

5-2016

Information overload in structured data

Pinar Yanardag Delul
Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/open_access_dissertations



Part of the [Computer Sciences Commons](#)

Recommended Citation

Yanardag Delul, Pinar, "Information overload in structured data" (2016). *Open Access Dissertations*. 729.
https://docs.lib.purdue.edu/open_access_dissertations/729

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

**PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance**

This is to certify that the thesis/dissertation prepared

By Pinar Yanardag Delul

Entitled

INFORMATION OVERLOAD IN STRUCTURED DATA

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Jennifer Neville

Chair

SVN Vishwanathan

Buster Dunsmore

Elena Grigorescu

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Jennifer Neville, SVN Vishwanathan

Approved by: Sunil Prabhakar / William J. Gorman

Head of the Departmental Graduate Program

4/25/2016

Date

INFORMATION OVERLOAD IN STRUCTURED DATA

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Pinar Yanardag Delul

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2016

Purdue University

West Lafayette, Indiana

“If you look at the cycles of the moon, it starts as a thin crescent and then gradually waxes until it becomes full; then it gradually wanes back into another crescent and then it is gone. The moon reflects sunlight like humans reflect information. We wax and wane and when we become full moons, our egos are full. We think we have this knowledge when in fact, the information we have is pure. And how it reflects or shines off of us, is something we take credit for as though the moon could take credit for its brightness when, in fact, it is only reflecting light from the sun. We have to understand that we are egoless just as the moon is without light. It and we are simply reflectors. The ego is not responsible for the information. It can reflect the information in creative ways, but the information itself is pure.” –Maynard James Keenan

ACKNOWLEDGMENTS

Throughout my career, I was extremely lucky to be surrounded by so many exceptional and inspirational people that pass their reflections over me. I would like to acknowledge these people who have made a big impact on my life and study, and this dissertation would not have come to existence without them.

First, I would like to thank my advisor Prof. S.V.N. (Vishy) Vishwanathan for his advice and support. This dissertation would not have been possible without freedom and encouragement he has given me over the last five years I spent at Purdue. As my academic advisor, he has tried hard to turn a spoiled open source developer into an independent researcher. During the first few years of my PhD when I impatiently jumped into coding vague ideas (and failing), he has patiently taught me the habit of *thinking*. It has always been clear how much he wanted me to succeed—for which he has constantly provided me the right combination of guidance and freedom. As a true ADHD, I got distracted countless number of times during my PhD, or as Vishy puts it, I “jumped from one idea to another”. However, instead of forcing me to stay on one particular direction, he always respected and adapted to the different research areas I got drawn into—which helped me to find my own way as a researcher. There are so many things that Vishy has done for me, for which I would be forever grateful.

I would also like to express my gratitude to Prof. Jennifer Neville for becoming my co-advisor after Vishy has switched to UCSC, and for teaching me various data mining tricks in CS573.

I feel extremely lucky to cross paths with Prof. Buster Dunsmore. I cannot thank him enough for not only being in my qualifying, preliminary and final exam committees, but also for all the support he has given me during my PhD. I am grateful for his genuinely kind nature, and for his willingness to help students. He has been a great role model for me and for many other students in the department.

I am grateful to all faculty members at the Purdue Department of Computer Science for making our department a top-notch one in the nation. I owe a special thanks to Prof. Chris Clifton for mentoring me during the first two years of my PhD, and for his great lectures on database systems in CS541. I thank to Prof. Elena Grigorescu for being in my preliminary and final examination committees, and for her great lectures on theoretical computer science in CS590-CTT; Prof. Vernon Rego for serving in my qualifying exam, for his positive nature, and for his great lectures on simulation systems in CS543; Prof. Dan Goldwasser for serving in my qualifying exam, for several inspiring discussions on natural language processing, and for his great lectures on machine learning in CS578; Prof. Kihong Park for helping me on several departmental issues, and for his excellent lectures on operating systems in CS503; Prof. Ananth Grama for his inspiring lectures on parallel computing in CS525; Prof. Voicu Popescu for helping me during my qualifying exam scheduling and Prof. Mathias Payer for his great lectures on systems security in CS590-SEC. I am also indebted to the staff at both Computer Science and Statistics Departments, including Renate Mallus-Medot, Shaun Ponder, Sandra Freeman, Katherine Huseman, Pam Graf and Dr. William Gorman.

I would also like to deeply appreciate the advisors I had before starting my PhD. I would like to thank to my undergraduate advisor Prof. Necdet Yucel for introducing me to open source community, encouraging me to give talks and presentations at open source conferences, and helped me to collaborate with several open source projects. I would like to thank to Prof. Ethem Alpaydin who introduced me with the concept of machine learning for the first time, and for guiding me through my master's thesis at Bogazici University. I also deeply express my gratitude for Prof. Reyyan Ayfer who always encouraged me to pursue my dreams and for being such an excellent role model. I would also thank to Prof. Suzan Uskudarli, for several interesting discussions on social networks, and for being such a positive and genuinely nice person.

I spent two wonderful summers at VMware where I had the opportunity to work with excellent engineers. I first would like to thank to my recruiter, Matthew Wendorf for discovering me from LinkedIn, and connecting me with Distributed Resource Management

Group. I thank to Rean Griffith for mentoring me, teaching me several things related to systems and machine learning, and giving me several invaluable advices for my PhD. I also thank to many great engineers I worked at VMware, including Anne Holler, Ravi Soundararajan, Xiaoyun Zhu, Adarsh Jagadeeshwaran, and many others. I also had the chance to spend a great summer at Amazon, where I worked with many great engineers on interesting machine learning problems. I thank to my manager, Grant Emery for being such an inspiring and helpful person and my mentor Michael Quinn for being a top-notch programmer and teaching me several cool tricks.

During my PhD, I have built friendships that I hold closely to my heart. I would like to thank to Pelin Angin for being a genuinely nice person, and for helping me during several aspects of my PhD; Amani Abu Jabal for being such a good friend, for studying several exams and pulling all-nighters at Lawson Commons with me; Parameswaran Raman for always being there to talk to, and for all the gossip; Hyokun Yun for always sharing his expertise on machine learning, and helping to implement Pitman-Yor processes; Joon Hee Choi for all the fun, and sharing his expertise on tensor factorizations; Nan Ding, Vasil Denchev and Bin Shen for always answering any question I might have in the lab. I would like to thank to my UCSC gang at Santa Cruz, where I had some of the best times of my PhD. I owe a special thanks to Paris Miri for welcoming me at her home, and for being such a funny and nice person. I thank to all my friends at E2-489 for all the fun, grills, workouts and video games. I thank to Andre Ignat for working late with me several nights at Lulu's, for his great food and coffee, allowing me to steal his cigarettes, and for reviewing and fixing several parts of my KDD submission; Ehsan Amid for sharing his expertise in online learning, helping several derivations of my KDD submission and for his delicious chicken kebabs; Holakou Rahmanian for introducing me to several great Persian singers, and playing Mortal Combat with me; Sriram Srinivasan for teaching me Hindi and for singing Bollywood songs with me while pulling all nighters at the lab.

Lastly, I would like to thank my family for their endless love and encouragement. Everything was possible due to their strong support. I am indebted to my grandmother Ikbal, my mother Tulin, my uncle Turker and my brother Ozgur for their continuous support and

encouragement over the years. Finally, my beloved husband Selem who didn't hesitate to move from Istanbul to Mid-West with me, is the unsung hero behind this PhD. Your encouragement and support was unwavering, and I dedicate this dissertation to you.

TABLE OF CONTENTS

| | Page |
|--|-----------|
| LIST OF TABLES | x |
| LIST OF FIGURES | xii |
| ABSTRACT | xiv |
| 1 INTRODUCTION | 1 |
| 1.1 Information Overload in Graphs | 1 |
| 1.2 Information Overload in Text | 6 |
| 1.3 Thesis Statement and Main Contributions | 8 |
| 1.4 Thesis Outline | 11 |
| 1.5 Related Publications | 12 |
| 2 BACKGROUND | 14 |
| 2.1 Graph Kernels | 14 |
| 2.1.1 Notation | 14 |
| 2.1.2 Graph Kernels Based on Subgraphs | 16 |
| 2.1.3 Graph Kernels Based on Subtree Patterns | 16 |
| 2.1.4 Graph Kernels Based on Random-walks | 17 |
| 2.1.5 R-convolution Framework | 18 |
| 2.2 Submodularity | 18 |
| 2.3 Word Embedding Models | 19 |
| 2.3.1 Continuous Bag-of-words | 20 |
| 2.3.2 Skip-gram | 21 |
| I Information Overload in Graphs | 23 |
| 3 DEEP GRAPH KERNELS | 24 |
| 3.1 Motivation | 24 |
| 3.2 Methodology | 25 |
| 3.2.1 Sub-structure Similarity via Edit-distance | 25 |
| 3.2.2 Sub-structure Similarity via Learning | 27 |
| 3.2.3 Deep Graph Kernels | 28 |
| 3.2.4 Deep String Kernels | 30 |
| 3.3 Related Work | 31 |
| 3.4 Experiments | 32 |
| 3.4.1 Experimental Setup | 32 |

| | Page |
|--|-----------|
| 3.4.2 Datasets | 33 |
| 3.4.3 Parameter Selection | 38 |
| 3.4.4 Computational Cost | 38 |
| 3.4.5 Results for Graph Kernels | 39 |
| 3.4.6 Results for String Kernels | 40 |
| 3.5 Conclusions | 43 |
| 4 SMOOTHED GRAPH KERNELS | 44 |
| 4.1 Motivation | 44 |
| 4.2 Methodology | 45 |
| 4.2.1 Smoothing Multinomial Distributions | 45 |
| 4.2.2 Structural Smoothing | 47 |
| 4.2.3 Pitman-Yor Smoothing | 51 |
| 4.2.4 Other Smoothed Graph Kernels | 54 |
| 4.3 Related Work | 55 |
| 4.4 Experiments | 55 |
| 4.4.1 Experimental Setup | 56 |
| 4.4.2 Datasets | 56 |
| 4.4.3 Analyzing Feature Space | 56 |
| 4.4.4 Parameter Selection | 57 |
| 4.5 Results | 61 |
| 4.6 Conclusions | 65 |
| | |
| II Information Overload in Text | 66 |
| | |
| 5 A SUBMODULAR RECOMMENDER SYSTEM FOR REDDIT | 67 |
| 5.1 Introduction | 67 |
| 5.2 Methodology | 73 |
| 5.2.1 Coverage | 74 |
| 5.2.2 Covering Subreddits | 76 |
| 5.2.3 Covering Topics | 78 |
| 5.2.4 Personalization | 81 |
| 5.3 Related Work | 85 |
| 5.4 Experiments | 86 |
| 5.4.1 Quantitative Experiments | 87 |
| 5.4.2 Qualitative Experiments | 89 |
| 5.5 Conclusions | 94 |
| | |
| 6 A SUBMODULAR FRAMEWORK TO SUMMARIZE TED TALKS | 95 |
| 6.1 Motivation | 95 |
| 6.2 Methodology | 96 |
| 6.2.1 Designing the Relevancy Component | 99 |
| 6.2.2 Designing the Diversity Component | 101 |

| | Page |
|--|------|
| 6.3 Related Work | 103 |
| 6.4 Experiments | 103 |
| 6.4.1 Datasets | 104 |
| 6.4.2 Evaluation | 104 |
| 6.4.3 Results | 106 |
| 6.5 Conclusions | 108 |
| 7 CROWDSOURCED RESOURCE-SIZING OF VIRTUAL APPLIANCES . . | 109 |
| 7.1 Motivation | 109 |
| 7.2 Datasets | 111 |
| 7.3 Methodology | 113 |
| 7.3.1 Feature Selection | 114 |
| 7.3.2 Exploratory Data Analysis | 116 |
| 7.4 Experiments | 119 |
| 7.5 Related Work | 123 |
| 7.6 Conclusions | 124 |
| 8 FUTURE WORK | 126 |
| 8.1 A Submodular Approach for Graph Sampling | 126 |
| 8.1.1 Proposed Solution | 127 |
| 8.1.2 Preliminary Experiments | 129 |
| 8.2 A Submodular Approach for Color Palette Generation | 130 |
| 8.2.1 Proposed Solution | 133 |
| 8.2.2 Preliminary Experiments | 136 |
| 9 CONCLUSIONS | 138 |
| LIST OF REFERENCES | 140 |
| VITA | 149 |

LIST OF TABLES

| Table | Page |
|---|------|
| 2.1 Common characteristics of different graph kernel families. | 17 |
| 3.1 Properties of the bioinformatics datasets used in graph kernel experiments. . | 33 |
| 3.2 Properties of the social network datasets used in graph kernel experiments. . | 34 |
| 3.3 Comparison of classification accuracy of the Graphlet kernel and Deep Graphlet kernel on social network datasets. | 39 |
| 3.4 Comparison of classification accuracy of several graph kernels with their deep variants. | 41 |
| 3.5 Comparison of classification accuracy of Ramon & Gärtner, p -Random-walk, and Random-walk graph kernels. | 42 |
| 3.6 Classification accuracy for string kernel experiments. | 43 |
| 4.1 Graphlet kernel feature statistics for benchmark datasets that shows median, maximum and minimum number of features. | 57 |
| 4.2 Weisfeiler-Lehman kernel feature statistics for benchmark datasets that shows median, maximum and minimum number of features. | 59 |
| 4.3 Shortest-Path kernel feature statistics for benchmark datasets that shows median, maximum and minimum number of features. | 59 |
| 4.4 List of parameters used in experiments for smoothing experiments. | 60 |
| 4.5 Comparison of classification accuracy between base kernels and smoothed kernels. | 62 |
| 4.6 Runtime for constructing the DAG and smoothing the kernels. | 63 |
| 4.7 Comparison of classification accuracy for state-of-the-art graph kernels. . . | 64 |
| 5.1 A list of sample posts and their subreddits. | 70 |
| 5.2 Top ten most similar subreddits for a given subreddit. | 75 |
| 6.1 ROUGE measures for different methods. | 107 |
| 7.1 List of environments and their characteristics. | 111 |
| 7.2 List of example categories from profiler logs. | 112 |

| Table | Page |
|---|------|
| 7.3 The list of features that are discovered by mutual information test. | 113 |
| 7.4 Top triple features with high mutual information. | 114 |
| 7.5 Physical memory usage on Linux platform with 70/30 dataset. | 121 |
| 7.6 Physical memory usage on Windows platform with 70/30 dataset. | 122 |
| 7.7 Physical memory usage on Linux platform with cross-validation. | 123 |
| 7.8 Physical memory usage on Windows platform with cross-validation. | 124 |
| 8.1 Comparison of classification accuracy for the Graphlet kernel and Submodular Graphlet kernel. | 130 |

LIST OF FIGURES

| Figure | Page |
|---|------|
| 1.1 An example communication graph on Reddit. | 2 |
| 1.2 Exponential growth and power-law behavior on graphlet kernels. | 4 |
| 1.3 An illustration of the diagonal dominance problem on existing graph kernels. | 5 |
| 1.4 An example dependency schema of graphlets. | 6 |
| 2.1 Sample chemical compounds that are represented as graphs. | 15 |
| 2.2 An illustration of induced, non-isomorphic induced sub-graphs of size $k \leq 5$ | 16 |
| 2.3 Architecture for the Continuous Bag-of-words and Skip-gram models. | 22 |
| 3.1 Undirected edit-distance graph for graphlets of size $k \leq 5$ | 26 |
| 3.2 An illustration of the learned shortest-path sub-structures. | 27 |
| 3.3 An example of sub-structure regularity in graphlet sub-structure space. | 29 |
| 3.4 An example protein from TIM beta/alpha-barrel fold. | 36 |
| 3.5 Classification accuracy vs. edge noise on different datasets using Graphlet kernel and Edit-distance Graphlet kernel. | 37 |
| 3.6 Classification accuracy vs. edge noise on different datasets using Graphlet kernel and Deep Graphlet kernel. | 37 |
| 4.1 An example DAG for Graphlet kernels. | 48 |
| 4.2 An illustration of smoothing on Graphlet kernels. | 49 |
| 4.3 An illustration of table assignment for Pitman-Yor smoothing. | 51 |
| 4.4 Comparison of accuracy vs. noise for base graph kernels. | 58 |
| 5.1 Default list of subreddits on Reddit. | 68 |
| 5.2 Exponential growth of subreddits and user base on Reddit. | 69 |
| 5.3 A diagram that shows different components of our framework. | 71 |
| 5.4 Hot-score as a function of increasing upvote and downvote difference. | 73 |
| 5.5 An illustration of topic-based framework. | 77 |

| Figure | Page |
|---|------|
| 5.6 Average click position for baselines and our proposed method. | 85 |
| 5.7 Coverage of subreddits based on hypothetical user. | 88 |
| 5.8 Learned weights by Exponential Gradient and Decaying Past algorithms. . . | 90 |
| 5.9 Interestingness and diversity results from user study. | 92 |
| 5.10 Discoverability and personalization results from user study. | 93 |
| 6.1 A sample speaker-based word cloud. | 97 |
| 6.2 A sample audience-based word cloud. | 97 |
| 6.3 Top topics and themes of TED talks. | 105 |
| 7.1 Visualization of mutual information on different datasets. | 115 |
| 7.2 PCA analysis on Windows platform. | 117 |
| 7.3 PCA analysis on Linux platform. | 118 |
| 7.4 3D plot with a heatmap based on physical memory usage on Linux platform. | 119 |
| 7.5 3D plot with a heatmap based on physical memory usage on Windows platform. | 120 |
| 8.1 A sample decomposition on MUTAG dataaset. | 127 |
| 8.2 A sample decomposition of graphlets with normalized frequencies. | 128 |
| 8.3 K-means on “The Arnolfini Portrait” by Jan van Eyck. | 131 |
| 8.4 A comparison of extracted color palettes for “The Arnolfini Portrait” by Jan van Eyck. | 137 |

ABSTRACT

Yanardag Delul, Pinar PhD, Purdue University, May 2016. Information Overload in Structured Data. Major Professors: S.V.N. Vishwanathan, Jennifer Neville.

Information overload refers to the difficulty of making decisions caused by too much information. In this dissertation, we address information overload problem in two separate structured domains, namely, graphs and text.

Graph kernels have been proposed as an efficient and theoretically sound approach to compute graph similarity. They decompose graphs into certain sub-structures, such as subtrees, or subgraphs. However, existing graph kernels suffer from a few drawbacks. First, the dimension of the feature space associated with the kernel often grows exponentially as the complexity of sub-structures increase. One immediate consequence of this behavior is that small, non-informative, sub-structures occur more frequently and cause information overload. Second, as the number of features increase, we encounter sparsity: only a few informative sub-structures will co-occur in multiple graphs. In the first part of this dissertation, we propose to tackle the above problems by exploiting the dependency relationship among sub-structures. First, we propose a novel framework that *learns* the latent representations of sub-structures by leveraging recent advancements in deep learning. Second, we propose a general smoothing framework that takes *structural* similarity into account, inspired by state-of-the-art smoothing techniques used in natural language processing. Both the proposed frameworks are applicable to popular graph kernel families, and achieve significant performance improvements over state-of-the-art graph kernels.

