI '12). 31-46.

[141] Lafferty, J., Mccallum, A., & Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proceedings of International Conference on Machine Learning (ICML). 282-289.

[142] Lauri, J., & Scapellato, R. (2003). Topics in Graph Automorphisms and Reconstruction. Cambridge University Press. March 17, 2003. ISBN-10: 0521529034.

[143] Le, Q.V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. Proceedings of the 31st International Conference on Machine Learning. Beijing, China. 1188-1196.

[144] Lecun, Y. & Bengio, Y. (1995). Convolutional Networks for Images, Speech, and Time Series. Arbib, Michael A. (Ed.). the Handbook of Brain Theory and Neural Networks (2nd Edition). the MIT Press. 276-278.

[145] Lee, C.C., & Lee, D.T. (1985). A Simple Online Bin-Packing Algorithm. Journal of ACM, Volume 32, 562-572.

[146] Lee, H., Brown, K.J., Sujeeth, A.K., Rompf, T., & Olukotun, K. (2014). Locality-Aware Mapping of Nested Parallel Patterns on GPUs. Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-47). IEEE Computer Society, USA, 63–74. https://doi.org/10.1109/MICRO.2014.23.

[147] Leek, T.R. (1997). Information Extraction Using Hidden Markov Models. Master Thesis. University of California at San Diego.

[148] Leskovec, J., Chakrabarti, D., Kleinberg, J. M. & Faloutsos, C. (2005). Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. PKDD 2005. PKDD 2005. Lecture Notes in Computer Science, Volume 3721, 133-145. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11564126_17.

[149] Leskovec, J., Kleinberg, J., & Faloutsos, C. (2005). Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. 177-187.

[150] Leskovec, J., Kleinberg, J.M., & Faloutsos, C. (2006). Graph Evolution: Densification and Shrinking Diameters. ACM Transactions on Knowledge Discovery from Data, Volume 1, No. 1, 2-es, 2007. https://doi.org/10.1145/1217299.1217301.

[151] Leskovec, J., Lang, K.J., Dasgupta, A., & Mahoney, M.W. (2008). Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics, Volume 6, 123-29.

[152] Leskovec, J., Lang, K.J., Dasgupta, A., & Mahoney, M.W. (2010). Empirical Comparison of Algorithms for Network Community Detection. Proceedings of the 19th International Conference on World Wide Web (WWW ‘10). 631-640.

[153] Li, Y., Lin, H., & Yang, Z. (2009). Incorporating Rich Background Knowledge for Gene Named Entity Classification and Recognition. BMC Bioinformatics, Volume 10, 223-223.

[154] Liao, C.S., Lu, K., Baym, M., Singh, R., & Berger, B. (2009). Isorankn: Spectral Methods for Global Alignment of Multiple Protein Networks. Bioinformatics, Volume 25, 253-258.

[155] Lieberman, M.D., Sankaranarayanan, J., & Samet, H. (2008). A Fast Similarity Join Algorithm Using Graphics Processing Units, 2008 IEEE 24th International Conference on Data Engineering, Cancun, Mexico. 1111-1120. doi: 10.1109/ICDE.2008.4497520.

[156] Liu, B. & Elke A. Rundensteiner, E.A. (2005). Revisiting Pipelined Parallelism in Multi-Join Query Processing. Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05). VLDB Endowment, 829-840.

[157] Liu, P., Qiu, X., Chen, X., Wu, S., & Huang, X. (2015). Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents. Conference on Empirical Methods in Natural Language Processing. 2326-2335.

[158] Liu, P., Chang, S., Huang, X., Tang, J., & Cheung, J.C. (2018). Contextualized Non-Local Neural Networks for Sequence Learning. Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and 31st Innovative Applications of Artificial Intelligence Conference and 9th AAAI Symposium on Educational Advances in Artificial Intelligence. Article 830, 6762–6769. https://doi.org/10.1609/aaai.v33i01.33016762

[159] Liu, P., Qiu, X., & Huang, X. (2016). Recurrent Neural Network for Text Classification with Multitask Learning. arXiv:1605.05101.

[160] Liu, S., Tang, B., Chen, Q., & Wang, X. (2015). Effects of Semantic Features on Machine Learning-Based Drug Name Recognition Systems. Information 2015, Volume 6, Issue 4, 848-865. https://doi.org/10.3390/info6040848.

[161] Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., & Hellerstein, J. (2010). Graphlab: A New Framework for Parallel Machine Learning. Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI'10). AUAI Press, Arlington, Virginia, USA. 340-349.

[162] Lu, H., Tan, K. & Shan, M. (1990). Hash-based Join Algorithms for Multiprocessor Computers with Shared Memory. Proceedings of the 16th VLDB Conference, Brisbane, Australia. 198-209.

[163] Luo, G., Wang, L., Yi, K., & Cormode. G. (2016). Quantiles over Data Streams: Experimental Comparisons, New Analyses, and Further Improvements. The VLDB Journal, Volume 25, 449-472.

[164] Luo, L., Wong, M.D., & Hwu, W.W. (2010). An Effective GPU Implementation of Breadth-First Search. Design Automation Conference, Anaheim, CA, USA. 52-55.

[165] Luong, M.T., Pham, H., & Manning, C.D. (2015). Effective Approaches to Attention-Based Neural Machine Translation. arXiv, Abs/1508.04025.

[166] Iyyer, M., Manjunatha, V., Boyd-Graber, J.L., & Daumé, H. (2015). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Volume 1, 1681-1691.

[167] Mahoney, M.W., Orecchia, L., & Vishnoi, N.K. (2009). A Local Spectral Method for Graphs: with Applications to Improving Graph Partitions and Exploring Data Graphs Locally. The Journal of Machine Learning Research, Volume 13, 2339-2365.

[168] Maji, S., Vishnoi, N.K., & Malik, J. (2011). Biased Normalized Cuts. Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA. 20-25.

[169] Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., & Czajkowski, G. (2010). Pregel: A System for Large-Scale Graph Processing. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10). Association for Computing Machinery, New York, NY, USA, 135-146. https://doi.org/10.1145/1807167.1807184.

[170] Maynard, D., Tablan, V., Bontcheva, K., Cunningham, H., & Wilks, Y. (2003). MUSE: A Multi-Source Entity Recognition System. Submitted to Computer and Humanities.

[171] Mcgregor, A. (2014). Graph Stream Algorithms: A Survey. ACM SIGMOD Record, Volume 43, 9-20. https://doi.org/10.1145/2627692.2627694.

[172] Mclaughlin, A., & Bader, D.A. (2014). Scalable and High Performance Betweenness Centrality on the GPU. International Conference for High Performance Computing, Networking, Storage and Analysis, 572-583. https://doi.org/10.1109/SC.2014.52.

[173] Menegola, B. (2010). An External Memory Algorithm for Listing Triangles. Technical Report, Universidade Federal Do Rio Grande Do Sul.

[174] Merrill, D., Garland, M., & Grimshaw, A.S. (2012). Scalable GPU Graph Traversal. ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming, Volume 47. ACM, 117-128.

[175] Meyer, U., & Sanders, P. (1998). Delta-Stepping: A Parallel Single Source Shortest Path Algorithm. European Symposium on Algorithms. Elsevier. 393-404.

[176] Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. 1st International Conference on Learning Representations, (ICLR 2013), Scottsdale, Arizona, USA. https://arXiv:1301.3781.

[177] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. & Dean, J. (2013). Distributed Representations of Words and Phrases and Their Compositionality. arXiv:1310.4546.

[178] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. (2002). Network Motifs: Simple Building Blocks of Complex Networks. Science. New York, New York, Volume 298, Issue 5594, 824-827. https://doi.org/10.1126/Science.298.5594.824.

[179] Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., & Bhattacharjee, B. (2007). Measurement and Analysis of Online Social Networks. Proceedings of the 7th ACM/SIGCOMM Internet Measurement Conference. ACM. 29-42.