In the second part of this dissertation, we tackle information overload in text. We first focus on a popular social news aggregation website, Reddit, and design a submodular recommender system that tailors a *personalized* frontpage for individual users. Second, we propose a novel submodular framework to summarize videos, where both transcript and

comments are available. Third, we demonstrate how to apply filtering techniques to select a small subset of informative features from virtual machine logs in order to predict resource usage.

1 INTRODUCTION

Information overload refers to the difficulty of making decisions caused by too much information. An example of information overload can be found as early as 1st century A.D., when the Roman rhetorician and writer Seneca the Elder commented that “the abundance of books is distraction” [1]. Today, Seneca’s distraction that is caused by information overload is present in nearly every domain, from online news aggregators to social networks, question/answering platforms to system logs, and even in biological networks.

Information overload affects users on the web as well as systems, as both are interfered with their ability to filter and process substantial amount of new observations. In the recent years, there has been a significant increase in the amount of content generated daily, which forces both users and systems to cope with information overload. For instance, there are over 850 thousand communities on Reddit [2], over 500 million tweets generated daily on Twitter [3], and hundreds of gigabytes of system logs generated by virtual machines on VMware’s cloud services [4]. One of the common causes of information overload problem is the lack of methods for accurately comparing and processing different kinds of observations. In this dissertation, we address these challenges in two separate domains: (1) information overload in graphs and (2) information overload in text.

1.1 Information Overload in Graphs

Graphs are universal data structures that model a network of relationships between objects. In particular, they offer a flexible and natural way to represent data in various domains, including social networks, bioinformatics, chemoinformatics and robotics. We list some of the important fields of application for graphs in the following.

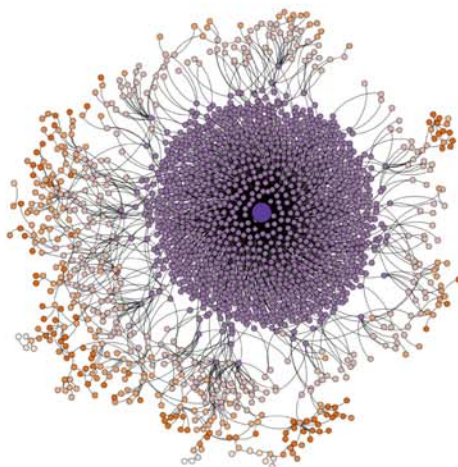


Figure 1.1. A graph of a random post on `http://reddit.com/r/askreddit` where nodes represent users and edges represent whether two users interact by responding to each others comments.

Social Networks An important source of graph structured data is social network analysis. In social networks, users are represented as nodes, and the interaction between them are represented as edges (see Figure 1.1¹ for the graph of a random discussion thread from Reddit²). Graphs can be used to encode various kind of relationships. For instance, friendship graphs model whether people know each other [5], influence graphs model how certain individuals influence the behavior of others [6], and collaboration graphs model how two individuals work together, such as producing a scientific publication [7].

Bioinformatics Another major source of graph structured data is molecular biology. In this area, graphs are used for modeling molecular structures such as RNAs and proteins, or modeling biological networks such as protein-protein interaction [8] where the task is to assign functions to these structures and networks.

Chemoinformatics Graphs are also frequently used to model molecules in chemistry. In chemoinformatics, molecules are modeled as a graph where nodes represent atoms and

¹Image is generated with ForceAtlas-2 layout using Gephi: <http://gephi.github.io>.

²Reddit is a popular content-aggregation website: <http://reddit.com>.

edges represent bonds between them. This approach is especially used in chemoinformatics to model characteristics of molecules from their graph structures, such as toxicity [9].

Systems This is another area where graph structures are frequently used. For instance, a computer program can be represented as a graph, so-called Program Dependence Graph (PDG) [10] where nodes represent statements or expressions, and edges represent data values or control conditions which execution of the program depends on. This representation is then used to perform several tasks including optimizing compilers and improve parallelism [10].

One of the central algorithmic problems involving graphs is measuring the similarity between two graphs. To illustrate one example where graph similarity can be useful, consider the problem of identifying a sub-community (also referred as *subreddits*) on Reddit. To tackle this problem, one can represent an online discussion thread as a graph in which nodes represent users, and edges represent whether two users interact, for instance, by responding to each others comments (see Figure 1.1). The task is then to predict which sub-community a discussion thread belongs to by analyzing its communication graph. Similarly, in bioinformatics, one might be interested in the problem of identifying whether a given protein is an enzyme or not. In this case, the secondary structure of a protein is represented as a graph in which nodes correspond to atoms and edges represent the chemical bonds between atoms. If the graph structure of the protein is similar to the known enzymes, one can predict that the given graph is also an enzyme [11]. Graph similarity can also be utilized in systems security, where one might represent computer programs as program dependence graphs and aim to predict whether a given program has security vulnerabilities such as buffer overflows.

Meanwhile graphs offer a flexible structure to represent various type of relationships, there is no universally accepted similarity function on graphs that can be computed efficiently [12]. Perhaps the most simplest approach to identify common parts of two graphs is to consider the set of all subgraphs between two graphs. However, given a graph with n

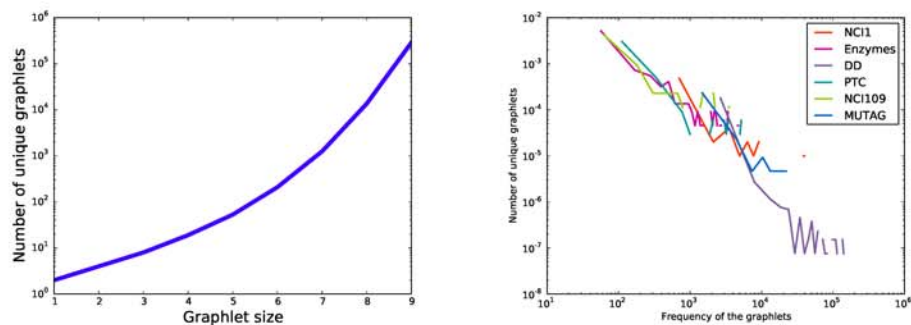


Figure 1.2. (a) Exponential growth behavior of feature space in graphlets up to $n \leq 9$. (b) Power-law behavior on graphlet kernel for different datasets.

nodes, there are 2^n possible subsets of nodes one needs to consider. Hence, the search space is exponential in the size of graphs and makes this approach prohibitively expensive [13].

Graph kernels have recently been introduced as an efficient and theoretically sound approach to address this problem [11]. Graph kernels approach the problem by defining a *kernel* between two graphs which captures the semantics of the structure while being computationally tractable. Roughly speaking, the majority of graph kernels are instances of a general framework called R-convolution [14] where the key idea is to recursively decompose structured objects into *sub-structures* and define local kernels between them. However, R-convolution based graph kernels suffer from a few drawbacks due to the information overload arise in the feature space. First, the dimension of the feature space associated with a kernel often grows exponentially. Figure 1.2 (a) illustrates the growth of number of *graphlets*, a popular sub-structure type that is used for decomposing graphs [15, 16], which are defined as induced, non-isomorphic sub-graphs of size k . One immediate consequence of this exponential behavior is that lower-level, non-informative sub-structures occur more frequently and cause an information overload problem. That is, the higher-level and informative sub-structures are being dominated due to the frequently occurring lower-level sub-structures. In fact, the frequency of occurrence of certain sub-structures can be modeled as a power of the frequency of occurrence of other sub-structures (see Figure 1.2 (b)). Second, as the number of features increase, we encounter the sparsity problem: only a few

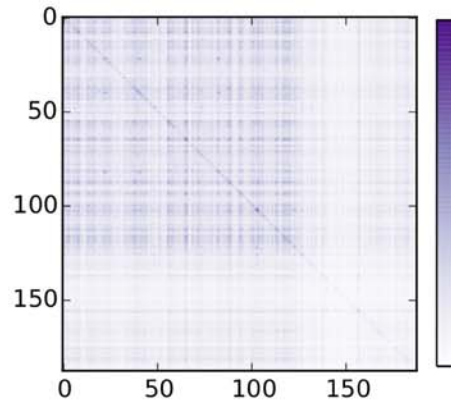


Figure 1.3. Kernel matrix K for Weisfeiler-Lehman subtree kernel on the MUTAG dataset [46]. Entry $K_{i,j}$ encodes the similarity between graph i and graph j and the color map encodes the degree of the similarity (darker color means higher similarity). Notice that in both cases we observe a block matrix since the first 125 entries of the matrix correspond to graphs which are labeled $+1$ and the remaining entries correspond to graphs which belong to the -1 class.

of sub-structures will be common across graphs. This leads to the diagonal dominance problem, that is, a given graph is similar to itself but not to any other graph in the dataset. Figure 1.3 illustrates such a kernel matrix using Weisfeiler-Lehman subtree kernels [17] where the diagonal dominance can be visually observed. Ideally, one would expect a kernel matrix where all entries belonging to a class are similar to each other, and dissimilar to anything else.

An important observation is that sub-structures in graph kernels are often related to each other. Let us consider an example on graphlets. Graphlets exhibit a strong dependence relationship, that is, size $k + 1$ graphlets can be derived from size k graphlets by addition of nodes or edges (similarly, size k graphlets can be recovered by deletion of nodes or edges from size $k + 1$ graphlets). For instance, G_{39} can be derived from G_{15} by adding a new node and an edge (see Figure 1.4).

In the first part of this dissertation, we aim to address above problems by considering the dependency relationship between sub-structures, and alleviate the information overload problem by incorporating the partial similarity between sub-structures into the kernel computation.



Figure 1.4. Dependency schema of a set of graphlets of size $k \in \{3, 4, 5\}$ where G_{39} can be derived from G_{15} (similarly, G_{15} can be derived from G_7) by adding a new node and an edge.

1.2 Information Overload in Text

In the second half of this dissertation, we investigate a large variety of information overload problems related to text. Text is a natural way to represent information, and has been utilized in various domains including natural language processing (NLP), social networks, news aggregation platforms and recommender systems [18, 19]. In the recent years, there has been a substantial increase in amount of content generated daily in these areas. We list some of the examples of information overload problem related to text in the following.

Micro-blogs and blogs In the recent years, micro-blogging services became a popular medium to spread breaking news, share personal updates, promote opinions and tracking real time events. Micro-blogging is a form of blogging where posts typically consist of short content such as quick comments, phrases, URLs, or media, such as images and videos. Due to their popularity, users are easily overwhelmed by the large amount of incoming messages. For instance, over 500 million tweets are generated daily on Twitter where users have to cope up with a substantial information overload [3]. A similar trend also appear in blogosphere, where users are faced with an endless torrent of information from various sources [68].

News aggregation websites News aggregation websites such as Reddit face with information overload both in terms of posted content, and in terms of user base. Reddit is one of the largest community-driven content aggregation websites where approximately, 6%

of online adults are estimated to be Reddit users [20]. Due to the increasing number of users and interests, there are more than 850 thousand communities on Reddit devoted to various topics. Given that thousands of new threads and discussion topics generated hourly on thousands of communities, finding relevant and interesting content for users becomes a difficult task.

Documents and videos Another major source of information overload is text documents such as books and scientific articles. Transcripts of videos such as Youtube or video lectures such as TED talks [21] form a special case of documents. Similar to the previous cases, it is difficult for users to cope up with incoming torrent of documents and videos. For instance, there are over 300 hours of video uploaded per minute to Youtube [22].

System logs System logs are being employed in many software applications as files that record important events such as failures, warnings and system-specific messages. These log files are often used by system administrators to monitor the system or analyzed by software applications for automatically discovering patterns in the system. With the increasing number of logs generated hourly, a natural information overload arises in this context. For instance, Georgia Institute of Technology generates over four terabytes of data daily by its network [23]. This issue is particularly important in cloud computing systems where a single system can be setup to manage hundreds of thousands of virtual machines [4].

An important observation is that there is often a substantial overlap in content among these resources, and one can reduce the information overload by selecting informative and diverse observations. A second observation which especially holds in social network and news aggregation websites is to consider *personalized* preference of the users, where we can design a system that not only provides informative and diverse observations, but also satisfies the specific information needs of the users.

In the second part of this dissertation, we aim to address these problems using *submodularity*, a discrete optimization area that exhibits a natural diminishing returns property.

In particular, we design and use submodular objective functions in which we can provide efficient and near-optimal solutions for overcoming information overload problem.

1.3 Thesis Statement and Main Contributions

The core of this dissertation revolves around the following statement:

In many practical applications, including (1) graph mining and (2) text mining, it is important to consider the information overload problem. By exploiting the problem structure, one can design algorithms to select among most informative but diverse observations to produce robust and accurate solutions.

Throughout this dissertation, we evaluate this thesis statement on a variety of problems. In particular, the development of this hypothesis is separated into two components: (1) Information Overload in Graphs (2) Information Overload in Text.

Information Overload in Graphs In the first part of the dissertation, we consider information overload problem in graphs. We focus on graph comparison task, and propose novel frameworks to address information overload arise in feature space.

First, we propose a novel framework that *learns* the latent representations of sub-structures by leveraging the co-occurrence relationship of the features. Our contributions in this work are as follows:

- We propose a general framework that is applicable to any graph kernel that is an instance of R-convolution kernels. In particular, we embed sub-structures into a d -dimensional space by using the latest advancements in language modeling and deep learning.
- We demonstrate our framework on three popular graph kernels, namely Graphlet kernels, Weisfeiler-Lehman subtree kernels, and Shortest-Path kernels and achieve significant improvements on several benchmark datasets.

- We discuss the connection of our framework to R-convolution kernels and apply our framework to derive deep variants of *string* kernels.
- We introduce several new and large graph kernel datasets in social network domain, associated with novel tasks such as *community prediction*.
- We show that it is possible to perform analogy task on graphs similar to word analogy tasks.
- To best of our knowledge, our framework is among the first to introduce the usage of word embedding and deep learning techniques in graph comparison tasks, and leads to novel deep learning frameworks that automatically discover important graph patterns for graph mining tasks.

Second, we propose a general smoothing framework for graph kernels by taking *structural* similarity into account. Our framework is inspired by state-of-the-art smoothing techniques used in NLP. However, unlike NLP applications that primarily deal with strings, we show how one can apply smoothing to a richer class of inter-dependent sub-structures that naturally arise in graphs. Our contributions in this work are as follows:

- We propose a general framework that is applicable to any graph kernel that is an instance of R-convolution kernels by taking the structural similarity into account.
- We extend state-of-the-art smoothing techniques in natural language processing to structured objects by defining a Directed Acyclic Graph (DAG) which encodes the dependency relationships between sub-structures.
- We demonstrate our framework on three popular graph kernels, namely Graphlet kernels, Weisfeiler-Lehman subtree kernels, and Shortest-Path kernels and achieve significant improvements on several benchmark datasets.
- We derive a Bayesian extension of our framework using Pitman-Yor process, thereby leading to novel graph kernel extensions.

Information Overload in Text In the second part of this dissertation, we focus on information overload arise in text applications, namely, social news aggregators, video transcripts, and system logs.

First, we focus on information overload in social news aggregation websites. In particular, we focus on Reddit, one of the largest community-driven content aggregation websites that is commonly referred as the *frontpage of the Internet*. We propose a framework that tailors a *personalized* frontpage for individual users. Our contributions in this work are as follows.

- We propose a submodular recommender system to curate a frontpage for users in which we provide a near-optimal and efficient solution.
- We project subreddits into topical space, and discover novel relationships between communities.
- We formalize a two-step, personalized notion of coverage by learning the preference of individual users towards topics.
- We evaluate our algorithm quantitatively on a large-scale real voting data collected from Reddit users, and as well as conducting a user study.

Second, we focus on information overload arise in documents and video transcripts. In particular, we propose a novel summarization framework to summarize TED talks, a non-profit organization devoted to “Ideas Worth Spreading” [21]. We formulate our objective function as a submodular framework that balances coverage and diversity and reduces information overload by avoiding to select redundant content. Our contributions in this work are as follows.

- We propose a coverage function covers the ideas that speaker is promoting while also leveraging aspects that audience is focused on.
- We propose a novel diversity function that avoids redundancy by using the latest advancements in deep learning and neural language processing. In particular, our

framework evaluates diversity of selected sentences in the summary in terms of latent dimensions of the words.

- We demonstrate a novel application by summarizing TED talks that is applicable to any video summarization task where an audience is available, such as Youtube.

Third, we focus on information overload in system logs. In particular, we use a population of virtual machines from VMware [4], and demonstrate how to apply feature filtering and selection techniques to reduce the information overload and identify important features. Our contributions in this work are as follows.

- We present and quantitatively validate feature filtering and selection techniques to identify important features to use as model inputs. Our feature-selection results are corroborated and validated by domain experts from VMware.
- We demonstrate how to construct a crowdsourced model of memory usage for the virtual machines and provide sizing recommendations for the virtual appliances.

1.4 Thesis Outline

This dissertation is structured as follows:

Chapter 2. Background In this chapter, we review related background material, including graph kernels, submodularity and word embeddings.

Part I: Information Overload in Graphs Chapters 3-4. In the first part of this dissertation, we first introduce a novel framework that learns latent representations of sub-structures in graphs (Chapter 3). Then, we propose a new smoothing framework that is applicable to structured objects, and demonstrate our framework on several benchmark graph kernels (Chapter 4).

Part II: Information Overload in Text Chapters 5-7. Here, we first introduce a novel recommender system for Reddit (Chapter 5). Then, we introduce a summarization framework for TED talks (Chapter 6). Finally, we introduce a framework

that extracts features from system logs in order to provide sizing recommendations for virtual machines (Chapter 7).

Chapter 9: Conclusions and Future Work We summarize our contributions and provide a discussion on future work in the last chapter of the dissertation.

1.5 Related Publications

The material in this dissertation is based on the papers listed below, which published or in submission to the following conference proceedings.

- Pinar Yanardag Delul, SVN Vishwanathan. *A Structural Smoothing Framework For Robust Graph Comparison*, . In Proceedings of the Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), 2015
- Pinar Yanardag Delul, SVN Vishwanathan. *Deep Graph Kernels*. In Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2015
- Pinar Yanardag Delul, SVN Vishwanathan. *Submodular Graph Kernels*, Networks Workshop at The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), 2015
- Pinar Yanardag Delul, Rean Griffith, Anne Holler, K. Shankari, Xiaoyun Zhu, Ravi Soundararajan, Adarsh Jagadeeshwaran, Pradeep Padala, *Crowdsourced Resource Sizing of Virtual Appliances*, 7th IEEE International Conference on Cloud Computing (IEEE CLOUD)
- Pinar Yanardag Delul, SVN Vishwanathan, *Understanding and Analyzing Microblogs*, WWW Doctoral Consortium, Rio de Janeiro, 2013
- Pinar Yanardag Delul, SVN Vishwanathan, *Where does the narwhal bacon? Diversity and Discoverability in Online Communities*, (Submitted to KDD 2016)

- Pinar Yanardag Delul, SVN Vishwanathan, *Ideas Worth Summarizing: A Submodular Framework to Summarize TED Talks*, (Submitted to ACL 2016)

2 BACKGROUND

In this chapter, we review the notation and related background. First, we review graph kernel literature that we use throughout the first part of the dissertation. Then, we introduce background on submodularity that we use throughout the second part of this dissertation. Finally, we introduce background on word embedding methods.

2.1 Graph Kernels

In graph kernels, we are interested in designing a kernel function that respects the structural information embedded in the graph, while being efficient to compute. Next, we introduce brief notation we are going to employ throughout the dissertation, and then introduce popular graph kernel families.

2.1.1 Notation

A *graph* is a pair $\mathcal{G} = (V, E)$ where $V = \{v_1, v_2, \dots, v_{|V|}\}$ is an ordered set of *vertices* or *nodes* and $E \subseteq V \times V$ is a set of *edges*.

Given $\mathcal{G} = (V, E)$ and $H = (V_H, E_H)$, H is a *sub-graph* of \mathcal{G} iff there is an injective mapping $\alpha : V_H \rightarrow V$ such that $(v, w) \in E_H$ iff $(\alpha(v), \alpha(w)) \in E$.

A graph \mathcal{G} is called a *labeled graph* in which there is a function $l : V \rightarrow \Sigma$ that assigns labels from an alphabet Σ to vertices in the graph.

A graph \mathcal{G} is called an *unlabeled graph* in which individual vertices have no distinct identifications other than their inter-connectivity.

Given two graphs \mathcal{G} and \mathcal{G}' , we are interested in defining a kernel $\mathcal{K}(\mathcal{G}, \mathcal{G}')$ that measures the similarity between \mathcal{G} and \mathcal{G}' . Graph classification task considers the problem of classifying graphs into two or more categories. Given a set of graphs \mathbb{G} and a set of class

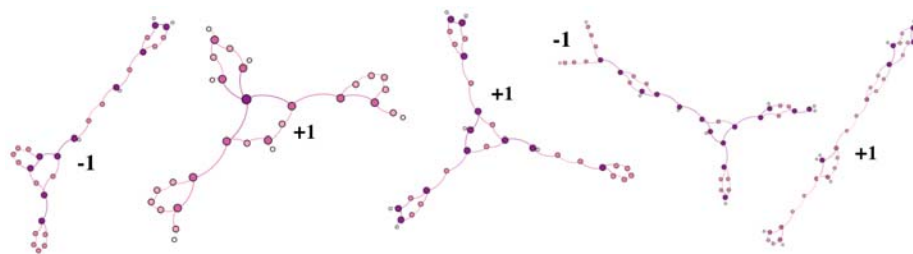


Figure 2.1. Sample graphs from NCI1 dataset [24] where each graph represents a chemical compound and labeled as +1 (active in an anti-cancer screen) or -1 (non-active in an anti-cancer screen). Graphs are created using Gephi [25].

labels \mathcal{Y} , the task in graph classification is then to learn a model that maps graphs in \mathbb{G} to the label set \mathcal{Y} (see Figure 2.1). A popular approach is to first use a graph kernel to compute a kernel matrix K of size $n \times n$ where K_{ij} represents the similarity between \mathcal{G}_i and \mathcal{G}_j , and then to plug the computed kernel matrix into a kernelized learning algorithm such as SVM [26] to perform classification. Thus, graph kernels serve as a bridge between graph structured data and kernelized learning algorithms.

R-convolution [14] is a general framework for handling discrete objects where the key idea is to recursively decompose structured objects into “atomic” sub-structures and define valid local kernels between them. In the case of graphs, given a graph \mathcal{G} , let $\phi(\mathcal{G})$ denote a vector which contains counts of atomic sub-structures, and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denote a dot product in a reproducing kernel Hilbert space \mathcal{H} , then the kernel between two graphs \mathcal{G} and \mathcal{G}' is given by

$$\mathcal{K}(\mathcal{G}, \mathcal{G}') = \langle \phi(\mathcal{G}), \phi(\mathcal{G}') \rangle_{\mathcal{H}}. \quad (2.1)$$

Existing graphs kernels can be categorized into three major families: graph kernels based on limited-sized subgraphs [16,27,], graph kernels based on subtree patterns [17,28,], and graph kernels based on walks [11, 29,] and paths [30,]. Next, we discuss each of the above kernels, and recap how they can be viewed as instances of the more general R-Convolution framework [14].

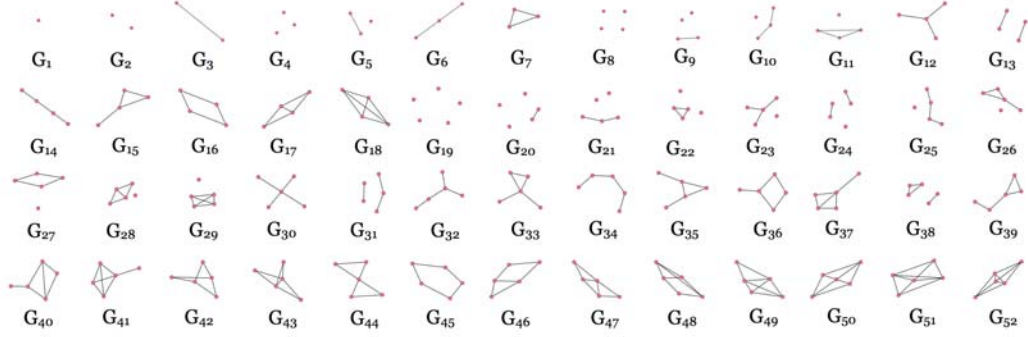


Figure 2.2. An illustration of induced, non-isomorphic induced sub-graphs of size $k \leq 5$.

2.1.2 Graph Kernels Based on Subgraphs

A *graphlet* G is an induced and non-isomorphic sub-graph of size- k (see Figure 2.2) [15]. Let $\mathcal{V}_k = \{G_1, G_2, \dots, G_{n_k}\}$ be the set of size- k graphlets where n_k denotes the number of unique graphlets of size k . Given two *unlabeled* graphs \mathcal{G} and \mathcal{G}' , the graphlet kernel is defined as follows [16]:

$$\mathcal{K}_{GK}(\mathcal{G}, \mathcal{G}') = \langle \mathbf{f}^{\mathcal{G}}, \mathbf{f}^{\mathcal{G}'} \rangle, \quad (2.2)$$

where $\mathbf{f}^{\mathcal{G}}$ and $\mathbf{f}^{\mathcal{G}'}$ are vectors of normalized counts, that is, the i -th component of $\mathbf{f}^{\mathcal{G}}$ (resp. $\mathbf{f}^{\mathcal{G}'}$) denotes the frequency with which graphlet G_i occurs as a sub-graph of \mathcal{G} (resp. \mathcal{G}'). Furthermore, $\langle \cdot, \cdot \rangle$ denotes the Euclidean dot product.

2.1.3 Graph Kernels Based on Subtree Patterns

The second family of graph kernels decomposes a graph into its subtree patterns. The Weisfeiler-Lehman subtree kernel [17] belongs to this family. The key idea here is to iterate over each vertex of a *labeled* graph and its neighbors in order to create a *multiset label*. The multiset at every iteration consists of the label of the vertex, and the sorted labels of its neighbors. The resultant multiset is given a new label, which is then used for the next iteration. When comparing graphs, we simply count the co-occurrences of labels in both graphs. This procedure is inspired by the Weisfeiler-Lehman test of graph

Table 2.1.

Common characteristics of different graph kernel families are given with representative instances where WL denotes Weisfeiler-Lehman kernel and SP denotes Shortest Path kernel. All three graph kernel families are based on MLE estimation and use frequency-based vector representation.

| Graph Family | Representative Kernel | Structure Type |
|--------------|--------------------------|------------------|
| Subgraph | Graphlet Kernel | Graphlets |
| Subtree | Weisfeiler-Lehman Kernel | Labeled Subtrees |
| Path & Walk | Shortest-Path Kernel | Shortest-paths |

isomorphism, and is equivalent to comparing the number of shared subtrees between two graphs. Formally, given \mathcal{G} and \mathcal{G}' , the Weisfeiler-Lehman subtree kernel is defined as:

$$\mathcal{K}_{WL}(\mathcal{G}, \mathcal{G}') = \langle \mathbf{I}^{\mathcal{G}}, \mathbf{I}^{\mathcal{G}'} \rangle. \quad (2.3)$$

As before, $\langle \cdot, \cdot \rangle$ denotes the Euclidean dot product. If we assume that we perform h iterations of relabeling, then $\mathbf{I}^{\mathcal{G}}$ consists of h blocks. The i -th component in the j -th block of $\mathbf{I}^{\mathcal{G}}$ contains the frequency with which the i -th label was assigned to a node in the j -th iteration.

2.1.4 Graph Kernels Based on Random-walks

The third family of graph kernels decomposes a graph into random-walks [11, 29] or paths [30] and counts the co-occurrence of random-walks or paths in two graphs. Let $\mathbb{P}_{\mathcal{G}}$ represent the set of all shortest-paths in graph \mathcal{G} , and $p_i \in \mathbb{P}_{\mathcal{G}}$ denote a triplet (l_s^i, l_e^i, n_k) where n_k is the length of the path and l_s^i and l_e^i are the labels of the starting and ending vertices, respectively. The shortest-path kernel between *labeled* graphs \mathcal{G} and \mathcal{G}' is defined as [30]:

$$\mathcal{K}_{SP}(\mathcal{G}, \mathcal{G}') = \langle \mathbf{p}^{\mathcal{G}}, \mathbf{p}^{\mathcal{G}'} \rangle, \quad (2.4)$$

where i -th component of $\mathbf{p}^{\mathcal{G}}$ contains the normalized frequency with which the i -th triplet occurs in graph \mathcal{G} . The vector $\mathbf{p}^{\mathcal{G}'}$ is defined analogously for \mathcal{G}' .

2.1.5 R-convolution Framework

One can show that all graph kernels summarized above are all instances of the R-Convolution framework (also see Table 2.1). In a nutshell, the recipe for defining graph kernels using R-convolution can be summarized as follows: A graph is recursively decomposed into its subgraphs. Next, the decomposed subgraphs are represented as a vector of frequencies where each item of the vector represents how many times a given subgraphs occurs in the graph. Finally, using the Euclidean space or some other domain specific RKHS, the dot product between the vector of frequencies is defined. Many existing graph kernels can be recovered using this general recipe. For instance, the graphlet kernel of [16] decomposes a graph into graphlets (size- k , connected, non-isomorphic sub-graphs), Weisfeiler-Lehman subtree kernel of [31] decomposes a graph into subtrees, and the shortest-path kernel of [30] decomposes a graph into shortest-paths.

2.2 Submodularity

Submodularity is a discrete optimization method that shares similar characteristics with concavity, while resembling convexity. Submodularity appears in a wide range of application areas including social networks, viral marketing [6] and document summarization [32].

Submodular functions exhibit a natural *diminishing returns* property, *i.e.*, given two sets S and T , where $S \subseteq T \subseteq V \setminus v$, the incremental *value* of an item v decreases as the context in which v is considered grows from S to T .

More formally, submodularity is a property of set functions, *i.e.*, the class of functions $f : 2^V \rightarrow R$ that maps subsets $S \subseteq V$ to a value $f(S)$ where V is a finite ground set. The function f maps any given subset to a real number. The function f is called normalized if $f(\emptyset) = 0$, and it is monotone if $f(S) \leq f(T)$, whenever $S \subseteq T$. The function f is called submodular if the following equation holds for any $S, T \subseteq V$:

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \quad (2.5)$$

Algorithm 1 Greedy submodular function maximization with budget constraint

Require: V, k
Ensure: Selected set of posts S

- 1: Initialize $S \leftarrow \emptyset$
 - 2: **while** $|S| \leq k$ **do**
 - 3: $v \leftarrow \operatorname{argmax}_{z \in V \setminus S} (f(S \cup \{z\}) - f(S))$
 - 4: $S \leftarrow S \cup \{v\}$
 - 5: **end while**
 - 6: **return** S
-

It has been shown that submodular function minimization can be solved in polynomial time [33], while submodular function maximization is an NP-complete optimization problem and intractable. However, it has been shown by [34] that the maximization of a monotone submodular function under a cardinality constraint can be solved near-optimally using a greedy algorithm. In submodular function maximization, we are interested in solving the following optimization problem:

$$A^* = \operatorname{argmax}_{A \subseteq V: |A| \leq k} f(A)$$

subject to a cardinality constraint k . If a function f is submodular, takes only non-negative values, and is monotone, then even though the maximization is still NP complete, we can use a greedy algorithm (see Algorithm 1) to approximate the optimum solution within a factor of $(1 - 1/e) \approx 0.63$ [34].

2.3 Word Embedding Models

Traditional language models estimate the likelihood of a sequence of words appearing in a corpus. Given a sequence of training words $\{w_1, w_2, \dots, w_T\}$, n -gram based language models aims to maximize the following probability

$$\Pr(w_t | w_1, \dots, w_{t-1}). \tag{2.6}$$

In other words, they estimate the likelihood of observing w_t given n previous words observed so far.

Recent work in language modeling focused on distributed vector representation of words, also referred as *word embeddings*. These neural language models improve classic n -gram language models by using continuous vector representations for words. Unlike traditional n -gram models, neural language models take advantage of the notion of *context* where a context is defined as a fixed number of preceding words. In practice, the objective of word embedding models is to maximize the following log-likelihood,

$$\sum_{t=1}^T \log \Pr(w_t | w_{t-n+1}, \dots, w_{t-1}), \quad (2.7)$$

where $w_{t-n+1}, \dots, w_{t-1}$ are the context of w_t . Continuous bag-of-words (CBOW) and Skip-gram models [35] are two popular methods that approximate Equation 2.7.

2.3.1 Continuous Bag-of-words

CBOW model predicts the current word given the surrounding words within a given window. The model architecture is similar to feedforward neural network language model [36] where the non-linear hidden layer is removed and the projection layer is shared for all words (see Figure 2.3). Formally, CBOW model aims to maximize the following log-likelihood,

$$\sum_{t=1}^T \log \Pr(w_t | w_{t-c}, \dots, w_{t+c}), \quad (2.8)$$

where c is the length of the context. The probability $\Pr(w_t | w_{t-c}, \dots, w_{t+c})$ is computed using the softmax, defined as

$$\frac{\exp(\bar{\mathbf{v}}^\top \mathbf{v}'_{w_t})}{\sum_{w=1}^{\mathcal{V}} \exp(\bar{\mathbf{v}}^\top \mathbf{v}'_w)}. \quad (2.9)$$

Here, \mathbf{v}_w corresponds to the input vector representation of w and \mathbf{v}'_{w_t} corresponds to the output vector representation of w_t . The averaged vector representation from the context is computed as

$$\bar{\mathbf{v}} = \frac{1}{2c} \sum_{-c \leq j \leq c, j \neq 0} \mathbf{v}_{w_{t+j}}. \quad (2.10)$$

2.3.2 Skip-gram

The Skip-gram model maximizes co-occurrence probability among the words that appear within a given window. In other words, instead of predicting the current word based on surrounding words, the main objective of the Skip-gram is to predict the surrounding words given the current word (see Figure 2.3). More precisely, the objective of the Skip-gram model is to maximize the following log-likelihood,

$$\sum_{t=1}^T \log \Pr(w_{t-c}, \dots, w_{t+c} | w_t). \quad (2.11)$$

where the probability $\Pr(w_{t-c}, \dots, w_{t+c} | w_t)$ is computed as

$$\prod_{-c \leq j \leq c, j \neq 0} \Pr(w_{t+j} | w_t). \quad (2.12)$$

Here, the contextual words and the current word are assumed to be independent. Furthermore, $\Pr(w_{t+j} | w_t)$ is defined as

$$\frac{\exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_{w_{t+j}})}{\sum_{w=1}^{\mathcal{V}} \exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_w)} \quad (2.13)$$

where \mathbf{v}_w and \mathbf{v}'_w are the input and output vectors of word w .

Hierarchical softmax and Negative Sampling are two efficient algorithms that are used in training the Skip-gram and CBOW models. Hierarchical softmax uses a binary Huffman tree to factorize expensive partition function of the Skip-gram model. An alternative to the Hierarchical softmax is negative sampling, which selects the contexts at random instead of considering all words in the vocabulary. In other words, if a word w appears in the context of another word w' , then the vector representation of the word w is closer to the vector

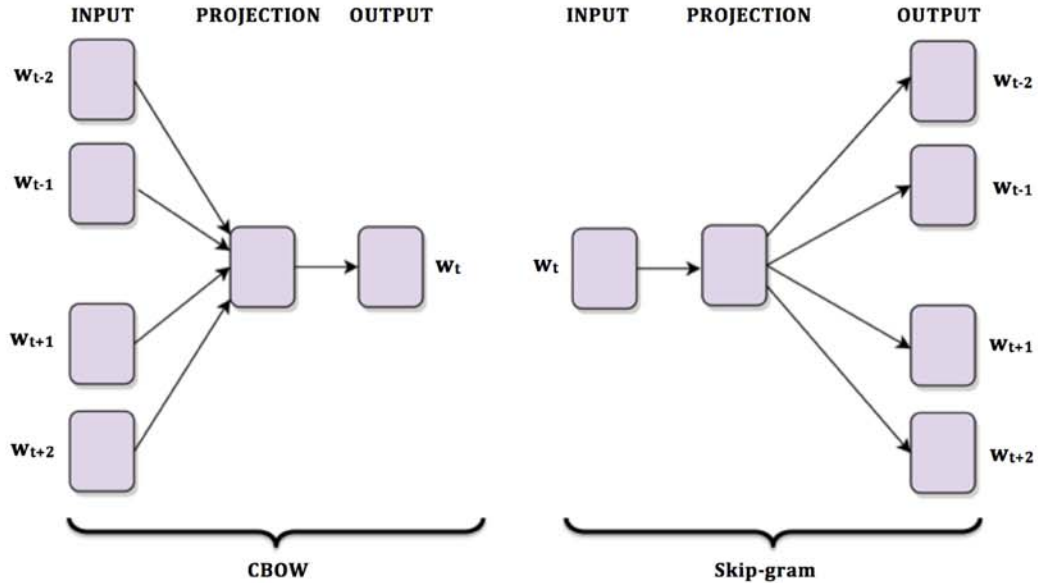


Figure 2.3. Architecture for the CBOW and Skip-gram method [35]. w_t is the current word, while w_{t+j} are the surrounding words where c is the size of the context, and $-c \leq j \leq c$.

representation of word w' comparing to any other randomly chosen words. In practice, one should try both the Skip-gram and CBOW models with Hierarchical softmax and negative sampling algorithms in order to decide which pair is more suitable to the application in hand. After the training converges, similar words are mapped to similar positions in the vector space. Word vectors are empirically shown to preserve semantics, and can be used to answer analogy questions using simple vector algebra. For instance, the result of a vector calculation $\mathbf{v}(\text{"Madrid"}) - \mathbf{v}(\text{"Spain"}) + \mathbf{v}(\text{"France"})$ is closer to $\mathbf{v}(\text{"Paris"})$ than any other word vector [35].