[180] Mizutani, E., Dreyfus, S.E., & Nishio, K. (2000). On Derivation of MLP Backpropagation from the Kelley-Bryson Optimal-Control Gradient Formula and Its Application. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy Volume 2, 167-172. doi: 10.1109/IJCNN.2000.857892.

[181] Munkhdalai, T., Li, M., Kim, T., Namsrai, O., Jeong, S., Shin, J., & Ryu, K.H. (2012). Bio Named Entity Recognition Based on Co-Training Algorithm. 2012 26th International Conference on Advanced Information Networking and Applications Workshops, Fukuoka, Japan. 857-862. doi: 10.1109/WAINA.2012.75.

[182] Muthukrishnan, S. (2015). Data Streams: Algorithms and Applications. Foundations and Trends in Theoretical Computer Science, Volume 1, Issue 2, 117-236.

[183] Nasre, R., Burtscher, M., & Pingali, K. (2013). Atomic-Free Irregular Computations on Gpus. Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units. ACM. 96-107.

[184] Nasre, R., Burtscher, M., & Pingali, K. (2013). Morph Algorithms on GPUs. Proceedings of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP '13). Association for Computing Machinery, New York, NY, USA, 147–156. https://doi.org/10.1145/2442516.2442531

[185] Newman, M.E. (2003). The Structure and Function of Complex Networks. SIAM Review, Volume 45, No. 2, 167-256.

[186] Ngo, H.Q., Ré, C., & Rudra, A. (2013). Skew Strikes Back: New Developments in the Theory of Join Algorithms. SIGMOD Record, Volume 42, 5-16.

[187] Nguyen, D., Lenharth, A., & Pingali, K. (2013). A Lightweight Infrastructure for Graph Analytics. Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP '13). Association for Computing Machinery, New York, NY, USA, 456–471. https://doi.org/10.1145/2517349.2522739

[188] Ortega-Arranz, H., Torres, Y., Ferraris, D.R., & González-Escribano, A. (2013). A New GPU-Based Approach to the Shortest Path Problem. *013 International Conference on High Performance Computing & Simulation (HPCS)*, Helsinki, Finland, 2013, 505-511, doi: 10.1109/HPCSim.2013.6641461.

[189] Irsoy, O., & Cardie, C. (2014). Deep Recursive Neural Networks for Compositionality in Language. Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14). MIT Press, Cambridge, MA, USA, Volume 2, 2096–2104.

[190] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. The Web Conference. http://www.eecs.harvard.edu/~michaelm/CS222/pagerank.pdf.

[191] Pagh, R., & Silvestri, F. (2013). The Input/Output Complexity of Triangle Enumeration. Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '14). Association for Computing Machinery, New York, NY, USA, 224–233. https://doi.org/10.1145/2594538.2594552.

[192] Pai, S., & Pingali, K. (2016). A Compiler for Throughput Optimization of Graph Algorithms on GPUs. Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2016). Association for Computing Machinery, New York, NY, USA, 1–19. https://doi.org/10.1145/2983990.2984015.

[193] Park, H., & Chung, C. (2013). An Efficient MapReduce Algorithm for Counting Triangles in a Very Large Graph. Proceedings of the 22nd ACM international conference on Information & Knowledge Management (CIKM '13). Association for Computing Machinery, New York, NY, USA, 539–548. https://doi.org/10.1145/2505515.2505563

[194] Park, H-M., Silvestri, F., Kang, U., & Pagh, R. (2014). MapReduce Triangle Enumeration with Guarantees. in CIKM '14 Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. 1739-1748. Association for Computing Machinery. https://doi.org/10.1145/2661829.2662017.

[195] Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., & Yang, Q. (2018). Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1063–1072. https://doi.org/10.1145/3178876.3186005.

[196] Peng, H., Li, J., Gong, Q., Wang, S., He, L., Li, B., Wang, L., & Yu, P.S. (2019). Hierarchical Taxonomy-Aware and Attentional Graph Capsule RCNNs for Large-Scale Multi-Label Text Classification. IEEE Transactions on Knowledge and Data Engineering, Volume 33, No. 6, 2505-2519. doi: 10.1109/TKDE.2019.2959991.

[197] Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532–1543. Doha, Qatar. Association for Computational Linguistics.

[198] Pentland, A.S., & Heibeck, T. (2008). Understanding "Honest Signals" in Business. MIT Sloan Management Review, Volume 50, 70-75.

[199] Perevalov, A. & andreas Both, A. (2021). Improving Answer Type Classification Quality through Combined Question Answering Datasets. In: Qiu, H., Zhang, C., Fei, Z., Qiu, M., Kung, SY. (eds) Knowledge Science, Engineering and Management . KSEM 2021. Lecture Notes in Computer Science(), Volume 12816, 191-204. Springer, Cham. https://doi.org/10.1007/978-3-030-82147-0_16.

[200] Pinar, A., Comandur, S., & Vishal, V. (2017). ESCAPE: Efficiently Counting All 5-Vertex Subgraphs. Proceedings of the 26th International Conference on World Wide Web (WWW '17). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1431-1440. https://doi.org/10.1145/3038912.3052597.

[201] Pineda, F.J. (1987). Generalization of Back Propagation to Recurrent and Higher Order Neural Networks. Proceedings of the 1987 International Conference on Neural Information Processing Systems (NIPS'87). MIT Press, Cambridge, MA, USA, 602–611. https://doi.org/10.1103/Physrevlett.59.2229.

[202] Pingali, K., Nguyen, D., Kulkarni, M., Burtscher, M., Hassaan, M.A., Kaleem, R., Lee, T., Lenharth, A., Manevich, R., Méndez-Lojo, M., Prountzos, D., & Sui, X. (2011). The Tao of Parallelism in Algorithms. Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '11). Association for Computing Machinery, New York, NY, USA, 12–25. https://doi.org/10.1145/1993498.1993501.

[203] Portes, A. (1998). Social Capital: Its Origins and Applications in Modern Sociology. Annual Review of Sociology, Volume 24, 1–24. http://www.jstor.org/stable/223472.

[204] Proux D., Rechenmann F., Julliard L., Pillet V., and Jacq B. (1998) Detecting Gene Symbols and Names in Biological Texts: A First Step Toward Pertinent Information Extraction. Genome Informatics. Workshop on Genome Informatics, Volume 9, 72-80.

[205] Przulj, N., Corneil, D.G., & Jurisica, I. (2004). Modeling Interactome: Scale-Free or Geometric? Bioinformatics, Volume 20, Issue 18, 3508-3515. https://doi.org/10.1093/bioinformatics/bth436

[206] Qasim, R., Bangyal, W.H., Alqarni, M.A., & Ali Almazroi, A. (2022). A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification. Journal of Healthcare Engineering, 3498123. https://doi.org/10.1155/2022/3498123.

[207] Rahman, M., Bhuiyan, M., & Hasan, M.A. (2014). Graft: An Efficient Graphlet Counting Method for Large Graph Analysis. IEEE Transactions on Knowledge and Data Engineering, Volume 26, No. 10, 2466-2478. doi: 10.1109/TKDE.2013.2297929.

[208] Rocktäschel, T., Huber, T., Weidlich, M., & Leser, U. (2013). WBI-NER: the Impact of Domain-Specific Features on the Performance of Identifying and Classifying Mentions of Drugs. Proceedings of the 7th International Workshop on Semantic Evaluation, Volume 2, 356-363, Atlanta, Georgia, USA.

[209] Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning Internal Representations by Back- Propagating Errors. in D. Rumelhart and J. Mcclelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1, 318-362. MIT Press.

[210] Sabet, A.H., Qiu, J., & Zhao, Z. (2018). Tigr: Transforming Irregular Graphs for GPU-Friendly Graph Processing. Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18). Association for Computing Machinery, New York, NY, USA, 622-636. https://doi.org/10.1145/3173162.3173180

[211] Sadowski, G., & Rathle, P. (2014). Fraud Detection: Discovering Connections with Graph Databases. White Paper-Neo Technology-Graphs are Everywhere. https://go.neo4j.com/rs/710-RRC-335/images/Neo4j_WP-Fraud-Detection-with-Graph-Databases.pdf?_ga=2.152229817.1435723348.1577409683-120002542.1565112145

[212] Sarıyüce, A.E., Kaya, K., Saule, E., & Çatalyürek, Ü.V. (2013). Betweenness Centrality on Gpus and Heterogeneous Architectures. Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units (GPGPU-6). Association for Computing Machinery, New York, NY, USA, 76–85. https://doi.org/10.1145/2458523.2458531.

[213] Sartori, J., & Kumar, R. (2013). Branch and Data Herding: Reducing Control and Memory Divergence for Error-Tolerant GPU Applications. IEEE Transactions on Multimedia, Volume 15, Issue 2, 279-290. https://doi.org/10.1109/TMM.2012.2232647.

[214] Saucan, E., Samal, A., & Jost, J. (2020). A Simple Differential Geometry for Complex Networks. Network Science, Volume 9, 106-133.

[215] Schank, T., & Wagner, D. (2005). Approximating Clustering Coefficient and Transitivity. J. Graph Algorithms Appl., Volume 9, 265-275.

[216] Schmidhuber, J. Deep Learning in Neural Networks: An Overview. Neural Networks, Volume 61, 85-117, ISSN 0893-6080. https://doi.org/10.1016/J.Neunet.2014.09.003.

[217] Sengupta, D., Song, S. L., Agarwal, K. & Schwan, K. (2015). GraphReduce: Processing Large-Scale Graphs on Accelerator-Based Systems. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, USA, 2015, 1-12, doi: 10.1145/2807591.2807655.

[218] Seshadhri, C., Kolda, T.G., & Pinar, A. (2012). Community Structure and Scale-Free Collections of Erdos-Renyi Graphs. Physical Review E, Volume 85, Issue 5. https://doi.org/10.1103/Physreve.85.056109.

[219] Comandur, S., Pinar, A., & Kolda, T.G. (2012). Fast Triangle Counting through Wedge Sampling. arXiv, Abs/1202.5230.

[220] Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G. (2014). A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14). Association for Computing Machinery, New York, NY, USA, 101–110. https://doi.org/10.1145/2661829.2661935.

[221] Shervashidze, N., Schweitzer, P., Leeuwen, E.J., Mehlhorn, K., & Borgwardt, K.M. (2011). Weisfeiler-Lehman Graph Kernels. Journal of Machine Learning Research, Volume 12, 2539-2561.

[222] Shun, J., & Blelloch, G.E. (2013). Ligra: A Lightweight Graph Processing Framework for Shared Memory. Proceedings of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP '13). Association for Computing Machinery, New York, NY, USA, 135–146. https://doi.org/10.1145/2442516.2442530

[223] Siek, J.G., The Boost Graph Library: User Guide and Reference Manual. Addison Wesley Professional. ISBN: 1282692038, 978-1-2826-9203-9

[224] Socher, R., Huang, E.H., Pennington, J., Ng, A.Y. & Manning, C.D. (2011). Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11). Curran Associates Inc., Red Hook, NY, USA, 801-809. https://Proceedings.Neurips.Cc/Paper/2011/File/3335881e06d4d23091389226225e17c7-Paper.Pdf

[225] Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2013). Recursive Deep Models for Semantic Compositionality over a Sentiment Tree-Bank. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics..

[226] Soman, J., Kothapalli, K., & Narayanan, P.J. (2010). A Fast GPU Algorithm for Graph Connectivity. 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), Atlanta, GA, USA, 2010, 1-8. doi: 10.1109/IPDPSW.2010.5470817.

[227] Spielman, D.A., & Teng, S. (2003). Nearly-Linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems. arXiv, Cs.DS/0310051.

[228] Spielman, D.A., & Teng, S. (2008). A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Partitioning. SIAM Journal on Computing, Volume 42, Issue 1, 1-26. https://doi.org/10.1137/080744888

[229] Sreejith, R.P., Jost, J., Saucan, E. & Samal, A. (2017). Systematic Evaluation of a New Combinatorial Curvature for Complex Networks, Chaos, Solitons & Fractals, Volume 101, 50-67, ISSN 0960-0779, https://doi.org/10.1016/J.Chaos.2017.05.021.

[230] Stumpf, M.P., Wiuf, C., & May, R.M. (2005). Subnets of Scale-Free Networks are Not Scale-Free: Sampling Properties of Networks. Proceedings of the National Academy of Sciences of the United States of America, Volume 102, Issue 12, 4221-4224.

[231] Suárez-Paniagua, V., Zavala, R.M., Segura-Bedmar, I., & Martínez, P. (2019). A Two-Stage Deep Learning Approach for Extracting Entities and Relationships from Medical Texts. Journal of Biomedical Informatics, Volume 99, 103285.

[232] Sumner, D.P. (1973). Graphs Indecomposable with Respect to the X-Join. Discrete Math. Volume 6, Issue 3, 281-298. https://doi.org/10.1016/0012-365X(73)90100-3

[233] Sun, C., Agrawal, D. & Abbadi, A.E. (2003). Hardware Acceleration for Spatial Selections and Joins. Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03). Association for Computing Machinery, New York, NY, USA, 455-466. https://doi.org/10.1145/872757.872813

[234] Suri, S., & Vassilvitskii, S. (2011). Counting Triangles and the Curse of the Last Reducer. Proceedings of the 20th international conference on World wide web (WWW '11). Association for Computing Machinery, New York, NY, USA, 607–614. https://doi.org/10.1145/1963405.1963491.

[235] Tai, K.S., Socher, R., & Manning, C.D. (2015). Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks. arXiv, Abs/1503.00075.

[236] C. Tsourakakis, M. N. Kolountzakis, and G. Miller. Triangle Sparsifiers. Journal of Graph Algorithms and Applications. Volume 15, No. 6, 703-726. http://Jgaa.Info/

[237] Tsourakakis, C.E., Drineas, P., Michelakis, E., Koutis, I., & Faloutsos, C. (2011). Spectral Counting of Triangles Via Elementwise Sparsification and Triangle-Based Link Recommendation. Social Network Analysis and Mining, Volume 1, 75-81.

[238] Tsourakakis, C.E., Kang, U., Miller, G.L., & Faloutsos, C. (2009). DOULION: Counting Triangles in Massive Graphs with a Coin. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09). Association for Computing Machinery, New York, NY, USA, 837–846. https://doi.org/10.1145/1557019.1557111.

[239] Tsuruoka, Y., & Tsujii, J. (2004). Improving the Performance of Dictionary-Based Approaches in Protein Name Recognition. Journal of Biomedical Informatics, Volume 37, Issue 6, 461-470. https://doi.org/10.1016/J.Jbi.2004.08.003

[240] Tzeng, S., Patney, A., & Owens, J.D. (2010). Task Management for Irregular-Parallel Workloads on the GPU. Proceedings of the Conference on High Performance Graphics (HPG '10). Eurographics Association, Goslar, DEU, 29–37.

[241] Ugander, J., Backstrom, L., & Kleinberg, J.M. (2013). Subgraph Frequencies: Mapping the Empirical and Extremal Geography of Large Graph Collections. Proceedings of the 22nd international conference on World Wide Web (WWW '13). Association for Computing Machinery, New York, NY, USA, 1307–1318. https://doi.org/10.1145/2488388.2488502.

[242] Jauregi Unanue, I., Borzeshi, E.Z., & Piccardi, M. (2017). Recurrent Neural Networks with Specialized Word Embeddings for Health-Domain Named-Entity Recognition. Journal of Biomedical Informatics, Volume 76, 102-109.

[243] Valiant, L.G. (1990). A Bridging Model for Parallel Computation. Communication of the ACM, Volume 33, No. 8, 103-111.

[244] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., & Polosukhin, I., Attention is All You Need. Advances in Neural Information Processing Systems, Volume 30, 5998-6008. http://arXiv.org/Abs/1706.03762.

[245] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., & Bengio, Y. (2017). Graph Attention Networks. ArXiv, abs/1710.10903.