Part I

Information Overload in Graphs

3 DEEP GRAPH KERNELS

In this chapter, we present a unified framework to learn latent representations of sub-structures for graphs, inspired by latest advancements in language modeling and deep learning. Our framework leverages the co-occurrence information between sub-structures by *learning* their latent representations, and alleviate the information overload problem arise in feature space by incorporating the relationship between sub-structures into the kernel computation. We demonstrate instances of our framework on three popular graph kernels, namely Graphlet kernels, Weisfeiler-Lehman subtree kernels, and Shortest-Path graph kernels. Our experiments on several benchmark datasets show that Deep Graph Kernels achieve significant improvements in classification accuracy over state-of-the-art graph kernels.

3.1 Motivation

Given a graph \mathcal{G} , let $\phi(\mathcal{G})$ denote a vector which contains counts of atomic sub-structures, and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denote a dot product in a RKHS \mathcal{H} . Then, the kernel between two graphs \mathcal{G} and \mathcal{G}' is given by

$$\mathcal{K}(\mathcal{G}, \mathcal{G}') = \langle \phi(\mathcal{G}), \phi(\mathcal{G}') \rangle_{\mathcal{H}}. \quad (3.1)$$

However, this representation does not take a number of important observations into account and suffer from information overload problem in the feature space (see the discussion in Chapter 1). To alleviate this problem, consider an alternative kernel between two graphs \mathcal{G} and \mathcal{G}' such that,

$$\mathcal{K}(\mathcal{G}, \mathcal{G}') = \phi(\mathcal{G})^T \mathcal{M} \phi(\mathcal{G}') \quad (3.2)$$

where \mathcal{M} represents a $|\mathcal{V}| \times |\mathcal{V}|$ positive semi-definite matrix that encodes the relationship between sub-structures and \mathcal{V} represents the vocabulary of sub-structures obtained from

the training data. Therefore, one can design a matrix \mathcal{M} that respects the similarity of the sub-structure space. In cases where there is a strong mathematical relationship between sub-structures, such as *edit-distance*, one can design matrix \mathcal{M} that respects the geometry of the space. In cases where a clear mathematical relationship between sub-structures might not exist, one can *learn* the geometry of the space directly from data. In this chapter, we propose recipes for designing such \mathcal{M} matrices for graph kernels. For our first recipe, we exploit an edit-distance relationship between sub-structures and directly compute a matrix \mathcal{M} . In our second recipe, we propose a framework that computes an \mathcal{M} matrix by *learning* latent representations of sub-structures.

The rest of this chapter is as follows. In Section 3.2, we first design a matrix \mathcal{M} for Graphlet kernels by exploiting edit-distance relationship between sub-structures, and then we introduce a framework that learns the relationship between sub-structures. In Section 3.3, we discuss related work. In Section 3.4, we compare the classification performance of deep graph kernels to their base variants as well as to other state-of-the-art graph kernels. We report results on classification accuracy on graph benchmark datasets and discuss the run-time cost of our framework. Section 3.5 concludes the chapter.

3.2 Methodology

In this section, we first discuss how to compute a matrix \mathcal{M} by using the *edit-distance* relationship between sub-structures (Section 3.2.1). Then, we discuss how to compute a matrix \mathcal{M} by *learning* the similarity between sub-structures inspired by latest advancements in language modeling and deep learning (Section 3.2.2).

3.2.1 Sub-structure Similarity via Edit-distance

When sub-structures exhibit a clear mathematical relationship, one can exploit the underlying similarities between sub-structures to compute a matrix \mathcal{M} . For instance, in Graphlet kernels, one can derive an *edit-distance* relationship to encode how similar one graphlet is to another. Given a graphlet G_i of size k , and a graphlet G_j of size $k + 1$,

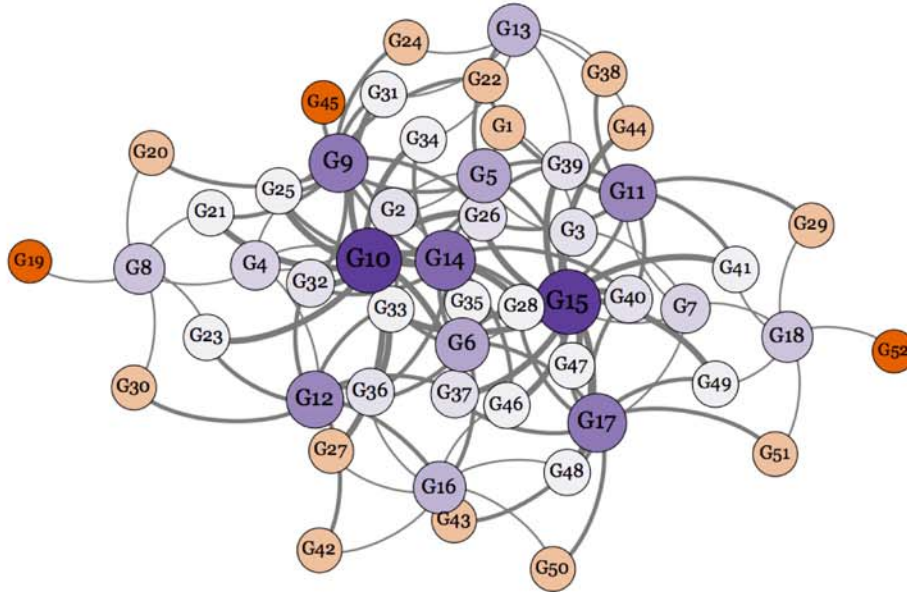


Figure 3.1. Undirected edit-distance graph \mathfrak{G} for graphlets of size $k \leq 5$ (size and colors of the nodes are based on degree).

let us build an undirected edit-distance graph \mathfrak{G} by adding an undirected edge from G_i to G_j if and only if G_i can be obtained from G_j by deleting a node of G_j (or vice versa, if G_j can be obtained from G_i by adding a node to G_i). Given such an undirected graph \mathfrak{G} , one can simply compute the shortest-path distance between G_i and G_j in order to compute their *edit-distance* (see Figure 3.1). While this approach enables us to directly compute a matrix \mathcal{M} , the cost of computing the shortest-path distances on \mathfrak{G} becomes prohibitively expensive as a function of graphlet size k . For instance, in order to compute a matrix \mathcal{M} at level $k = 9$, one needs to compute all pairwise shortest-path distances of an undirected graph having 288,267 nodes. On the other hand, one can observe that while the number of unique graphlets grow exponentially, only a few of them will be observed in a given graph. Therefore, instead of computing a complete \mathcal{M} matrix of size $|\mathcal{V}| \times |\mathcal{V}|$, one can design an \mathcal{M} matrix of size $|\mathcal{V}'| \times |\mathcal{V}'|$ with $|\mathcal{V}'| \ll |\mathcal{V}|$ by only taking the *observed* sub-structures into account. In next section, we discuss an approach that utilizes this observation.

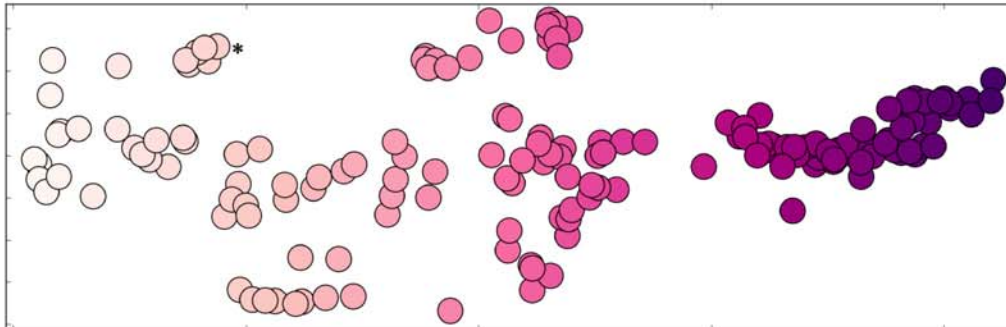


Figure 3.2. The learned shortest-path sub-structures on ENZYMES dataset [8] in \mathbb{R}^2 . Each node corresponds to a different shortest-path sub-structure and colored by the value of their first dimension (labels on nodes are omitted for clarity). For instance, the shortest path sub-structures embedded in the cluster marked with * are $2 \cdot 2_{23}$, $2 \cdot 2_{24}$, $2 \cdot 2_{25}$, $2 \cdot 2_{26}$, $2 \cdot 2_{27}$, $2 \cdot 2_{28}$ where first two characters represent the start and end point of the shortest-path and sub-script represents the length of the path). Note that the shortest-paths having the same start and endpoints with similar shortest-path lengths are close to each other in the latent space.

3.2.2 Sub-structure Similarity via Learning

Our second approach is to *learn* the latent representations of sub-structures by using recently introduced language modeling and deep learning techniques. The learned representations are then utilized to compute an \mathcal{M} matrix that respects the similarity between sub-structures.

Neural language models recently gained an immense popularity due to their ability to excel various natural language processing tasks. Continuous bag-of-words (CBOW) and Skip-gram models [35] are two such methods that take advantage of distributed vector representations of words. Unlike traditional language models, these methods utilize the notion of a *context* where a context is defined as a fixed number of preceding words (see Chapter 2.3). The ability to represent words as vectors, and the flexible notion of ‘nearness’ by using the contexts make neural language models attractive for our task.

A key intuition we utilize in our framework is to view *sub-structures* in graph kernels as *words* that are generated from a special language. In other words, different sub-structures

compose graphs in a similar way that different words form sentences when used together. With this analogy in mind, one can utilize word embedding models to unveil dimensions of similarity between sub-structures. The main expectation here is that similar sub-structures will be close to each other in the d -dimensional latent space. Figure 3.2 illustrates shortest-path sub-structures in \mathbb{R}^2 learned by our framework. Note that similar sub-structures are close together in latent space.

3.2.3 Deep Graph Kernels

Our framework first takes a list of graphs \mathbb{G} and decomposes each graph into its sub-structures. The list of decomposed sub-structures for each graph is then treated as a sentence that is generated from a vocabulary \mathcal{V} where vocabulary \mathcal{V} simply corresponds to the unique set of *observed* sub-structures in the training data. However, unlike words in a traditional text corpora, sub-structures do not have a linear co-occurrence relationship. Therefore, one needs to build a corpus where the co-occurrence of the sub-structures is meaningful. Next, we discuss how to generate corpora where co-occurrence relationship is meaningful on three major graph kernel families.

Corpus Generation for Graphlet Kernels: Exhaustive enumeration of all graphlets in a graph \mathcal{G} is prohibitively expensive for even moderate sized graphs [15]. Several sampling heuristics are proposed for sampling sub-graphs efficiently, such as random sampling scheme of [16]. In practice, the random sampling of graphlets of size k in a graph \mathcal{G} involves placing a randomly generated window of size $k \times k$ on the adjacency matrix of \mathcal{G} and collecting the observed graphlet in that window. This procedure is repeated n times where n being the number of graphlets we would like to sample. However, since this is a random sampling scheme, it does not preserve any notion of co-occurrence relationship which is a desired property for our framework. Therefore, we modify the random sampling scheme to partially preserve the co-occurrence between graphlets by using the notion of *neighborhoods*. That is, whenever we randomly sample a graphlet G , we also sample its immediate neighbors. The graphlet and its neighbors are then interpreted as *co-occurred*

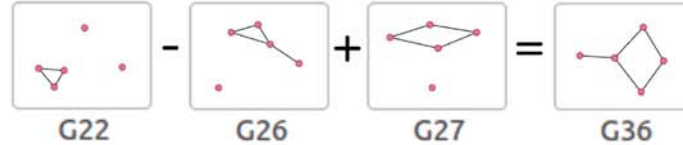


Figure 3.3. An example of sub-structure regularity in graphlet sub-structure space.

by our method. Therefore, graphlets which have similar neighborhoods will acquire similar representations. While in this chapter we utilized only the immediate neighbors of a graphlet, one can extend the co-occurrence relationship to consider neighborhoods of distance ≥ 1 .

In order to verify this intuition, we explored the idea of having a similar effect of linguistic regularities in language models to graphs. Word embeddings are known to successfully answer queries such as “*What is the word that is similar to small in the same sense as biggest is similar to big?*” where answer is correctly recovered as *smallest* [37]. Therefore, we investigate whether a meaningful response with a similar analogy can be recovered, such that “*What is the graphlet that is similar to a square (G_{27}) in the same sense as triangle with a tail (G_{26}) is similar to triangle (G_{22})?*”. We used the multiplicative combination objective proposed by [38]:

$$\operatorname{argmax}_{b^* \in \mathcal{V}} \frac{\cos(b^*, b) \cos(b^*, a^*)}{\cos(b^*, a) + \epsilon} \quad (3.3)$$

where a and b are sub-structures from a vocabulary \mathcal{V} , \cos is the cosine similarity between sub-structure vectors and $\epsilon = 0.001$ is used to prevent division by zero. This objective function amplifies the differences between small quantities and reduces the differences between larger ones. We used two positive (G_{22} , G_{27}) and one negative example (G_{26}), and recovered G_{36} as the top sub-structure by this arithmetic operation on embedded vectors (see Figure 3.3).

Corpus Generation for Shortest-Path Graph Kernels: Shortest-path graph kernel compares the sorted endpoints and the length of shortest-paths that are common between two

graphs. Similar to graphlet kernel, one needs to find a meaningful co-occurrence relationship between shortest-path sub-structures. One can show that all sub-paths of a shortest-path are also shortest-paths with the same source [39]. In other words, whenever we observe a shortest-path sub-structure p of length l , we must also observe all of its sub-paths of length $< l$ as well. Inspired by this property, whenever we generate a shortest-path sub-structure, we also collect all possible shortest-path sub-structures that share the same source node, and treat them as *co-occurred*. Therefore, shortest-path sub-structures which have similar labels will acquire similar representations (see Figure 3.2).

Corpus Generation for Weisfeiler-Lehman Kernels: The Weisfeiler-Lehman subtree kernel iterates over each vertex and its neighbors in order to create a multiset label. The resultant multiset is given a new label, which is then used for the next iteration. Therefore, multiset labels that belong a given iteration h can be treated as *co-occurred* in order to partially preserve a notion of similarity.

After generating a corpus where a co-occurrence relationship is partially preserved, we build a model by using CBOW and Skip-gram algorithms¹. Let s represent an arbitrary sub-structure from a vocabulary \mathcal{V} , and Φ_s represent learned vector representation of s using our framework. Given the vector representations of sub-structures, we compute a diagonal \mathcal{M} matrix such that each entry on the diagonal, \mathcal{M}_{ii} computed as $\langle \Phi_i, \Phi_i \rangle$ where Φ_i corresponds to learned d -dimensional hidden of sub-sequence i and $\mathcal{M}_{ij} = 0$ where $i \neq j$ and $1 \leq i \leq |\mathcal{V}|$ (resp. j). After computing the \mathcal{M} matrix, we simply plug it into Equation 3.2 in order to compute the kernel between each sub-structure.

3.2.4 Deep String Kernels

In a similar fashion, we can plug other graph kernels into our framework such as Random-walk kernels [40], labeled version of graphlet kernel [16], subtree kernels [28,41], cyclic pattern kernels [27] and p -step Random-walk kernel [42]. Moreover, our framework

¹We used Gensim library [85] for all algorithms.

is applicable to any R-convolution kernel where there is a notion of dependency between sub-structures, such as *string kernels*.

String kernels are another popular instance of R-convolution kernels where the task is to compute a kernel between two *sequences*, such as DNA strings. Given an input sequence \mathcal{S} over an alphabet \mathcal{V} and a number $k \geq 1$, k -spectrum of the sequence \mathcal{S} is defined as the set of all k -length contiguous sub-sequences \mathcal{S} contains [43]. The feature vector $\phi(\mathcal{S})$ is then simply constructed as a frequency vector over sub-sequences in its k -spectrum and the kernel between two sequences are computed via Equation 3.1. Similar to graph kernels, the co-occurrence relationship between sub-sequences are not taken into account in k -spectrum kernel. Similar to graph kernels, we treat all length k sub-sequences of a string as co-occurred and learn the hidden representation of each spectrum using our framework. In case of string kernels, we compute \mathcal{M} matrix such that each entry \mathcal{M}_{ij} computed as $\langle \Phi_i, \Phi_j \rangle$ where Φ_i corresponds to learned d -dimensional vector of sub-sequence i (resp. Φ_j).

3.3 Related Work

The closest work to our framework is the recently proposed model, DeepWalk by [44]. DeepWalk learns social representations of vertices of graphs by modeling short Random-walks. We distance ourselves from DeepWalk in several aspects. First, instead of learning similarities between *nodes* we are interested in learning similarities between *structured objects*, such as graphs and strings. In other words, DeepWalk operates on a *single* graph, while we are interested in the relationship between *multiple* graphs. Moreover, instead of using Random-walks, our framework can be configured to work with any type of sub-structures, including graphlets, shortest-paths, sub-trees and strings.

Many different graph kernels focusing on different types of subgraphs have been defined in the past which can be categorized into three major families: graph kernels based on limited-sized subgraphs [27], [16], graph kernels based on subtree patterns [28], [17] and graph kernels based on walks [29] and paths [30]. Our framework is *complementary*

to existing graph and string kernels where the sub-structures have a similarity relationship between them.

3.4 Experiments

The aim of our experiments is threefold. First, we want to show that using an \mathcal{M} matrix that infers the relationship between sub-structures improves the classification accuracy. Second, we want to show that our framework is *robust* to random noise. Third, we want to show that the deep kernels are comparable to or outperform state-of-the-art graph kernels in terms of classification accuracy, while remaining competitive in terms of computational requirements.

3.4.1 Experimental Setup

We compare our framework against representative instances of major families of graph kernels in the literature. Other than base kernels of our framework, namely, Weisfeiler-Lehman subtree kernel [16], Graphlet kernel [16], and Shortest-path kernel [30], we also compare our kernels with the random walk kernel [40], the subtree kernel [28], and p -step Random-walk kernel [42]. The Random-walk, p -step Random-walk and Ramon-Gärtner kernels are written in Matlab and were obtained from the authors of [16]. All other kernels were coded in Python. In order to ensure a fair comparison, all experiments are performed on the same hardware.