[246] Vinyals, O., Fortunato, M. & Jaitly, N. (2015). Pointer Networks. arXiv:1506.03134v2. https://doi.org/10.48550/arXiv.1506.03134.

[247] Viterbi, A.J .(1967). Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. IEEE Transactions on Information Theory. Volume 13, Issue 2, 260-269. doi:10.1109/TIT.1967.1054010.

[248] Voevodski, K., Teng, S. H., & Xia, Y. (2009). Finding Local Communities in Protein Networks. BMC Bioinformatics, Volume 10, Issue 297. https://doi.org/10.1186/1471-2105-10-297

[249] Wagner, S.M., & Neshat, N. (2010). Assessing the Vulnerability of Supply Chains Using Graph Theory. International Journal of Production Economics, Volume 126, 121-129.

[250] Waller, D.A. (1976). Double Covers of Graphs. Bulletin of the Australian Mathematical Society, Volume 14, 233-248.

[251] Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., & Cheng, X. (2015). A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16). AAAI Press, 2835–2841.

[252] Wang, K., Xu, G.H., Su, Z., & Liu, Y.D. (2015). GraphQ: Graph Query Processing with Abstraction Refinement - Scalable and Programmable Analytics Over Very Large Graphs on A Single PC. USENIX Annual Technical Conference. Santa Clara, CA. USENIX Association. 387-401 ISBN: 978-1-931971-225.

[253] Wang, K., Hussain, A., Zuo, Z., Xu, G.H., & Sani, A.A. (2017). Graspan: A Single-Machine Disk-Based Graph System for Interprocedural Static Analyses of Large-Scale Systems Code. Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17). Association for Computing Machinery, New York, NY, USA, 389-404. https://doi.org/10.1145/3037697.3037744.

[254] Wang, S.I., & Manning, C.D. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12), Volume 2, 90-94. Association for Computational Linguistics, USA.

[255] Wang, S., & Jiang, J. (2015). Learning Natural Language Inference with LSTM. arXiv, Abs/1512.08849.

[256] Wang, W., Gu, Y., Wang, Z., Yu, G. (2013). Parallel Triangle Counting over Large Graphs. in: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (Eds) Database Systems for Advanced Applications. DASFAA 2013. Lecture Notes in Computer Science, Volume 7826, 301-308. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37450-0_23.

[257] Watts, D. J., & Strogatz, S. H. (1998). Collective Dynamics of 'Small-World' Networks. Nature, Volume 393, Issue 6684, 440-442. https://doi.org/10.1038/30918.

[258] Whang, J.J., Gleich, D.F. & Dhillon, I.S. (2013). Overlapping Community Detection Using Seed Set Expansion. Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13). Association for Computing Machinery, New York, NY, USA, 2099-2108. https://doi.org/10.1145/2505515.2505535.

[259] Woeginger, G.J. (1997). There is No Asymptotic PTAS for Two-Dimensional Vector Packing. Information Processing Letters, Volume 64, Issue 6, 293-297.

[260] Wong,E., Baur, B., Quader, S.A., & Huang, C. (2012). Biological Network motif Detection : Principles and Practice. Briefings in bioinformatics, Volume 13, Issue 2, 202-215. https://doi.org/10.1093/bib/bbr033.

[261] Wu, F., Zhang, T., Souza, A.H., Fifty, C., Yu, T., & Weinberger, K.Q. (2019). Simplifying Graph Convolutional Networks. arXiv, Abs/1902.07153.

[262] Wu, Y., Jin, R., Li, J., & Zhang, X. (2015). Robust Local Community Detection: on Free Rider Effect and Its Elimination. Proceedings of VLDB Endowment, Volume 8, No. 7, 798-809. https://doi.org/10.14778/2752939.2752948

[263] Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of 32nd International Conference on Machine Learning, ICML 2015. Volume 3, 2048-2057.

[264] Xu, Q., Jeon, H., & Annavaram, M. (2014). Graph Processing on GPUs: Where Are the Bottlenecks? 2014 IEEE International Symposium on Workload Characterization (IISWC), Raleigh, NC, USA, 140-149, doi: 10.1109/IISWC.2014.6983053.

[265] Yang, J., & Leskovec, J. (2012). Defining and Evaluating Network Communities Based on Ground-Truth. Knowledge and Information Systems, Volume 42, 181-213. https://doi.org/10.1007/s10115-013-0693-z

[266] Yang, Y., Li, C., & Zhou, H. (2015). CUDA-NP: Realizing Nested Thread-Level Parallelism in GPGPU Applications. Proceedings of the 19th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP '14). Association for Computing Machinery, New York, NY, USA, 93–106. https://doi.org/10.1145/2555243.2555254.

[267] Yang, Z., Lin, H., & Li, Y. (2008). Exploiting the Performance of Dictionary-Based Bio-Entity Name Recognition in Biomedical Literature. Computational Biology and Chemistry, Volume 32, Issue 4, 287-291. https://doi.org/10.1016/j.compbiolchem.2008.03.008.

[268] Yao, L., Mao, C., & Luo, Y. (2018). Graph Convolutional Networks for Text Classification. AAAI Conference on Artificial Intelligence. Volume 33, No. 1, 7370-7377.

[269] Yih, W., Toutanova, K., Platt, J.C., & Meek, C. (2011). Learning Discriminative Projections for Text Similarity Measures. Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL '11). Association for Computational Linguistics, USA, 247–256.

[270] Yu, S., Su, J., & Luo, D. (2019). Improving BERT-Based Text Classification with Auxiliary Sentence and Domain Knowledge. IEEE Access, Volume 7, 176600-176612.

[271] Zhang, E.Z., Jiang, Y., Guo, Z., Tian, K., & Shen, X. (2011). On-the-fly Elimination of Dynamic Irregularities for GPU Computing. Proceedings of the 16th international conference on Architectural support for programming languages and operating systems (ASPLOS XVI). Association for Computing Machinery, New York, NY, USA, 369–380. https://doi.org/10.1145/1950365.1950408.

[272] Zhao. Y. Graph Theory and Additive Combinatorics. Notes for MIT 18.217 (Fall 2019). 49-58. https://Yufeizhao.Com/Gtac/Gtac.Pdf.

[273] Zhao, Z., Wu, B., & Shen, X. (2014). Challenging the "Embarrassingly Sequential": Parallelizing Finite State Machine-Based Computations through Principled Speculation. Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS 2014: 543-558.

[274] Zhao, Z., & Shen, X. (2015). On-The-Fly Principled Speculation for FSM Parallelization. Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2015, Istanbul, Turkey, 619-630. https://doi.org/10.1145/2694344.2694369.

[275] Zhong, J. & He, B. (2014). Medusa: Simplified Graph Processing on GPUs, IEEE Transactions on Parallel and Distributed Systems, Volume 25, No. 6, 1543-1552, doi: 10.1109/TPDS.2013.111.

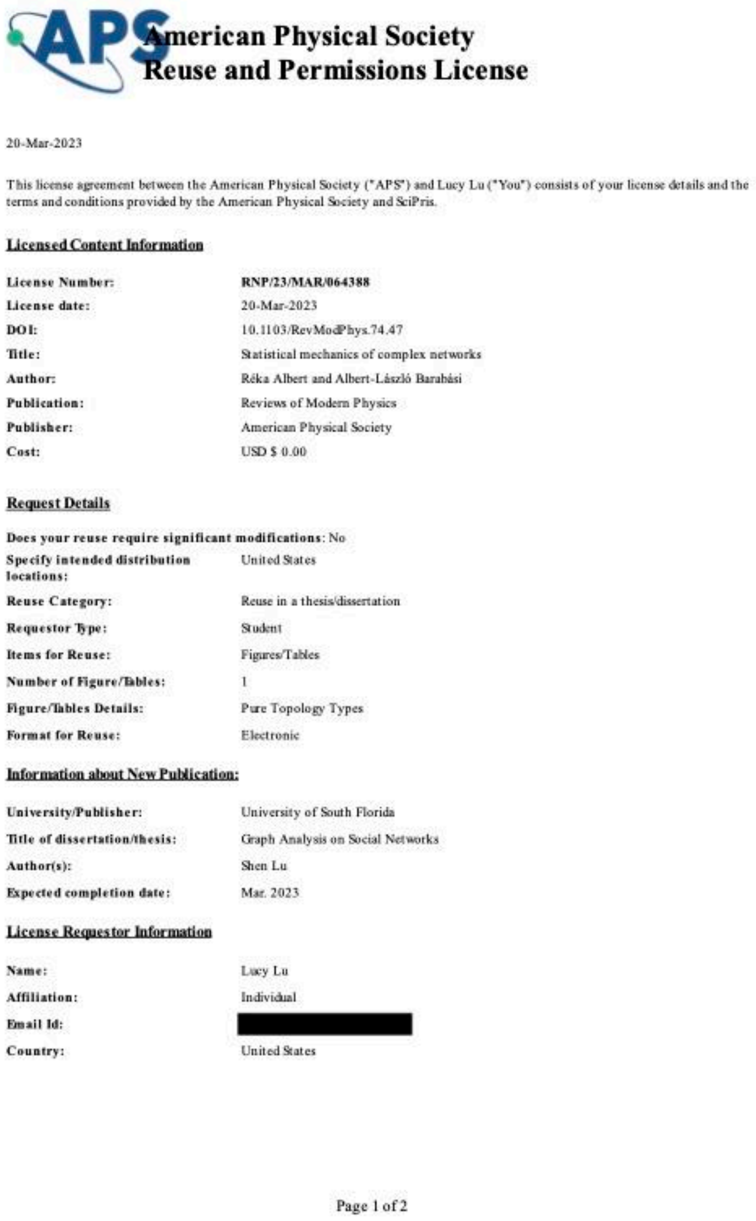
[276] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H. & Xu, B. (2016). Text Classification Improved by Integrating Bidirectional LSTM With Two-Dimensional Max Pooling. arXiv:1611.06639. https://arXiv:1702.03814.

[277] Zhu, X., Sobhani, P., & Guo, H. (2015). Long Short-Term Memory Over Recursive Structures. Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML'15), Volume 37, 1604–1612. https://proceedings.mlr.press/v37/zhub15.html.

[278] Ztech CPU and GPU. (2018). Are GPUs Really Taking over CPUs? Source: https://hypertec.com/blog/gpus-taking-over-cpus/.

Appendix A: Copyright Permissions

The permission below is for the use of Figure 3.2. Pure Topology Types in Chapter 3.



American Physical Society
Reuse and Permissions License

20-Mar-2023

This license agreement between the American Physical Society ("APS") and Lucy Lu ("You") consists of your license details and the terms and conditions provided by the American Physical Society and SciPris.

Licensed Content Information

License Number:	**RNP/23/MAR/064388**
License date:	20-Mar-2023
DOI:	10.1103/RevModPhys.74.47
Title:	Statistical mechanics of complex networks
Author:	Réka Albert and Albert-László Barabási
Publication:	Reviews of Modern Physics
Publisher:	American Physical Society
Cost:	USD $ 0.00

Request Details

Does your reuse require significant modifications: No

Specify intended distribution locations:	United States
Reuse Category:	Reuse in a thesis/dissertation
Requestor Type:	Student
Items for Reuse:	Figures/Tables
Number of Figure/Tables:	1
Figure/Tables Details:	Pure Topology Types
Format for Reuse:	Electronic

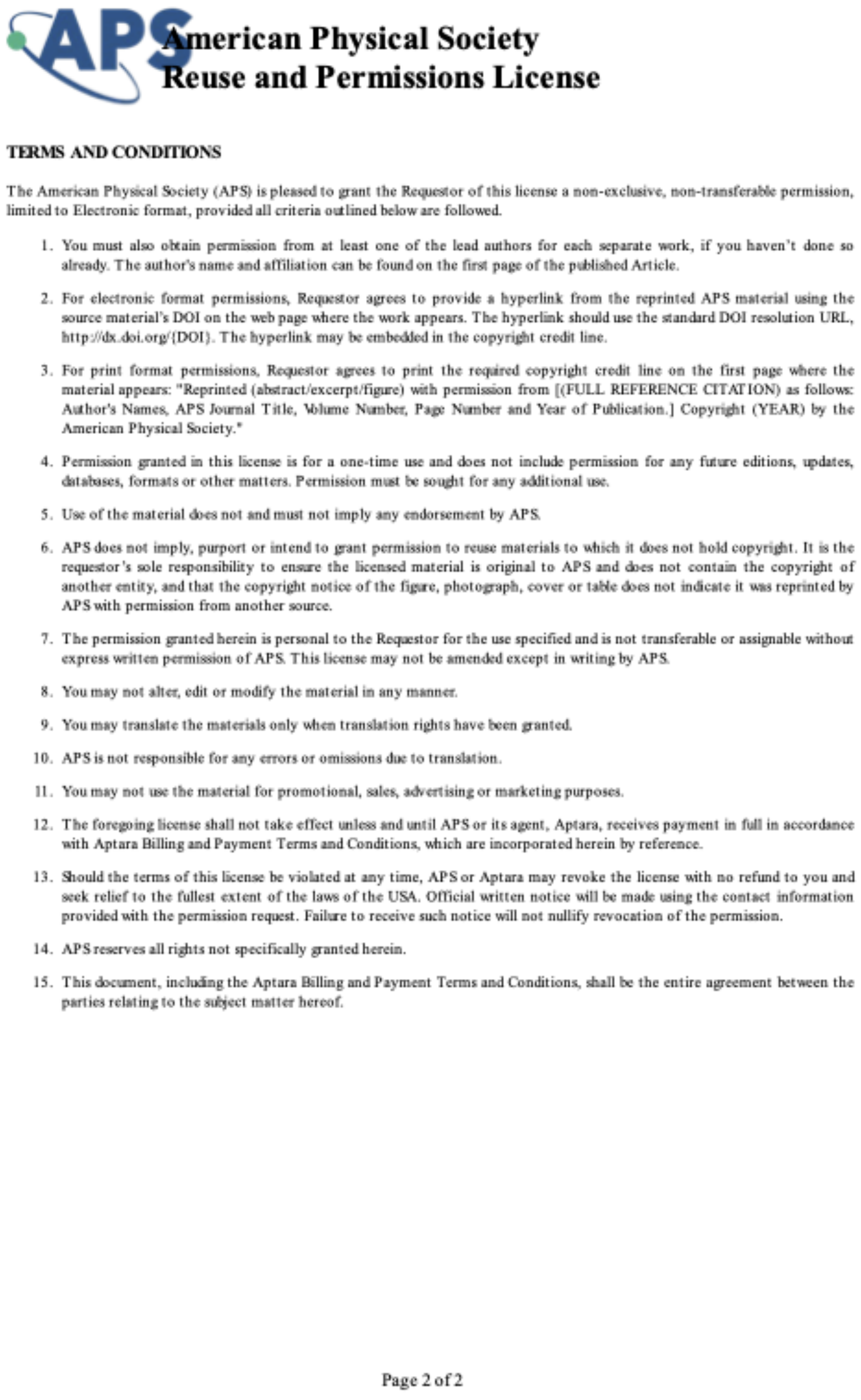
Information about New Publication:

University/Publisher:	University of South Florida
Title of dissertation/thesis:	Graph Analysis on Social Networks
Author(s):	Shen Lu
Expected completion date:	Mar. 2023

License Requestor Information

Name:	Lucy Lu
Affiliation:	Individual
Email Id:	[redacted]
Country:	United States