All kernels are normalized to have a unit length in the feature space. Moreover, we use 10-fold cross validation with a binary C -SVM [45] to test classification performance. The C value for each fold is independently tuned using training data from that fold. In order to exclude random effects of the fold assignments, this experiment is repeated 10 times, and average prediction accuracies with their standard deviations are reported.

Table 3.1.
Properties of the bioinformatics datasets used in graph kernel experiments.

| Dataset | Size | Classes | Avg.nodes | Labels |
|----------|------|---------|-----------|--------|
| MUTAG | 188 | 2 | 17.9 | 7 |
| PTC | 344 | 2 | 25.5 | 19 |
| ENZYMES | 600 | 6 | 32.6 | 3 |
| PROTEINS | 1113 | 2 | 39.1 | 3 |
| NCI1 | 4110 | 2 | 29.8 | 37 |
| NCI109 | 4127 | 2 | 29.6 | 38 |

3.4.2 Datasets

In this section, we introduce the datasets that are used in our experiments. In particular, we first use several benchmark graph kernel datasets from bioinformatics. Then, we derive several new and large datasets related to social network domain. Finally, we introduce benchmark datasets used in string kernel experiments.

Bioinformatics Datasets We applied our framework to benchmark graph kernel datasets, namely, MUTAG, PTC, ENZYMES, PROTEINS and NCI1, NCI109. MUTAG is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds [46] with 7 discrete labels. PTC [47] is a dataset of 344 chemical compounds that reports the carcinogenicity for male and female rats and it has 19 discrete labels. NCI1 and NCI109 [24] datasets (4100 and 4127 nodes, respectively), made publicly available by the National Cancer Institute (NCI) are two subsets of balanced datasets of chemical compounds screened for ability to suppress or inhibit the growth of a panel of human tumor cell lines, having 37 and 38 discrete labels respectively. ENZYMES is a balanced dataset of 600 protein tertiary structures obtained from [8] and has 3 discrete labels. PROTEINS is a dataset obtained from [8] where nodes are secondary structure elements (SSEs) and there is an edge between two nodes if they are

Table 3.2.
Properties of the social network datasets used in graph kernel experiments.

| Dataset | Size | Classes | Avg.nodes |
|------------------|-------|---------|-----------|
| COLLAB | 5000 | 3 | 74.49 |
| IMDB-BINARY | 1000 | 2 | 19.77 |
| IMDB-MULTI | 1500 | 3 | 13 |
| REDDIT-BINARY | 2000 | 2 | 429.61 |
| REDDIT-MULTI-5K | 5000 | 2 | 508.5 |
| REDDIT-MULTI-12K | 11929 | 11 | 391.4 |

neighbors in the amino-acid sequence or in 3D space. It has 3 discrete labels, representing *helix*, *sheet* or *turn*. See Table 3.1 for detailed statistics of the datasets.

Social Network Datasets In order to test the efficacy of our framework on social network domain, we derive several unlabeled graph datasets with different tasks as follows. Table 3.2 lists detailed statistics of the datasets.

- **Reddit datasets:** *REDDIT-BINARY* is a balanced dataset where each graph corresponds to an online discussion thread where nodes correspond to users, and there is an edge between two nodes if at least one of them responded to another’s comment. We crawled top submissions from four popular subreddits, namely, *IAmA*, *AskReddit*, *TrollXChromosomes*, *atheism*. *IAmA* and *AskReddit* are two question/answer-based subreddits and *TrollXChromosomes* and *atheism* are two discussion-based subreddits. The task is then to identify whether a given graph belongs to a *question/answer*-based community or a *discussion*-based community. *REDDIT-MULTI-5K* is a balanced dataset from five different subreddits, namely, *worldnews*, *videos*, *AdviceAnimals*, *aww* and *mildlyinteresting* where we simply label each graph with their correspondent subreddit. *REDDIT-MULTI-12K* is a larger variant of *REDDIT-MULTI-5K*, consists of 11 different subreddits, namely, *AskReddit*, *AdviceAnimals*, *atheism*,

aww, *IAmA*, *mildlyinteresting*, *Showerthoughts*, *videos*, *todayilearned*, *worldnews*, *TrollXChromosomes*. The task in both datasets is to predict which subreddit a given discussion graph belongs to.

- **Scientific Collaboration Dataset:** *COLLAB* is a scientific-collaboration dataset, derived from 3 public collaboration datasets [48], namely, *High Energy Physics*, *Condensed Matter Physics* and *Astro Physics*. Following the approach of [49], we generated ego-networks of different researchers from each field, and labeled each graph as the field of the researcher. The task is then to determine whether the ego-collaboration graph of a researcher belongs to High Energy, Condensed Matter or Astro Physics field.
- **Movie Collaboration Datasets:** *IMDB-BINARY* is a movie-collaboration dataset where we collected *actor/actress* and *genre* information of different movies on IMDB. For each graph, nodes represent actors/actresses and there is an edge between them if they appear in the same movie. We generated collaboration graphs on *Action* and *Romance* genres and derived ego-networks for each actor/actress. Note that a movie can belong to both genres at the same time, therefore we discarded movies from Romance genre if the movie is already included to the Action genre. Similar to *COLLAB* dataset, we simply labeled each ego-network with the genre graph it belongs to. The task is then simply to identify which genre an ego-network graph belongs to. *IMDB-MULTI* is multi-class version of *IMDB-BINARY* and contains a balanced set of ego-networks derived from *Comedy*, *Romance* and *Sci-Fi* genres.

String Datasets In order to test the efficacy of our model, we applied our method to benchmark datasets in string kernels. SCOP (Structural Classification of Proteins) is a manually-curated database that groups proteins together based on their 3-D structures [50] (see Figure 3.4 for a sample protein from TIM beta/alpha-barrel fold²). The task is then to classify protein sequences into 7 distinct super-families. SCOP database has a 4-level

²Image is generated by using PyMOL toolkit: <http://http://pymol.org>

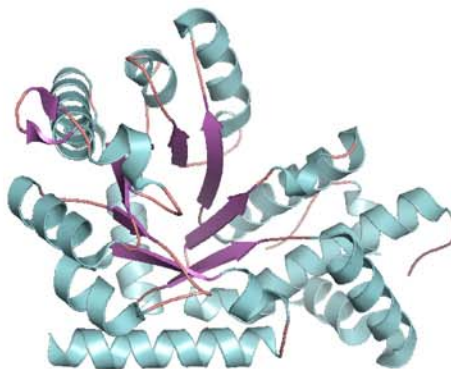


Figure 3.4. An example protein *2ffc* from Triose Phosphate Isomerase beta/alpha-barrel fold.

structure-based hierarchy of classes where protein sequences are classified into one of the classes, namely, *class*, *fold*, *super-family* and *family*. Similar to the setting in [51], we tackled family and super-family classification problems where a family contains proteins with clear evolutionary relationship, and super-family contains the same evolutionary origin without being detectable at the level of sequences [52]. In *family-classification* problem, we considered NAD(P)-binding Rossmann-fold and (trans)-glycosidases domains where our main task is to classify proteins in a super-family into their families. NAD(P)-Rossmann dataset has 246 sequences having an average length of 218 with 6 classes where (trans)-glycosidases has 95 sequences having an average length of 375 with 2 classes.

In *super-family* classification problem, we used Triose Phosphate Isomerase (TIM) beta/alpha-barrel protein fold where we classify each protein to one of the 7 distinct super-families. TIM beta/alpha dataset has 330 sequences having an average length of 332 with 7 classes. In order to derive the labels of each sequence, we used Astral SCOPe 2.04 genetic domain sequence database [53], based on PDB SEQRES records, with less than 95% identity to each other. Moreover, we derived a new dataset for string kernels using the transcripts of TED.com talks. We collected the transcript of 385 talks having an average length of 9425 from three categories, namely, **T**echnology, **E**ntertainment, **D**esign. The task is then to predict which of the three category a talk belongs to.

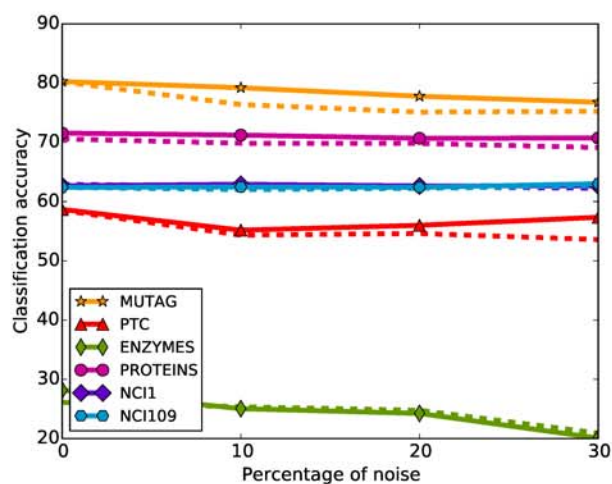


Figure 3.5. Classification accuracy (y -axis) vs. edge noise (x -axis) on different datasets using Graphlet kernel and Edit-distance Graphlet kernel. Dashed lines represent original Graphlet kernel while non-dashed lines represents its corresponding variant derived from our framework.

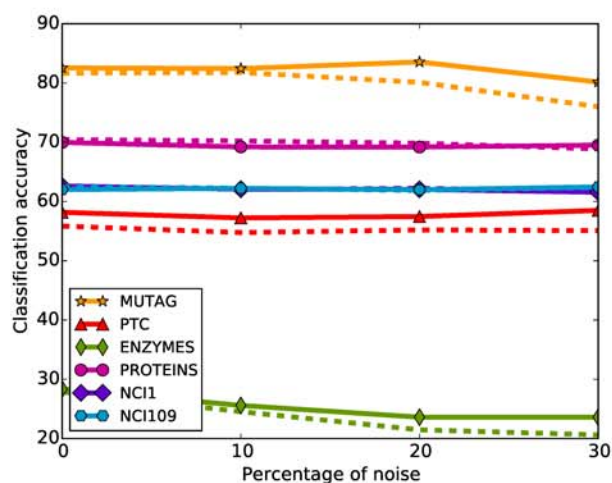


Figure 3.6. Classification accuracy (y -axis) vs. edge noise (x -axis) on different datasets using Graphlet kernel and Deep Graphlet kernel. Dashed lines represent original Graphlet kernel while non-dashed lines represents its corresponding variant derived from our framework.

3.4.3 Parameter Selection

We chose parameters for the various kernels as follows: the window size and dimension for deep graph kernels is chosen from $\{2, 5, 10, 25, 50\}$, the decay factor for Random-walk kernels is chosen from $\{10^{-6}, 10^{-5}, \dots, 10^{-1}\}$, the p value in the p -step Random-walk kernel is chosen from $\{1, 2, \dots, 10\}$ and the height parameter in Ramon-Gärtner subtree kernel is chosen from $\{1, 2, 3\}$. For each kernel, we report the results for the parameter which gave the best classification accuracy. For Weisfeiler-Lehman subtree kernel, we experimented with the height parameter $h = 2$ due to exponentially increasing feature space of the original kernel. For the Graphlet kernel, we set the size of the graphlets k to be 7 since it exhibits the sparsity problem that we are interested in. We used Nauty [54] to get canonically-labeled isomorphic representations of each graphlet which are then used to construct the feature representation.

3.4.4 Computational Cost

For Edit-distance Graphlet kernel, computing an \mathcal{M} matrix involves a one-time computation of the undirected graph between 1253 nodes for $k = 7$ which empirically takes 7 minutes. After that, one needs to compute all-pairs-shortest-path distances on the obtained undirected graph which empirically takes 8 seconds. For deep graph kernels, the overhead of computing an \mathcal{M} matrix involves learning latent representations of the observed substructures. The runtime averaged out of all datasets for learning the latent representations is 21.5 seconds for deep Graphlet kernel, 4.5 seconds for deep shortest-path graph kernel and 1.75 seconds for deep Weisfeiler-Lehman graph kernel. All runtime experiments use a fixed window size and dimension at 25 and this process is repeated 10 times to eliminate random effects.

Table 3.3.
Comparison of classification accuracy (\pm standard deviation) of the Graphlet kernel and Deep Graphlet kernel on social network datasets.

| Dataset | Graphlet kernel | Deep Graphlet kernel |
|------------------|------------------|----------------------|
| COLLAB | 72.84 ± 0.28 | 73.09 ± 0.25 |
| IMDB-BINARY | 65.87 ± 0.98 | 66.96 ± 0.56 |
| IMDB-MULTI | 43.89 ± 0.38 | 44.55 ± 0.52 |
| REDDIT-BINARY | 77.34 ± 0.18 | 78.04 ± 0.39 |
| REDDIT-MULTI-5K | 41.01 ± 0.17 | 41.27 ± 0.18 |
| REDDIT-MULTI-12K | 31.82 ± 0.08 | 32.22 ± 0.10 |

3.4.5 Results for Graph Kernels

In this section, we apply our framework to several benchmark datasets and compare the classification accuracy of our kernels against their base variants.

Graphlet kernels Under Noise We have two variants of Graphlet kernels, namely, Edit-distance Graphlet Kernel (EGK) introduced in Section 3.2.1 and Deep Graphlet kernel (DGK) introduced in Section 3.2.2. Since Graphlet kernels do not exploit label information on the vertices and only compare graphs based on their structural similarity, an interesting experiment is to see how our kernels behave under random noise on the edges. Therefore, we derive noisy variants of the datasets by randomly flipping 10%, 20% and 30% of the edges. Figure 3.5 shows the comparison between original Graphlet kernel and EGK where 0% represents the classification accuracy on the original dataset without noise. As can be seen from the figure, EGK outperforms the base kernel in MUTAG, PTC, PROTEINS, NCI1, NCI109, but outperformed by the original kernel in ENZYMES dataset. We believe this is due to the fact that EGK only uses a mathematical relationship between substructures rather than *learning* a sophisticated relationship. Therefore, we applied our deep kernel framework on Graphlet kernels (see Figure 3.6). As can be seen from the figure,

learning latent representations of the graphlets outperforms its base variant significantly in *all* datasets except PROTEINS.

Graphlet kernels on Social Network Datasets Next, we test the efficacy of our framework on several social network datasets using Graphlet kernels. As can be seen from Table 3.3, Deep Graphlet kernels are able to outperform its base variant in all cases.

Deep Graph Kernels on Bioinformatics Datasets Table 3.4 shows the classification accuracy between the Graphlet, Shortest-path and Weisfeiler-Lehman graph kernel and their deep variants. Deep variant of graph kernels are able to outperform their base variant for all cases.

Comparison Against Other Kernels Next, we compare the performance of Deep Graph Kernels with state-of-the-art graph kernels in the literature. Table 3.5 shows the comparison of Deep Graph Kernels with Ramon & Gärtner, p -Random-walk, and Random-walk graph kernels. The result for Deep Graph Kernels is constructed by picking the best result of Deep Graph Kernels from Table 3.4. As can be seen from the results, Deep Graph Kernels are able to outperform other graph kernels.

3.4.6 Results for String Kernels

As a proof-of-concept, we derive a deep variant of k -spectrum string kernel and perform experiments on benchmark bioinformatics datasets. The comparison between original k -spectrum string kernel with $k = 3$ and our method can be seen from Table 3.6. One can see that the deep variant of k -spectrum string kernel is able to outperform the base k -spectrum string kernel in all datasets.

Table 3.4.
 Comparison of classification accuracy (\pm standard deviation) of the graphlet kernel (GK), Shortest-path kernel (SP), Weisfeiler-Lehman kernel (WL) to their deep variants on bioinformatics datasets.

| Dataset | GK | Deep GK | SP | Deep SP | WL | Deep WL |
|----------|------------------|------------------|------------------|------------------|------------------|------------------|
| MUTAG | 81.66 \pm 2.11 | 82.66 \pm 1.45 | 85.22 \pm 2.43 | 87.44 \pm 2.72 | 80.72 \pm 3.00 | 82.94 \pm 2.68 |
| PTC | 57.26 \pm 1.41 | 57.32 \pm 1.13 | 58.24 \pm 2.44 | 60.08 \pm 2.55 | 56.97 \pm 2.01 | 59.17 \pm 1.56 |
| ENZYMES | 26.61 \pm 0.99 | 27.08 \pm 0.79 | 40.10 \pm 1.50 | 41.65 \pm 1.57 | 53.15 \pm 1.14 | 53.43 \pm 0.91 |
| PROTEINS | 71.67 \pm 0.55 | 71.68 \pm 0.50 | 75.07 \pm 0.54 | 75.68 \pm 0.54 | 72.92 \pm 0.56 | 73.30 \pm 0.82 |
| NCI1 | 62.28 \pm 0.29 | 62.48 \pm 0.25 | 73.00 \pm 0.24 | 73.55 \pm 0.51 | 80.13 \pm 0.50 | 80.31 \pm 0.46 |
| NCI109 | 62.60 \pm 0.19 | 62.69 \pm 0.23 | 73.00 \pm 0.21 | 73.26 \pm 0.26 | 80.22 \pm 0.34 | 80.32 \pm 0.33 |

Table 3.5.
 Comparison of classification accuracy (\pm standard deviation) of Ramon & Gärtner, p -Random-walk, and Random-walk graph kernels. $> 72\text{H}$ indicates that the computation did not finish after 72 hours.

| Dataset | Deep Graph Kernels | Ramon&Gärtner | p -Random-walk | Random-walk |
|----------|--------------------|------------------|------------------|------------------|
| MUTAG | 87.44 \pm 2.72 | 84.88 \pm 1.86 | 80.05 \pm 1.64 | 83.72 \pm 1.50 |
| PTC | 60.08 \pm 2.55 | 58.47 \pm 0.90 | 59.38 \pm 1.66 | 57.85 \pm 1.30 |
| ENZYMES | 53.43 \pm 0.91 | 16.96 \pm 1.46 | 30.01 \pm 1.01 | 24.16 \pm 1.64 |
| PROTEINS | 75.68 \pm 0.54 | 70.73 \pm 0.35 | 71.16 \pm 0.35 | 74.22 \pm 0.42 |
| NCI1 | 80.31 \pm 0.46 | 56.61 \pm 0.53 | $> 72\text{h}$ | $> 72\text{h}$ |
| NCI109 | 80.32 \pm 0.33 | 54.62 \pm 0.23 | $> 72\text{h}$ | $> 72\text{h}$ |

Table 3.6.

Classification accuracy for string kernel experiments where numbers next to the accuracy results represents the standard deviation.

| Dataset | K-Spectrum | Deep Spectrum |
|---------------------|------------------|------------------|
| TIM beta/alpha | 67.60 \pm 1.13 | 69.03 \pm 1.03 |
| (trans)glycosidases | 93.88 \pm 2.17 | 95.33 \pm 1.02 |
| NAD(P)-Rossmann | 69.87 \pm 0.78 | 75.54 \pm 0.85 |
| TED | 74.31 \pm 0.88 | 77.39 \pm 0.97 |

3.5 Conclusions

In this chapter, we presented a novel framework for graph kernels inspired by latest advancements in natural language processing and deep learning. We applied our framework to three popular graph kernels, namely, Graphlet kernel, Shortest-path kernel, and Weisfeiler-Lehman subtree kernels. We introduced several new and large graph kernel datasets in social network domain, and showed that our framework outperforms its base variants in terms of classification accuracy while introducing a negligible overhead.

Moreover, while we mainly restricted ourselves to graph kernels in this study, we discussed that our framework is rather general, and lends itself to many extensions. For instance, it can be plugged directly into any R-convolution kernel as long as there is a dependency between sub-structures. We demonstrated one such extension on string kernels and achieved significant improvements in classification accuracy.

4 SMOOTHED GRAPH KERNELS

In this chapter, we propose a general smoothing framework for graph kernels by taking *structural similarity* into account, and apply it to derive smoothed variants of popular graph kernels. Our framework is inspired by state-of-the-art smoothing techniques used in natural language processing. However, unlike NLP applications that primarily deal with strings, we show how one can apply smoothing to a richer class of inter-dependent sub-structures that naturally arise in graphs. Moreover, we discuss extensions of the Pitman-Yor process that can be adapted to smooth structured objects, thereby leading to novel graph kernels. Our kernels are able to tackle the diagonal dominance problem while respecting the structural similarity between features. Experimental evaluation shows that not only our kernels achieve statistically significant improvements over the unsmoothed variants, but also outperform several other graph kernels in the literature.

4.1 Motivation

Many graph kernels can be viewed as instances of R-convolution framework. However, R-convolution based graph kernels suffer from a few drawbacks. First, the size of the feature space often grows exponentially. As size of the space grows, the probability that two graphs will contain similar sub-structures becomes very small. Therefore, a graph becomes similar to itself but not to any other graph in the training data. This is well known as the *diagonal dominance problem* [55] where the resulting kernel matrix is close to the identity matrix. Second, lower order sub-structures tend to be more numerous while a vast majority of the sub-structures occurs rarely. In other words, a few sub-structures dominate the distribution. This exhibits a strong power-law behavior and results in underestimation of the true distribution. Third, the sub-structures used to define a graph kernel are often related to each other. However, an R-convolution kernel only respects exact matchings. This

problem is particularly important when noise is present in the training data and considering partial similarity between sub-structures might alleviate the noise problem.

In this study, we propose to tackle the above problems by using a general framework to *smooth* graph kernels that are defined using a frequency vector of decomposed structures. We use *structure* information by encoding relationships between lower and higher order sub-structures in order to derive our method.

The remainder of this chapter is structured as follows. In Section 4.2.1, we review smoothing methods for multinomial distributions. In Section 4.2.2, we introduce a framework for smoothing structured objects. In Section 4.2.3, we propose a Bayesian variant of our model that is extended from the Hierarchical Pitman-Yor process [56]. In Section 4.3, we discuss related work. In Section 4.4, we compare smoothed graph kernels to their unsmoothed variants as well as to other state-of-the-art graph kernels. We report results on classification accuracy on several benchmark datasets as well as their noisy-variants. Section 4.6 concludes the chapter.

4.2 Methodology

We first briefly review smoothing techniques for multinomial distributions, and then we propose a new interpolated smoothing framework that is applicable to a richer set of objects such as graphs by using a Directed Acyclic Graph (DAG).

4.2.1 Smoothing Multinomial Distributions

Let e_1, e_2, \dots, e_m be a sequence of n discrete events drawn from a ground set $\mathcal{A} = \{1, 2, \dots, V\}$. Suppose we would like to estimate the probability $P(e_i = a)$ for some $a \in \mathcal{A}$. It is well known that the Maximum likelihood estimate (MLE) can be computed as follows:

$$P_{MLE}(e_i = a) = \frac{c_a}{m} \quad (4.1)$$

where c_a denotes the number of times the event a appears in the observed sequence and $\sum_j c_j = m$ denotes the total number of observed events.

However, MLE estimates of the multinomial distribution are *spiky* since they assign zero probability to the events that did not occur in the observed sequence. What this means is that an event with low probability is often estimated to have zero probability mass. The general idea behind smoothing is to adjust the maximum likelihood estimate of the probabilities by pushing the high probabilities downwards and pushing low or zero probabilities upwards in order to produce a more accurate distribution on the events [57].

Laplace smoothing, or so-called *additive smoothing* [58], is perhaps the simplest and one of the oldest smoothing methods, where only a fixed count of α is added to every event. In the case of $\alpha = 1$, this results in the estimate

$$P_L(e_i = a) = \lambda P_{MLE}(e_i = a) + (1 - \lambda) \frac{1}{V}, \quad (4.2)$$

where $\lambda = \frac{m}{m+V}$ is a normalization factor which ensures that the distributions sum to one. The intuition behind Laplace smoothing is to interpolate a uniform distribution with the MLE distribution. Although Laplace smoothing resolves the zero-count problem, it takes away too much probability from seen events and assigns too much probability to unseen events which is undesirable.

Interpolated smoothing methods offer a middle ground by using a linear interpolation between the higher-order maximum likelihood model and lower-order smoothed model (or so-called, *fallback* model). The way the fallback model is designed is the key to defining a new smoothing method. Absolute discounting [59] and Interpolated Kneser-Ney [60] are two popular instances of interpolated smoothing methods:

$$P_A(e_i = a) = \frac{\max\{c_a - d, 0\}}{m} + \frac{m_d \cdot d}{m} P'_A(e_i = a). \quad (4.3)$$

Here, $d > 0$ is a discount factor, $m_d := |\{a : c_a > d\}|$ is the number of events whose counts are larger than d , while P'_A is the fallback distribution. Absolute discounting defines the fallback distribution as the smoothed version of the lower-order MLE while Kneser-Ney uses an unusual estimate of the fallback distribution by using number of different

contexts that the event follows in the lower order model. In the next section, we propose a new interpolated smoothing framework that is applicable to a richer set of objects such as graphs.

4.2.2 Structural Smoothing

The key to designing a new smoothing method is to define a fallback distribution, which not only incorporates domain knowledge but is also easy to estimate recursively. Suppose, we have access to a weighted DAG where every node at the k -th level represents an event from the ground set \mathcal{A} . Moreover let w_{ij} denote the weight of the edge connecting event i to event j , and \mathcal{P}_a (resp. \mathcal{C}_a) denote the parents (resp. children) of event $a \in \mathcal{A}$ in the DAG. We define our structural smoothing for events at level k as follows:

$$P_{SS}^k(e_i = a) = \frac{\max\{c_a - d, 0\}}{m} + \frac{m_d \times d}{m} \sum_{j \in \mathcal{P}_a} P_{SS}^{k-1}(j) \frac{w_{ja}}{\sum_{a' \in \mathcal{C}_j} w_{ja'}}. \quad (4.4)$$

The way to understand the above equation is as follows: we subtract a fixed discounting factor d from every observed event which accumulates to a total mass of $m_d \times d$. Each event a receives some portion of this accumulated probability mass from its parents. The proportion of the mass that a parent j at level $k - 1$ transmits to a given child a depends on the weight w_{ja} between the parent and the child (normalized by the sum of the weights of the edges from j to all its children), and the probability mass $P_{SS}^{k-1}(j)$ that is assigned to node j . In other words, the portion a child event a is able to obtain from the total discounted mass depends on how authoritative its parents are, and how strong the relationship between the child and its parents.

Designing the DAG In order to construct a DAG for smoothing structured objects, we first construct a vocabulary V that denotes the set of all unique sub-structures that are going to be *smoothed*. Each item in the vocabulary V corresponds to a node in the DAG. V can be generated statically or dynamically based on the type of sub-structure the graph kernel exploits. For instance, it requires a one-time $O(2^k)$ effort to generate the vocabulary of size $\leq k$ graphlets for graphlet kernel. However, we need to build the vocabulary dynamically in

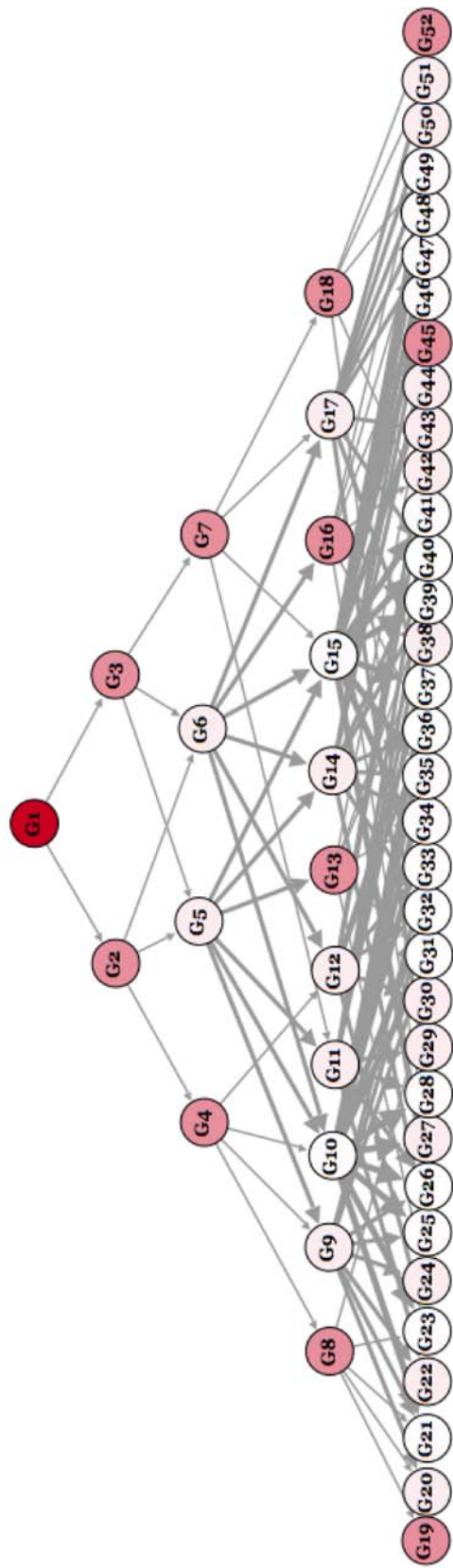


Figure 4.1. Topologically sorted graphlet DAG for $k \leq 5$ where nodes are colored based on their in-degree.

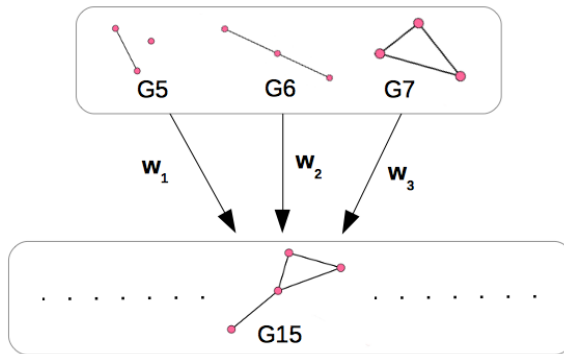


Figure 4.2. Graphlet G_{15} gets the probability mass to its parents G_7, G_6, G_5 according to the weights w_1, w_2, w_3 respectively.

Weisfeiler-Lehman and Shortest-Path kernels since the sub-structures depend on the node labels obtained from the datasets. After constructing the vocabulary V , the parent/child relationship between sub-structures needs to be obtained. Given a sub-structure s of size k , we apply a *transformation* to find all possible sub-structures of size $k - 1$ that s can be reduced into. Each sub-structure s' that is obtained by this transformation is assigned as a *parent* of s . After obtaining the parent/child relationship between sub-structures, the DAG is constructed by drawing a directed edge from each parent to its children nodes. Since all descendants of a given sub-structure at depth $k - 1$ are at depth k , this results in a topological ordering of the vertices, and hence the resulting graph is indeed a DAG. Next, we discuss how to construct such DAGs for different graph kernels.

DAG for Graphlet Kernel: We construct the vocabulary V for Graphlet Kernel by enumerating all canonical graphlets of size up to k^1 . Each canonically-labeled graphlet is a node in the DAG. We then apply a transformation to infer the parent/child relationship between graphlets as follows: we place a directed edge from graphlet G to G' if, and only if, G can be obtained from G' by deleting a node. In other words, all edges from a graphlet G of size $k - 1$ point to a graphlet G' of size k . In order to assign weights to the edges, given a graphlet pair G and G' , we count the number of times G can be obtained from G'

¹We used Nauty [54] to obtain canonically-labeled isomorphic representations of graphlets.

by deleting a node (call this number $n_{GG'}$). Recall that G is of size $k - 1$ and G' is of size k , and therefore $n_{GG'}$ can at most be k . Let \mathcal{C}_G denote the set of children of node G in the DAG, and $n_G := \sum_{\bar{G} \in \mathcal{C}_G} n_{G\bar{G}}$. Then we define the weight $w_{GG'}$ of the edge connecting G and G' as $n_{GG'}/n_G$. The idea here is that the weight encodes the proportion of different ways of extending G which results in the graphlet G' . For instance, let us consider G_{15} and its parents G_5, G_6, G_7 (see Figure 4.1 for the DAG of graphlets with size $k \leq 5$). Even if graphlet G_{15} is not observed in the training data, it still gets a probability mass proportional to the edge weight from its parents in order to overcome the sparsity problem of unseen data (see Figure 4.2).

DAG for Weisfeiler-Lehman Kernel: The Weisfeiler-Lehman kernel performs an *exact matching* between the compressed multiset labels. For instance, given two labels ABCDE and ABCDF, it simply assigns zero value for their similarity even though two labels have a partial similarity. In order to smooth Weisfeiler-Lehman kernel, we first run the original algorithm and obtain the multiset representation of each graph in the dataset. We then apply a transformation to infer the parent/child relationship between compressed labels as follows: in each iteration of Weisfeiler-Lehman algorithm, and for each multiset label of size k in the vocabulary, we generate its *power set* by computing all subsets of size $k - 1$ while keeping the root node fixed. For instance, the parents of a multiset label ABCDE are {ABCD, ABCE, ABDE, ACDE}. Then, we simply construct the DAG by drawing a directed edge from parent labels to children. Notice that considering only the set of labels generated from the Weisfeiler-Lehman kernel is not sufficient enough for constructing a valid DAG. For instance, it might be the case that none of the possible parents of a given label exists in the vocabulary simply due to the sparsity problem (*e.g.* out of all possible parents of ABCDE, we might only observe ABCE in the training data). Thus, restricting ourselves to the original vocabulary leaves such labels orphaned in the DAG. Therefore, we consider so-called *pseudo parents* as a part of the vocabulary when constructing the DAG. Since the sub-structures in this kernel are data-dependent, we use a uniform weight between a parent and its children.

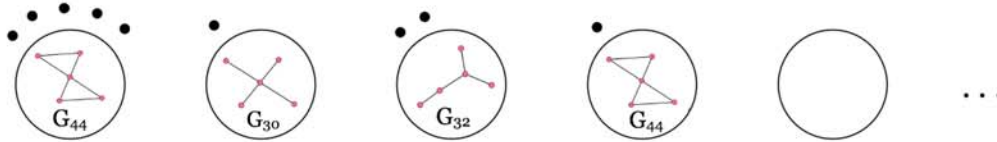


Figure 4.3. An illustration of table assignment, adapted from [61]. In this example, labels at the tables are given by $(l_1, \dots, l_4) = (G_{44}, G_{30}, G_{32}, G_{44})$. Black dots indicate the number of occurrences of each label in 10 draws from the Pitman-Yor process.

DAG for Shortest-Path Kernel: Similar to other graph kernels discussed above, shortest-path graph kernel does not take partial similarities into account. For instance, given two shortest-paths $ABCDE$ and $ABCDF$ (compressed as $AE5$ and $AF5$, respectively), it assigns zero for their similarity since their sink labels are different. However, one can notice that shortest-path sub-structures exhibit a strong dependency relationship. For instance, given a shortest-path $p_{ij} = \{ABCDE\}$ of size k , one can derive the shortest-paths $\{ABCD, ABC, AB\}$ of size $< k$ as a result of the *optimal sub-structure* property, that is, one can show that all sub-paths of a shortest-path are also shortest-paths with the same source node [39]. In order to smooth shortest-path kernel, we first build the vocabulary by computing all shortest-paths for each graph. Let p_{ij} be a shortest-path of size k and $p_{ij'}$ be a shortest-path of size $k - 1$ that is obtained by removing the sink node of p_{ij} . Let l_{ij} be the compressed form of p_{ij} that represents the sorted labels of its endpoints i and j concatenated to its length (resp. $l_{ij'}$). Then, in order to build the DAG, we draw a directed edge from $l_{ij'}$ of depth $k - 1$ to l_{ij} of depth k if and only if $p_{ij'}$ is a sub-path of p_{ij} . In other words, all ascendants of l_{ij} consist of the compressed labels obtained from sub-paths of p_{ij} of size $< k$. Similar to Weisfeiler-Lehman kernel, we assign a uniform weight between parents and children.

4.2.3 Pitman-Yor Smoothing

Pitman-Yor processes are known to produce power-law distributions [62]. A novel interpretation of interpolated Kneser-Ney is proposed by [56] as approximate inference in a

hierarchical Bayesian model consisting of Pitman-Yor process [63]. By following a similar spirit, we extend our model to adapt Pitman-Yor process as an alternate smoothing framework. A Pitman-Yor process P on a ground set \mathcal{G}_{k+1} of size- $(k+1)$ graphlets is defined via $P_{k+1} \sim PY(d_{k+1}, \theta_{k+1}, P_k)$ where d_{k+1} is a discount parameter, $0 \leq d_{k+1} < 1$, $\theta > -d_{k+1}$ is a strength parameter, and P_k is a base distribution. The most intuitive way to understand draws from the Pitman-Yor process is via the Chinese restaurant process (see Figure 4.3). Consider a restaurant with an infinite number of tables where customers enter the restaurant one by one. The first customer sits at the first table, and a graphlet is assigned to it by drawing a sample from the base distribution since this table is occupied for the first time. The label of the first table is the first graphlet drawn from the Pitman-Yor process. Subsequent customers when they enter the restaurant decide to sit at an already occupied table with probability proportional to $c_i - d_{k+1}$, where c_i represents the number of customers already sitting at table i . If they sit at an already occupied table, then the label of that table denotes the next graphlet drawn from the Pitman-Yor process. On the other hand, with probability $\theta_{k+1} + d_{k+1}t$, where t is the current number of occupied tables, a new customer might decide to occupy a new table. In this case, the base distribution is invoked to label this table with a graphlet. Intuitively the reason this process generates power-law behavior is because popular graphlets which are served on tables with a large number of customers have a higher probability of attracting new customers and hence being generated again, similar to a rich gets richer phenomenon. In a hierarchical Pitman-Yor process, the base distribution P_k is recursively defined via a Pitman-Yor process $P_k \sim PY(d_k, \theta_k, P_{k-1})$. In order to label a table, we need a draw from P_k , which is obtained by inserting a customer into the corresponding restaurant. However, adopting the traditional hierarchical Pitman-Yor process is not straightforward in our case since the *size* of the context differs between levels of hierarchy, that is, a *child* restaurant in the hierarchy can have more than one *parent* restaurant to request a label from. In other words, P_{k+1} is defined over \mathcal{G}_{k+1} of size n_{k+1} while P_k is defined over \mathcal{G}_k of size $n_k \leq n_{k+1}$. Therefore, one needs a *transformation function* to transform base distributions of different sizes. We incorporate edge weights between parent and child restaurants by using the same weighting scheme in Section 4.2.2.