Page 1 of 2



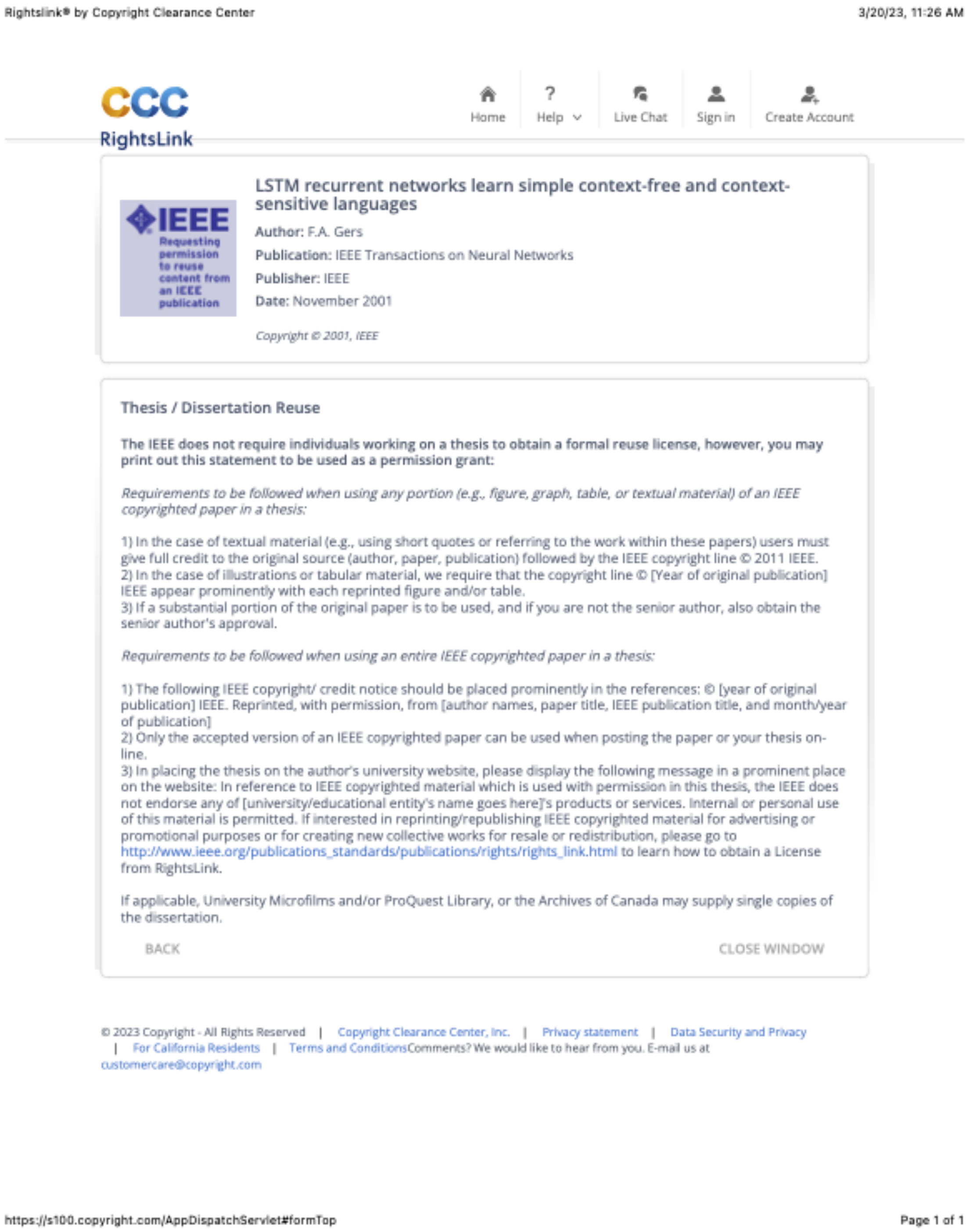
American Physical Society
Reuse and Permissions License

TERMS AND CONDITIONS

The American Physical Society (APS) is pleased to grant the Requestor of this license a non-exclusive, non-transferable permission, limited to Electronic format, provided all criteria outlined below are followed.

1. You must also obtain permission from at least one of the lead authors for each separate work, if you haven't done so already. The author's name and affiliation can be found on the first page of the published Article.
2. For electronic format permissions, Requestor agrees to provide a hyperlink from the reprinted APS material using the source material's DOI on the web page where the work appears. The hyperlink should use the standard DOI resolution URL, http://dx.doi.org/{DOI}. The hyperlink may be embedded in the copyright credit line.
3. For print format permissions, Requestor agrees to print the required copyright credit line on the first page where the material appears: "Reprinted (abstract/excerpt/figure) with permission from [(FULL REFERENCE CITATION) as follows: Author's Names, APS Journal Title, Volume Number, Page Number and Year of Publication.] Copyright (YEAR) by the American Physical Society."
4. Permission granted in this license is for a one-time use and does not include permission for any future editions, updates, databases, formats or other matters. Permission must be sought for any additional use.
5. Use of the material does not and must not imply any endorsement by APS.
6. APS does not imply, purport or intend to grant permission to reuse materials to which it does not hold copyright. It is the requestor's sole responsibility to ensure the licensed material is original to APS and does not contain the copyright of another entity, and that the copyright notice of the figure, photograph, cover or table does not indicate it was reprinted by APS with permission from another source.
7. The permission granted herein is personal to the Requestor for the use specified and is not transferable or assignable without express written permission of APS. This license may not be amended except in writing by APS.
8. You may not alter, edit or modify the material in any manner.
9. You may translate the materials only when translation rights have been granted.
10. APS is not responsible for any errors or omissions due to translation.
11. You may not use the material for promotional, sales, advertising or marketing purposes.
12. The foregoing license shall not take effect unless and until APS or its agent, Aptara, receives payment in full in accordance with Aptara Billing and Payment Terms and Conditions, which are incorporated herein by reference.
13. Should the terms of this license be violated at any time, APS or Aptara may revoke the license with no refund to you and seek relief to the fullest extent of the laws of the USA. Official written notice will be made using the contact information provided with the permission request. Failure to receive such notice will not nullify revocation of the permission.
14. APS reserves all rights not specifically granted herein.
15. This document, including the Aptara Billing and Payment Terms and Conditions, shall be the entire agreement between the parties relating to the subject matter hereof.

The permission below is for the use of Figure 3.5. Long Short-Term Memory Prototype in Chapter 3.


Home
Help
Live Chat
Sign in
Create Account
CCC RightsLink
IEEE Requesting permission to reuse content from an IEEE publication

Rightslink® by Copyright Clearance Center

3/20/23, 11:26 AM

LSTM recurrent networks learn simple context-free and context-sensitive languages