Algorithm 2 Insert a Customer

Input: $d_{k+1}, \theta_{k+1}, P_k$
 $t \leftarrow 0$ // Occupied tables

 $c \leftarrow ()$ // Counts of customers

 $l \leftarrow ()$ // Labels of tables

if $t = 0$ **then**
 $t \leftarrow 1$

 append 1 to c

 draw graphlet $G_i \sim P_k$ // Insert customer in parent

 draw $G_j \sim w_{ij}$

 append G_j to l
return G_j
else

 with probability $\propto \max(0, c_j - d)$
 $c_j \leftarrow c_j + 1$
return l_j

 with probability proportional to $\theta + dt$
 $t \leftarrow t + 1$

 append 1 to c

 draw graphlet $G_i \sim P_k$ // Insert customer in parent

 draw $G_j \sim w_{ij}$

 append G_j to l
return G_j
end if

This changes the Chinese Restaurant process as follows: When we need to label a table, we will first draw a size- k graphlet $G_i \sim P_k$ by inserting a customer into the corresponding restaurant. Given G_i , we will draw a size- $(k + 1)$ graphlet G_j proportional to w_{ij} , where w_{ij} is obtained from the DAG. See Algorithm 2 for pseudo code of inserting a customer. Deletion of a customer is handled similarly (see Algorithm 3).

Algorithm 3 Delete a Customer

Input: d, θ, P_0, C, L, t

with probability $\propto c_l$

$c_l \leftarrow c_l - 1$

$G_j \leftarrow l_j$

if $c_l = 0$ **then**

$P_k \propto 1/w_{ij}$

delete c_l from c

delete l_j from l

$t \leftarrow t - 1$

end if

return G

4.2.4 Other Smoothed Graph Kernels

In a similar fashion to Smoothed Weisfeiler-Lehman subtree kernel, Smoothed Shortest Path kernel and Smoothed Graphlet Kernel, we can plug other graph kernels into our smoothing framework. As discussed in earlier sections, the key aspect of our smoothing framework is defining the DAG which encodes the similarity between different sub-structures. Therefore, a clear requirement is that the base kernel should exhibit dependency and similarity among its features so that defining a fallback distribution would make sense and we can exploit the relationship between different sub-structures by smoothing. On the other hand, in graph kernels where feature representation is infinite, such as random walk

kernels [40], one needs to consider whether smoothing can be done efficiently. Other possible base kernels for our smoothing framework would be the labeled version of graphlet kernel [16], subtree kernels including [28] and [41], cyclic pattern kernels [27] and p -step random walk kernel [42].

As discussed in the earlier sections, graph kernels are instances of R-convolution kernels. Thus, our smoothing framework is applicable to any R-convolution kernel that has an inherent dependency and similarity between its features where features are represented by a vector of frequencies such as K-Spectrum string kernel [43].

4.3 Related Work

A survey of most popular graph kernel methods is already given in previous sections. Several methods proposed in smoothing structured objects [64], [65]. Our framework is similar to dependency tree kernels [64] since both methods are using the notion of smoothing for structured objects. However, our method is interested in the problem of smoothing the count of structured objects. Thus, while smoothing is achieved by using a DAG, we discard the DAG once the counts are smoothed. Another related work to ours is propagation kernels [66] that define graph features as counts of similar node-label distributions on the respective graphs by using Locality Sensitive Hashing (LSH). Our framework not only considers node label distributions, but also explicitly incorporates structural similarity via the DAG.

4.4 Experiments

The aim of our experiments is based on five aspects. First, we want to understand characteristics of feature space in three popular graph kernels, and motivate why smoothing is necessary. Then, we want to show that smoothing improves the classification accuracy of various graph kernels. Third, we want to show that the pre-processing cost of creating a DAG and smoothing is not prohibitively expensive. Then, we want to show that the smoothed kernels are comparable to or outperform state-of-the-art graph kernels in terms

of classification accuracy, while remaining competitive in terms of computational requirements. Finally, we want to show that our methods significantly outperforms base kernels when edge or label noise is presence.

4.4.1 Experimental Setup

We compare our framework against representative instances of major families of graph kernels in the literature. In addition to the base kernels, we also compare our smoothed kernels with the random walk kernel [40], the Ramon-Gärtner subtree [28], and p -step random walk kernel [42]. The Random Walk, p -step Random Walk and Ramon-Gärtner are written in Matlab and obtained from [16]. All other kernels were coded in Python except Pitman-Yor smoothing which is coded in C++². We used a parallel implementation for smoothing the counts of Weisfeiler-Lehman kernel for efficiency. All kernels are normalized to have a unit length in the feature space. Moreover, we use 10-fold cross validation with a binary C -Support Vector Machine (SVM) where the C value for each fold is independently tuned using training data from that fold. In order to exclude random effects of the fold assignments, this experiment is repeated 10 times and average prediction accuracy of 10 experiments with their standard deviations are reported.

4.4.2 Datasets

We used the following benchmark datasets used in graph kernels: MUTAG, PTC, ENZYMES, PROTEINS, NCI1 and NCI109 (see Section 3.4.2 for a detailed description of the datasets).

4.4.3 Analyzing Feature Space

In this experiment, we investigate the characteristics of the feature space in Graphlet Kernels, Weisfeiler-Lehman kernels and Shortest-Path Kernels on benchmark datasets. For

²We modified the open source implementation of PYP: <https://github.com/redpony/cpyp>.

Table 4.1.
Graphlet kernel feature statistics for benchmark datasets that shows median, maximum and minimum number of features.

| DATA SET | NUMBER OF FEATURES | MEDIAN | MAX | MIN |
|----------|--------------------|--------|-----|-----|
| MUTAG | 209 | 16 | 20 | 12 |
| PTC | 209 | 14 | 31 | 1 |
| ENZYMES | 209 | 26 | 97 | 1 |
| PROTEINS | 209 | 23 | 89 | 1 |
| NCI1 | 209 | 13 | 33 | 1 |
| NCI109 | 209 | 13 | 31 | 1 |

this purpose, we run the unsmoothed graph kernels on the benchmark datasets, and collect median, max and min statistics for number of unique features. Table 4.1 shows feature statistics for Graphlet Kernel where each feature corresponds to a graphlet. As we can see from the table, even though there are 209 possible graphlets for size $k = 6$, we only observe a few of them on each dataset. Table 4.2 shows feature statistics for Weisfeiler-Lehman graph kernels, where each feature corresponds to a compressed label. One can see that sparsity problem is much more severe in Weisfeiler-Lehman graph kernels. For instance, while the unique number of features is 15,208 on Enzymes dataset, only 16 on median is observed per graph. Similarly, Table 4.3 shows the feature statistics for shortest-path graph kernel where each feature corresponds to a shortest-path label. Similar to other graph kernels, feature space is sparse. For instance, while there are 1,084 unique shortest-path feature in NCI1 dataset, only 47 feature is observed per graph on median. Therefore, one can benefit from applying smoothing in order to reduce the data sparsity.

4.4.4 Parameter Selection

We chose parameters for the various kernels as follows: discount parameter for smoothed kernels is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 0.25, 0.50, 0.75\}$, decay factor for random walk

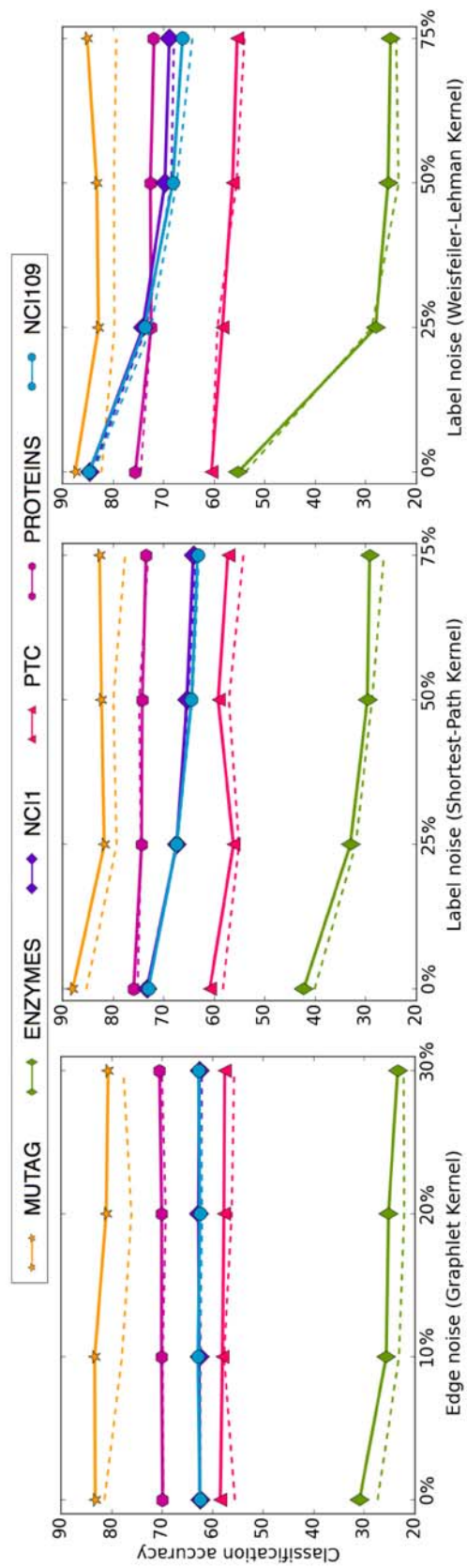


Figure 4.4. Classification accuracy vs. noise for base graph kernels (dashed lines) and their smoothed variants (non-dashed lines).

Table 4.2.

Weisfeiler-Lehman kernel feature statistics for benchmark datasets that shows median, maximum and minimum number of features.

| DATA SET | NUMBER OF FEATURES | MEDIAN | MAX | MIN |
|----------|--------------------|--------|-----|-----|
| MUTAG | 572 | 8 | 23 | 1 |
| PTC | 2624 | 10 | 57 | 1 |
| ENZYMES | 15208 | 16 | 83 | 1 |
| NCI1 | 22948 | 12 | 92 | 1 |
| NCI109 | 23411 | 12 | 83 | 1 |

Table 4.3.

Shortest-Path kernel feature statistics for benchmark datasets that shows median, maximum and minimum number of features.

| DATA SET | NUMBER OF FEATURES | MEDIAN | MAX | MIN |
|----------|--------------------|--------|-----|-----|
| MUTAG | 151 | 28 | 64 | 19 |
| PTC | 830 | 42 | 345 | 3 |
| ENZYMES | 178 | 31 | 139 | 2 |
| NCI1 | 1084 | 47 | 306 | 4 |
| NCI109 | 1053 | 47 | 311 | 8 |

kernels is chosen from $\{10^{-6}, 10^{-5}, \dots, 10^{-1}\}$, the p value in the p -step random walk kernel is chosen from $\{1, 2, \dots, 10\}$ and the height parameter in Ramon-Gärtner subtree kernel is chosen from $\{1, 2, 3\}$. For each kernel, we report the results for the parameter which gave the best classification accuracy. For Weisfeiler-Lehman kernel, we experimented the height parameter up to $h \leq 5$ due to exponentially increasing feature space of the original kernel. In all cases, since the relative classification performance of the original kernel and our proposed smoothed variant remain consistent, we fix the height parameter at $h = 3$ for the rest of the experiments. For the graphlet kernel, we set the size of the graphlets k to be

Table 4.4.

List of parameters used in experiments for Graphlet (GK), Weisfeiler-Lehman (WL) and Shortest-Path (SP) kernels, and their smoothed variants Smoothed Graphlet (S-GK), Smoothed Weisfeiler-Lehman (S-WL) and Smoothed Shortest-Path (S-SP) kernels. Additionally, Ramon & Gärtner (RG), p -Random Walk (p-RW) and Random Walk (RW) parameters are listed for each dataset. Parameter selection for base SP kernel is omitted since it does not use a parameter.

| Data set | GK | S-GK | WL | S-WL | S-SP | RG | p-RW | RW |
|----------|-----|--------------|-----|---------------|---------|-----|---------------------|-------|
| MUTAG | k=6 | k=6, d= 0.25 | h=3 | h=3, d= 0.75 | d=0.01 | h=2 | $\lambda = 10^{-2}$ | p =5 |
| PTC | k=6 | k=6, d= 0.25 | h=3 | h=3, d= 0.001 | d=0.25 | h=1 | $\lambda = 10^{-2}$ | p = 4 |
| ENZYMES | k=6 | k=6, d= 0.1 | h=3 | h=3, d= 0.001 | d=0.25 | h=2 | $\lambda = 10^{-4}$ | p = 7 |
| PROTEINS | k=6 | k=6, d= 0.01 | h=3 | h=3, d= 0.001 | d=0.25 | h=2 | $\lambda = 10^{-4}$ | p = 7 |
| NCI1 | k=6 | k=6, d= 0.25 | h=3 | h=3, d= 0.001 | d=0.001 | h=1 | – | – |
| NCI109 | k=6 | k=6, d= 0.01 | h=3 | h=3, d= 0.001 | d=0.001 | h=1 | – | – |

6 since it exhibits the sparsity problem that we are interested in. Counting all graphlets of size k for a graph with n nodes requires $O(n^k)$ effort which is intractable even for moderate values of k . Therefore, we use random sampling, as advocated by [16], in order to obtain an empirical distribution of graphlet counts that is close to the actual distribution of graphlets in the graph. For each graph, we first randomly sampled 1000 graphlets of size 6 and then within the sampled graphlets, we search for graphlets of size $2 \leq k < 6$ (in other words, we first fix a window of size $k = 6$, and then we look at all possible lower-order graphlets). This sampling method allows us to ensure that random sampling between levels are consistent. Table 4.4 lists all parameter values used in the experiments.

4.5 Results

In our first experiment, we compare the base kernels with their smoothed variants. As can be seen from Table 4.5, smoothing improves the classification accuracy of *every base kernel* on *every dataset* with majority of the improvements being statistically significant with $p \leq 0.05$. We observe that even though smoothing improves the accuracy of graphlet kernels on PROTEINS and NCI1, the improvements are not statistically significant. We believe this is due to the fact that these datasets are not sensitive to structural noise as much as the other datasets, thus considering the partial similarities do not improve the results significantly. Moreover, PYP smoothed graphlet kernels achieve statistically significant improvements in most of the datasets, however they are outperformed by smoothed graphlet kernels introduced in Section 4.2.1.

In our second experiment, we picked the best smoothed kernel in terms of classification accuracy for each dataset, and compared it against the performance of state-of-the-art graph kernels (see Table 4.7). Smoothed kernels outperform other methods on all datasets, and the results are statistically significant on every dataset except PTC.

In our third experiment, we investigated the runtime behavior of our framework with two major costs. First, one has to compute a DAG by using the original feature vectors. Next, the constructed DAG need to be used to compute *smoothed* representations of the

Table 4.5. Comparison of classification accuracy (\pm standard deviation) of shortest-path (SP), Weisfeiler-Lehman (WL), graphlet (GK) kernels with their smoothed variants (statistically significant improvements over the base kernels are shown in bold as measured by a t -test with a p value of ≤ 0.05).

| DATASET | MUTAG | PTC | ENZYMES | PROTEINS | NCI1 | NCI109 |
|-------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| SP | 85.22 \pm 2.43 | 58.24 \pm 2.44 | 40.10 \pm 1.50 | 75.07 \pm 0.54 | 73.00 \pm 0.24 | 73.00 \pm 0.21 |
| SMOOTHED SP | 87.94 \pm 2.58 | 60.82 \pm 1.84 | 42.27 \pm 1.07 | 75.85 \pm 0.28 | 73.26 \pm 0.24 | 73.01 \pm 0.31 |
| WL | 82.22 \pm 1.87 | 60.41 \pm 1.93 | 53.88 \pm 0.95 | 74.49 \pm 0.49 | 84.13 \pm 0.22 | 83.83 \pm 0.31 |
| SMOOTHED WL | 87.44 \pm 1.95 | 60.47 \pm 2.39 | 55.30 \pm 0.65 | 75.53 \pm 0.50 | 84.66 \pm 0.18 | 84.72 \pm 0.21 |
| GK | 81.33 \pm 1.02 | 55.56 \pm 1.46 | 27.32 \pm 0.96 | 69.69 \pm 0.46 | 62.46 \pm 0.19 | 62.33 \pm 0.14 |
| SMOOTHED GK | 83.17 \pm 0.64 | 58.44 \pm 1.00 | 30.90 \pm 1.51 | 69.83 \pm 0.46 | 62.48 \pm 0.15 | 62.48 \pm 0.11 |
| PYP GK | 83.11 \pm 1.23 | 57.44 \pm 1.44 | 29.63 \pm 1.30 | 70.00 \pm 0.80 | 62.50 \pm 0.20 | 62.68 \pm 0.18 |

Table 4.6.
 Runtime for constructing the DAG and smoothing the counts are also reported where '' indicates seconds and ' indicates minutes.

| DATASET | MUTAG | PTC | ENZYMES | PROTEINS | NCI1 | NCI109 |
|---------------------|---------|----------|-----------|----------|----------|-----------|
| DAG/SMOOTHING (GK) | 6" / 1" | 6" / 1" | 6" / 1" | 6" / 1" | 6" / 3" | 6" / 3" |
| DAG/SMOOTHING (SP) | 3" / 1" | 19" / 1" | 45" / 1" | 9' / 1" | 9' / 17" | 10' / 16" |
| DAG/SMOOTHING (WL) | 1" / 2" | 1" / 17" | 10" / 12' | 7' / 70' | 2" / 21' | 2" / 21' |
| DAG/SMOOTHING (PYP) | 6" / 5" | 6" / 12" | 6" / 21" | 6" / 1' | 6" / 8' | 6" / 8' |

Table 4.7.
 Comparison of classification accuracy (\pm standard deviation) of Ramon & Gärtner, p -random walk and random walk kernels where $> 72H$ indicates the computation did not finish in 72 hours.

| DATASET | MUTAG | PTC | ENZYMES | PROTEINS | NCI1 | NCI109 |
|---------------|------------------|------------------|------------------|------------------|------------------|------------------|
| RAM & GÄRTNER | 84.88 \pm 1.86 | 58.47 \pm 0.90 | 16.96 \pm 1.46 | 70.73 \pm 0.35 | 56.61 \pm 0.53 | 54.62 \pm 0.23 |
| P-RANDOM-WALK | 80.05 \pm 1.64 | 59.38 \pm 1.66 | 30.01 \pm 1.00 | 71.16 \pm 0.35 | $> 72H$ | $> 72H$ |
| RANDOM WALK | 83.72 \pm 1.50 | 57.85 \pm 1.30 | 24.16 \pm 1.64 | 74.22 \pm 0.42 | $> 72H$ | $> 72H$ |

feature vectors. Table 4.6 shows the total wallclock runtime taken by *all graphs* for constructing the DAG, and smoothing the counts for each dataset. As can be seen from the runtimes, our framework adds a constant factor to the original runtime for most of the datasets. While the DAG creation in Weisfeiler-Lehman kernel also adds a negligible overhead, the cost of smoothing becomes significant if the vocabulary size gets prohibitively large due to the exponential growing nature of the kernel *w.r.t.* to subtree parameter h .