Author: F.A. Gers

Publication: IEEE Transactions on Neural Networks

Publisher: IEEE

Date: November 2001

Copyright © 2001, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

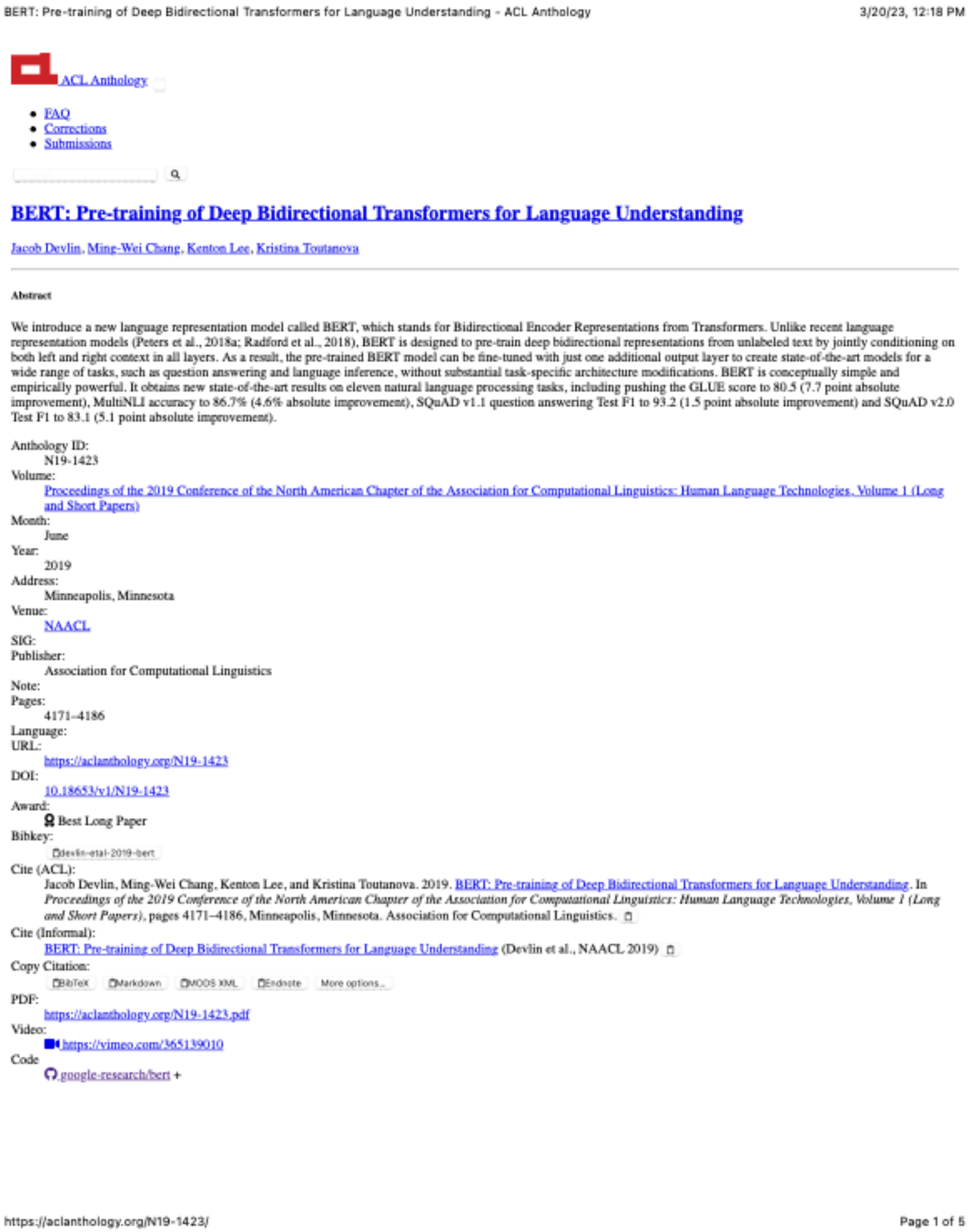
1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

https://s100.copyright.com/AppDispatchServlet#formTop

Page 1 of 1

The permission below is for the use of Figure 7.2. BERT Single Sentence Classification Network Architecture in Chapter 7.



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - ACL Anthology 3/20/23, 12:18 PM

ACL Anthology

- FAQ
- Corrections
- Submissions

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

Abstract

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5 (7.7 point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

Anthology ID:
N19-1423
Volume:
Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)
Month:
June
Year:
2019
Address:
Minneapolis, Minnesota
Venue:
NAACL
SIG:
Publisher:
Association for Computational Linguistics
Note:
Pages:
4171–4186
Language:
URL:
https://aclanthology.org/N19-1423
DOI:
10.18653/v1/N19-1423
Award:
Best Long Paper
Bibkey:
devlin-etal-2019-bert
Cite (ACL):
Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
Cite (Informal):
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019)
Copy Citation:

PDF:
https://aclanthology.org/N19-1423.pdf
Video:
https://vimeo.com/365139010
Code
google-research/bert +

https://aclanthology.org/N19-1423/ Page 1 of 5

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - ACL Anthology 3/20/23, 12:18 PM

```
    title = "{BERT}: Pre-training of Deep Bidirectional Transformers for Language Understanding",
    author = "Devlin, Jacob  and
      Chang, Ming-Wei  and
      Lee, Kenton  and
      Toutanova, Kristina",
    booktitle = "Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Langua
    month = jun,
    year = "2019",
    address = "Minneapolis, Minnesota",
    publisher = "Association for Computational Linguistics",
    url = "https://aclanthology.org/N19-1423",
    doi = "10.18653/v1/N19-1423",
    pages = "4171--4186",
    abstract = "We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transfon
}
```

Download as File Copy to Clipboard

```
<?xml version="1.0" encoding="UTF-8"?>
<modsCollection xmlns="http://www.loc.gov/mods/v3">
<mods ID="devlin-etal-2019-bert">
    <titleInfo>
        <title>BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding</title>
    </titleInfo>
    <name type="personal">
        <namePart type="given">Jacob</namePart>
        <namePart type="family">Devlin</namePart>
        <role>
            <roleTerm authority="marcrelator" type="text">author</roleTerm>
        </role>
    </name>
    <name type="personal">
        <namePart type="given">Ming-Wei</namePart>
        <namePart type="family">Chang</namePart>
        <role>
            <roleTerm authority="marcrelator" type="text">author</roleTerm>
        </role>
    </name>
    <name type="personal">
        <namePart type="given">Kenton</namePart>
        <namePart type="family">Lee</namePart>
        <role>
            <roleTerm authority="marcrelator" type="text">author</roleTerm>
        </role>
    </name>
    <name type="personal">
        <namePart type="given">Kristina</namePart>
        <namePart type="family">Toutanova</namePart>
        <role>
            <roleTerm authority="marcrelator" type="text">author</roleTerm>
        </role>
    </name>
    <originInfo>
        <dateIssued>2019-06</dateIssued>
    </originInfo>
    <typeOfResource>text</typeOfResource>
    <relatedItem type="host">
        <titleInfo>
            <title>Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Langua
        </titleInfo>
        <originInfo>
            <publisher>Association for Computational Linguistics</publisher>
            <place>
                <placeTerm type="text">Minneapolis, Minnesota</placeTerm>
            </place>
        </originInfo>
        <genre authority="marcgt">conference publication</genre>
    </relatedItem>
    <abstract>We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transforme
    <identifier type="citekey">devlin-etal-2019-bert</identifier>
    <identifier type="doi">10.18653/v1/N19-1423</identifier>
    <location>
        <url>https://aclanthology.org/N19-1423</url>
    </location>
    <part>
        <date>2019-06</date>
        <extent unit="page">
            <start>4171</start>
            <end>4186</end>
        </extent>
    </part>
</mods>
</modsCollection>
```

Download as File Copy to Clipboard

```
%0 Conference Proceedings
%T BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
%A Devlin, Jacob
%A Chang, Ming-Wei
%A Lee, Kenton
%A Toutanova, Kristina
%S Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,
%D 2019
%8 June
%I Association for Computational Linguistics
%C Minneapolis, Minnesota
%F devlin-etal-2019-bert
%X We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike
%R 10.18653/v1/N19-1423
%U https://aclanthology.org/N19-1423
%U https://doi.org/10.18653/v1/N19-1423
%P 4171-4186
```

Download as File Copy to Clipboard

Markdown (Informal)

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https://aclanthology.org/N19-1423) (Devlin et al., NAACL 2019)

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019)

ACL

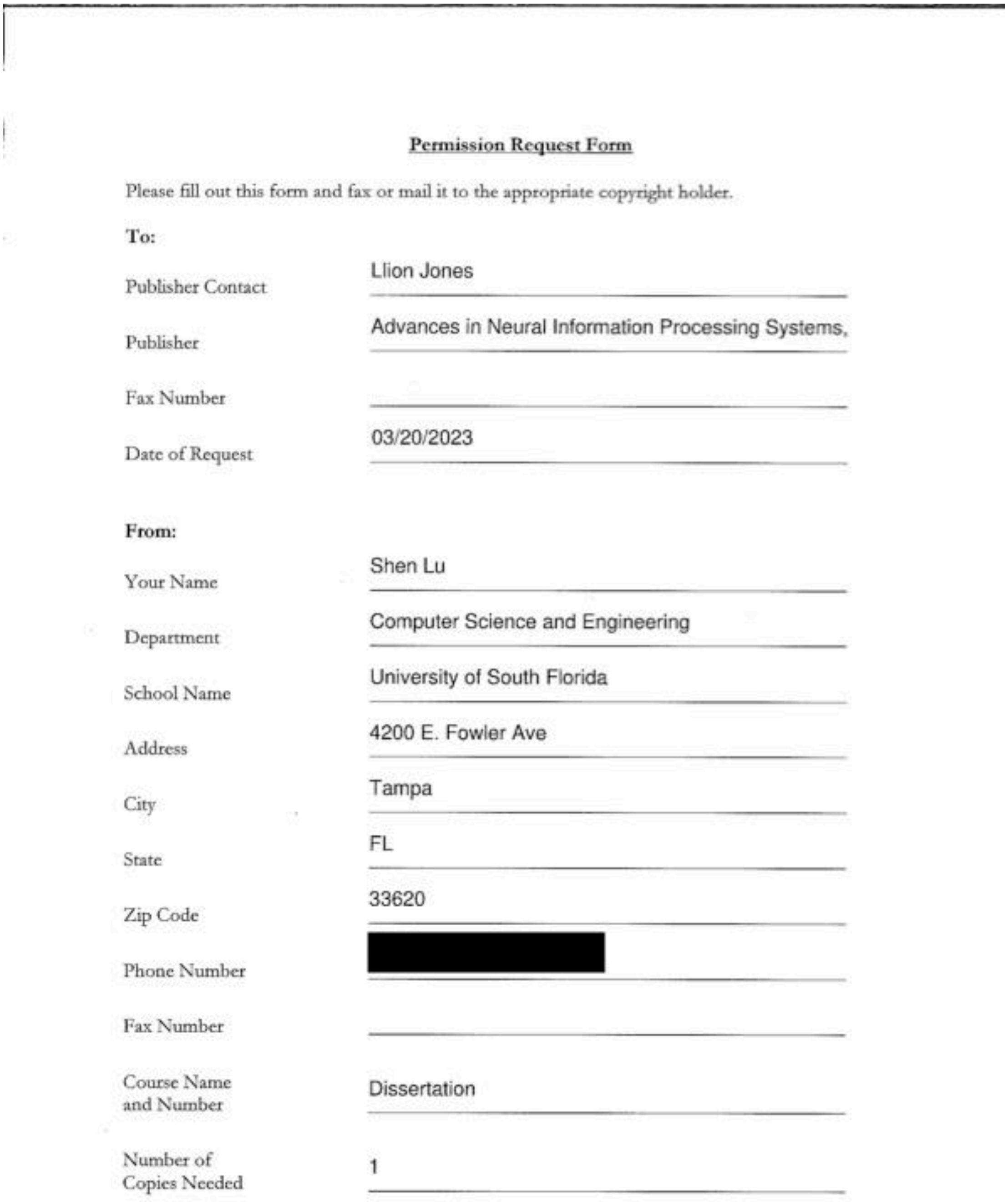
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Copy Markdown to Clipboard Copy ACL to Clipboard

The ACL Anthology is managed and built by the ACL Anthology team of volunteers.

https://aclanthology.org/N19-1423/ Page 4 of 5

The permission below is for the use of Figure 7.3. The Transformer Model Architecture and Figure 7.4. Attention Mechanism in Chapter 7.



Permission Request Form

Please fill out this form and fax or mail it to the appropriate copyright holder.

To:

Publisher Contact Llion Jones

Publisher Advances in Neural Information Processing Systems,

Fax Number

Date of Request 03/20/2023

From:

Your Name Shen Lu

Department Computer Science and Engineering

School Name University of South Florida

Address 4200 E. Fowler Ave

City Tampa

State FL

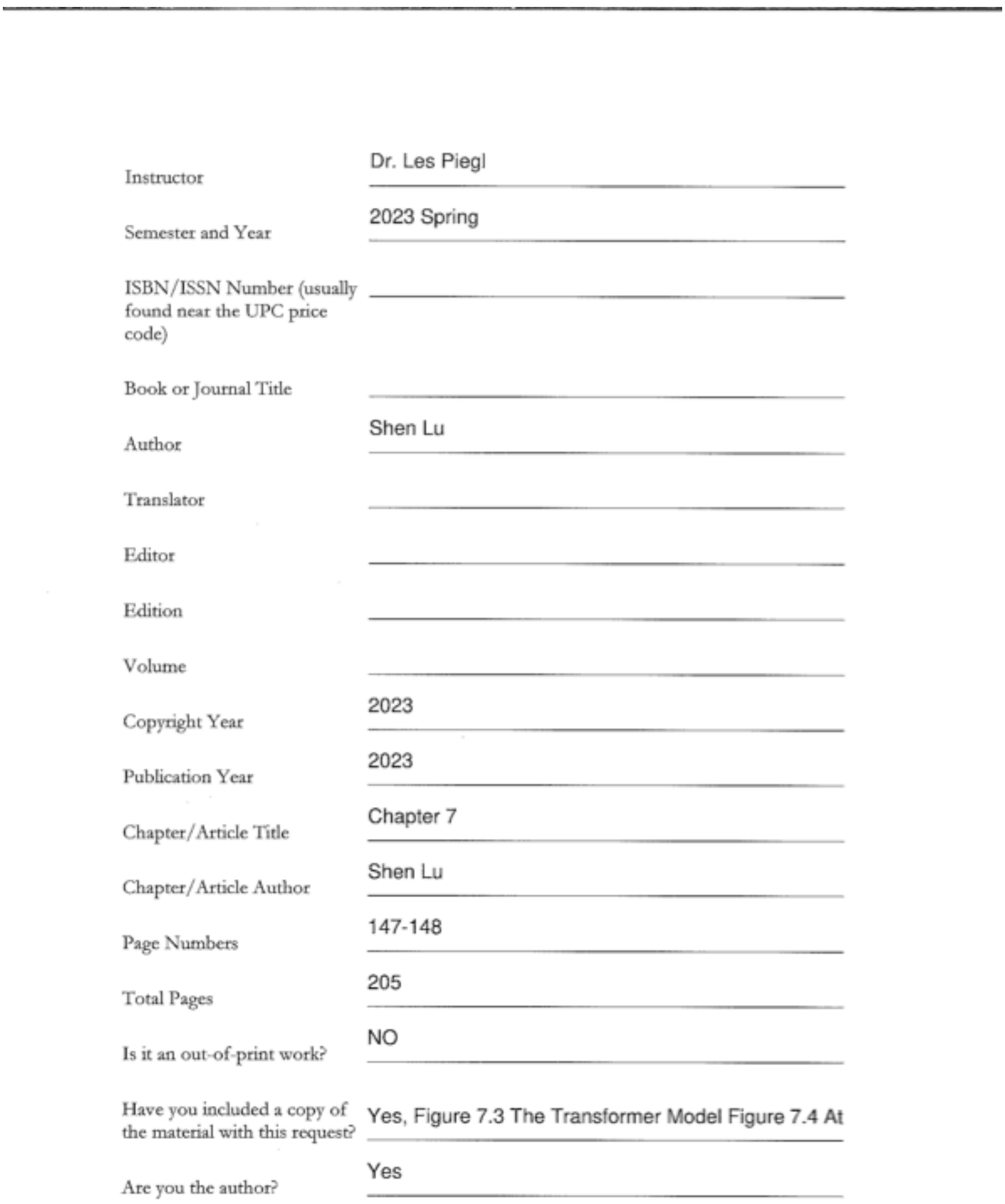
Zip Code 33620

Phone Number [redacted]

Fax Number

Course Name and Number Dissertation

Number of Copies Needed 1



Instructor	Dr. Les Piegl
Semester and Year	2023 Spring
ISBN/ISSN Number (usually found near the UPC price code)	
Book or Journal Title	
Author	Shen Lu
Translator	
Editor	
Edition	
Volume	
Copyright Year	2023
Publication Year	2023
Chapter/Article Title	Chapter 7
Chapter/Article Author	Shen Lu
Page Numbers	147-148
Total Pages	205
Is it an out-of-print work?	NO
Have you included a copy of the material with this request?	Yes, Figure 7.3 The Transformer Model Figure 7.4 At
Are you the author?	Yes

Print Name Llion Jones

Signature [redacted]

Date 2023-03-24