Finally, in our fourth experiment, we test the performance of graph kernels when *edge* or *label* noise is present. For edge noise, we randomly removed and added $\{10\%, 20\%, 30\%\}$ of the edges in each graph. For label noise, we randomly flipped $\{25\%, 50\%, 75\%\}$ of the node labels in each graph where random labels are selected proportionally to the original label-distribution of the graph. Figure 4.4 shows the performance of smoothed graph kernels under noise. As can be seen from the figure, smoothed kernels are able to outperform their base variants when noise is present. An interesting observation is that even though a significant amount of edge noise is added to PROTEINS and NCI datasets, the performance of base kernels do not change drastically. This further supports our observation that these datasets are not sensitive to structural noise as much as the other datasets.

4.6 Conclusions

We presented a novel framework for smoothing graph kernels inspired by smoothing techniques from natural language processing and applied our method to state-of-the-art graph kernels. Our framework is rather general, and lends itself to many extensions. For instance, by defining domain-specific parent-child relationships, one can construct different DAGs with different weighting schemes.

Moreover, even though we restricted ourselves to graph kernels in this study, our framework is applicable to any R-convolution kernel that uses a frequency-vector based representation, such as *string kernels*.

Part II

Information Overload in Text

5 A SUBMODULAR RECOMMENDER SYSTEM FOR REDDIT

Reddit is one of the largest community-driven content aggregation websites. Approximately, 6% of online adults are estimated to be Reddit users [20]. The immense popularity of Reddit can be attributed to its ability to surface freshest trends and content on the web, and it is commonly referred as the *front page of the Internet*. With over 800K communities devoted to various topics, a natural information overload problem arises. Given that thousands of new threads and discussion topics generated hourly, providing the best coverage of content that is relevant to the interests of the users becomes a critical task.

In this chapter, we propose a framework that tailors a *personalized* frontpage of the Internet. We formulate our framework as a submodular optimization problem, for which we can efficiently provide a near-optimal solution. We evaluate our framework both with offline and online experiments, and empirically demonstrate that our algorithm respects personal preference of the users, while presenting top stories on the web.

5.1 Introduction

Reddit is one of the largest community-driven social news and entertainment services on the web, often referred as *the frontpage of the Internet*. According to the site statistics¹, Reddit has over 234 million unique visitors and 3.1 million logged-in users from 217 different countries on a monthly basis. Moreover, it is ranked as the 32nd most popular website in the world, and 10th most popular website in United States². Users on Reddit, also referred as *redditors*, can contribute by creating a new post, so-called *self-posts*, by submitting an external link or by leaving a comment on an already existing submission. Posted content is then organized by communities called *subreddits*, representing various areas of interests

¹<http://www.reddit.com/about>, as of Feb 1, 2016.

²<http://www.alexa.com/siteinfo/reddit.com>, as of Feb 1, 2016.

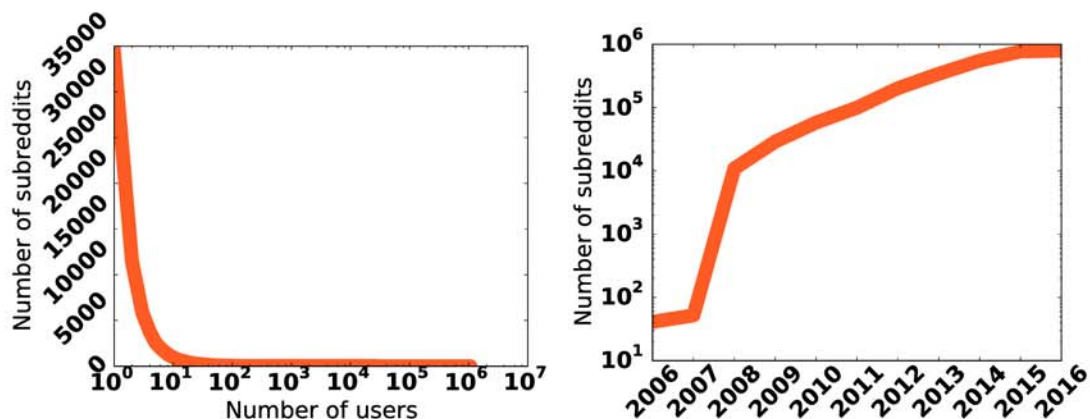


Figure 5.2. Growth of number of unique subreddits over time between 2006 to 2016 (y-axis is log-scaled) (left). Histogram based on user frequency per subreddit (x-axis is log-scaled) (right).

One way for users to mitigate this problem is to search for relevant content on Reddit, and subscribe to subreddits of interest. However, due to the decentralized nature of subreddits which allows any registered user to create their own community on any topic, the number of subreddits has grown exponentially over the past few years (see Figure 5.2 (right)). Today, there are more than 850,000 user-created subreddits³ with 11,200 of them being active on a daily basis (see Table 5.1 for a list of sample subreddits). Although there are various subreddits serving niche areas of interests, it may be difficult for the user to find them. For instance, in addition to having a general subreddit on `politics` topic, there are over 100 fine-grained politics-related subreddits⁴, some of which are titled as `Liberal`, `Republican`, `Anarchism`, `Socialism`, `Progressive` and `Marxism`.

Moreover, due to the rapidly changing content and trends on the web, users develop *temporal* and *topical* interests that change over time. For instance, a user might be visiting New York and subscribe to the `nyc` subreddit to get familiar with the city. Consider a scenario where, the interest of the user in this particular subreddit fades after a couple of weeks, and the user does not want to be exposed to the `nyc`-related posts as frequent as

³“Celebrating the best of 10 years of Reddit”: <http://tinyurl.com/subreddit-stats>

⁴“List of political subreddits”: <http://tinyurl.com/political-subreddits>

Table 5.1.
 Sample posts from subreddits with short descriptions (see corresponding links from <http://tinyurl.com/sample-subreddits>).

| Subreddit | Subreddit description and a sample post |
|--------------------|---|
| TodayILearned | Users post interesting and specific facts they found out, e.g. <i>“TIL: \$41 billion worth of gift cards have likely gone unredeemed from 2005 to 2011”</i> . |
| IAMA | Users having something interesting to share invites other users to ask them questions, e.g. <i>e.g. “I spent 18 years in prison for a murder I didn’t commit. AMA.”</i> |
| UpliftingNews | Users post uplifting, inspirational and feel good news stories from around the globe, e.g. <i>“This Man Builds Tiny Houses for the Homeless, Out of Recyclable Materials”</i> . |
| ShowerThoughts | Users post their thoughts, or philosophical questions that they have when in the shower, e.g. <i>“What if the lottery is an Institution to catch Time Travelers?”</i> . |
| PersonalFinance | Users help each other to learn how to better manage their money and debt, e.g. <i>“Should I put all my child’s savings into bitcoin?”</i> |
| LifeProTips | Users share best tips they’ve picked up throughout their life, e.g. <i>“Freeze grapes to chill white wine without watering it down.”</i> |
| ExplainLikeIAmFive | Users ask questions to understand certain topics in depth with simplified explanations, e.g. <i>“How did knowing Einstein’s theory of relativity lead scientists to make the first atom bomb?”</i> |

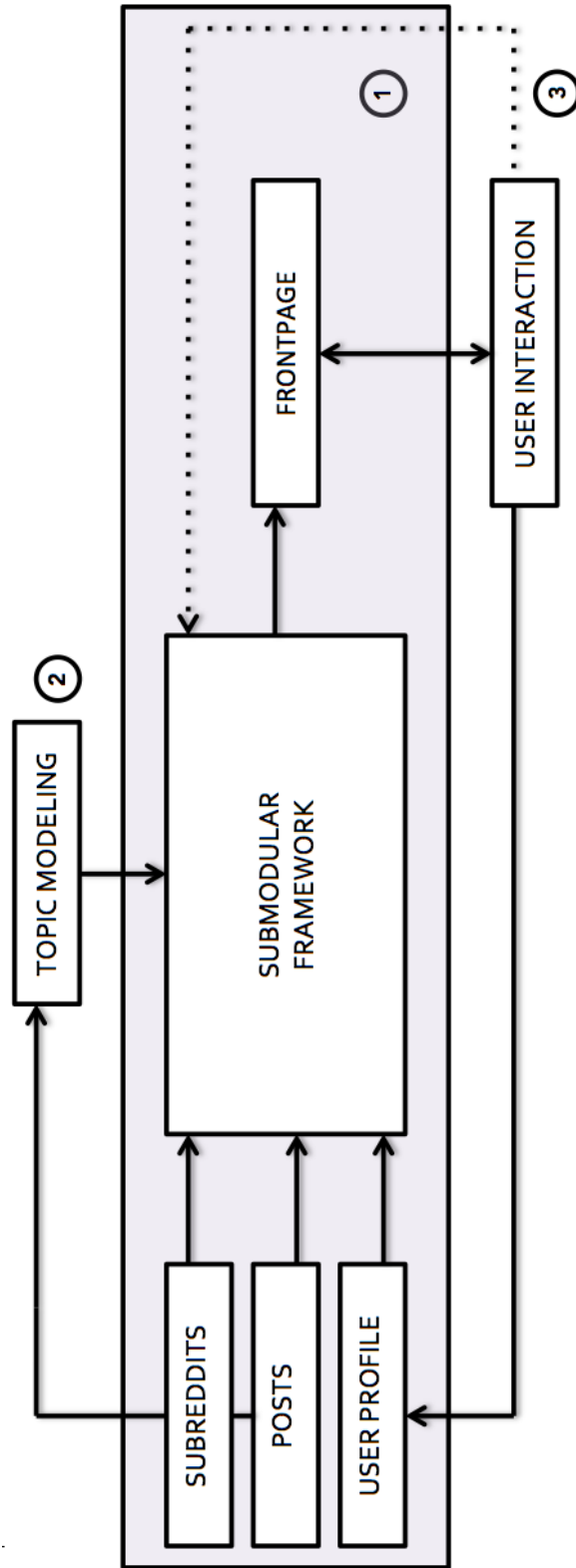


Figure 5.3. A diagram that shows different components of our framework. In particular, 1 represents the core framework that covers *subreddits*, 2 represents the extension of our framework into coverage of topics, and 3 represents the personalization component of our framework

before. However, since the subscription-based model enforces a strong commitment, the user is overwhelmed with posts that are not relevant anymore. Thus, we conjecture that expressing the interests of the user in terms of *topics* is a more natural way to capture the temporal interests of the users. Hence, the interest of the user towards the `nyc` subreddit might fade, but it would be still possible to capture the fact that the user is interested in *travel* topic.

The goal of this study is to address the above problems by formulating a framework that not only suggests relevant and diverse content that users are interested in, but also encourages discoverability by allowing us to inject posts from related subreddits. Our technical contributions are as follows: we first present a simple and elegant solution to cover posts from subreddits a user is subscribed to. We formulate an objective function that exhibits a natural diminishing returns property, also known as *submodularity*, for which we can efficiently provide a near-optimal solution [67].

Secondly, we project subreddits into a topical space by building a topic model using the textual content of the posts, and identify the related subreddits. We then extend the coverage of subreddits to *coverage of topics* which enables discoverability of novel content and communities.

Finally, we extend our model to learn a *personalized* coverage function. We formulate an interaction model by considering user feedback and learn preference of individual users. We evaluate our framework by performing offline evaluation on data from real users from Reddit. Results on average click position shows that our simple, subreddit-coverage based algorithm significantly outperforms the baseline methods. Moreover, we perform a user study by measuring diversity, discoverability and personalization. Results from the user study show that our methods provides better diversity and discoverability comparing to the baseline approach.

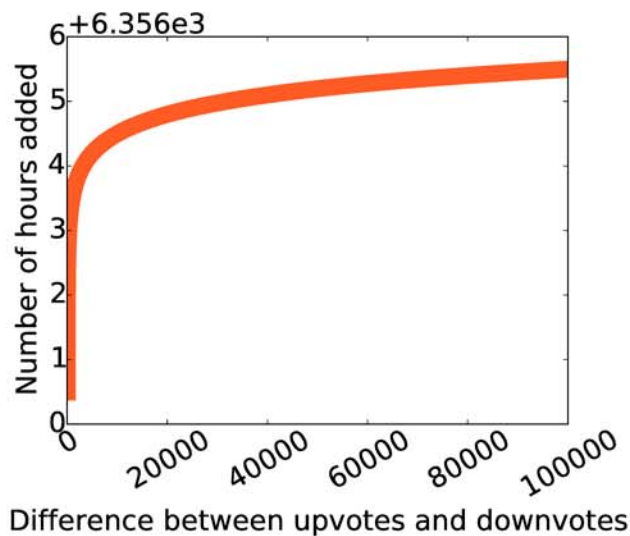


Figure 5.4. Hot-score as a function of increasing upvote and downvote difference.

5.2 Methodology

We first discuss an overview of our framework. Figure 5.3 illustrates the three key components of our framework. The core framework, discussed in Section 5.2.2, takes a user, a set of subreddits user is subscribed to, and a set of posts, and generates a *frontpage* that covers diverse *subreddits* user is interested in (indicated with 1).

We then build a topic model using subreddits (indicated with 2), and extend our framework to cover diverse set of *topics*, as discussed in Section 5.2.3. This component encourages a stronger notion of coverage, by means of *topics* and also encourages *discoverability* of new content and subreddits.

Finally, we extend our framework to learn a two-step personalized coverage function (indicated with 3). This component allows us to *learn* personalized preference of the individual users by collecting their interactions with the previous frontpage, and generates a new, *personalized frontpage*. We discuss this component in Section 5.2.4.

5.2.1 Coverage

Given the dynamic environment of Reddit with thousands of new posts generated hourly, our aim is to provide a good coverage of posts from subscribed subreddits of a user. Formally, let \mathcal{I} denote the set of subreddits a user has subscribed to, and \mathcal{P}^t denote the set of all candidate posts from \mathcal{I} that can be shown to a user at time t . Given an arbitrary subreddit $i \in \mathcal{I}$, let \mathcal{P}_i^t represent the set of all posts submitted to subreddit i at time t . Then, we are interested in selecting a subset $\mathcal{S} \subseteq \mathcal{P}^t$ that provides the best coverage of subreddits in \mathcal{I} . The first step towards coverage is to characterize posts with a score. Reddit uses an algorithm called *hot-score*⁵ and assigns a score for each post to represent the *freshness* and *popularity* of the post among users. For a post $p \in \mathcal{P}^t$, the hot-score is defined as,

$$\mathcal{H}(p) = \log_{10}(u_p - d_p) + \frac{\Delta t}{45000}, \quad (5.1)$$

where u_p is the number of *upvotes*, d_p is the number of *downvotes*, and Δt is the difference of time in seconds since an arbitrary epoch, *e.g.* 1970-01-01 00:00:00. The first term reflects *popularity* and increases as a logarithmic function of the difference between number of upvotes and downvotes (see Figure 5.4). The second term ensures the *freshness* of content. Since the score does not decrease over time, newer posts always get a higher score than previous posts, and older posts eventually get overtaken by new posts within hours.

Given the scoring function, we can design a cover function to measure how much subreddit $i \in \mathcal{I}$ is covered by posts in \mathcal{S} ,

$$\text{cover}_{\mathcal{S}}(i) = \sum_{j \in \mathcal{P}_i \cap \mathcal{S}} \mathcal{H}(j), \quad (5.2)$$

where \mathcal{P}_i represents all posts that belong to subreddit i , and $\mathcal{H}(j)$ represents hot-score of post j . In order to measure the overall coverage of subreddits in \mathcal{I} , we can define the following utility function,

$$F(\mathcal{S}) = \sum_{i \in \mathcal{I}} \text{cover}_{\mathcal{S}}(i), \quad (5.3)$$

⁵Reddit is open-sourced and its code can be found here: <https://github.com/reddit/reddit>

Table 5.2.
 Top ten most similar subreddits for a given subreddit (the subreddits can be reach via http://reddit.com/r/subreddit_name).

| Fitness | food | trees | MakeupAddiction | hiphopheads |
|----------------|---------------------|------------------|------------------------|--------------------|
| xxfitness | Cooking | weed | makeupexchange | hiphop |
| weightlifting | AskCulinary | saplings | asianbeautyexchng | listentothis |
| Stronglifts5x5 | cookingforbeginners | CannabisExtracts | muacirclejerk | rap |
| ketogains | slowcooking | uktrees | AustralianMakeup | coversongs |
| gainit | FoodPorn | glassheads | Indiemakeupandmore | electronicmsc |
| leangains | EatCheapAndHealthy | COents | Makeup | treemusic |
| loseit | ketorecipes | see | MakeupAddicts | indie |
| keto | tonightsdinner | eldertrees | AsianBeauty | mashups |
| fasting | cookingvideos | treedibles | MakeupRehab | PleasureMusic |
| xxketo | recipes | PipeTobacco | RedditLaqueristas | trapmuzik |

which simply sums over the total coverage of each subreddit the user is subscribed to. Since the number of posts are very high, and users have limited time to discover the content, one might aim to maximize the following objective function,

$$\mathcal{S}^* = \operatorname{argmax}_{\mathcal{S} \subseteq \mathcal{P}: |\mathcal{S}| \leq n} F(\mathcal{S}), \quad (5.4)$$

subject to a cardinality constraint n , which denotes total number of posts in set \mathcal{S}^* . Thus, finding a set of posts the user is interested in becomes a budgeted maximum coverage problem [68]. However, not only maximizing this function is NP-hard, but it also does not respect diversity. In other words, the value of covering a particular subreddit never diminishes. This contradicts with our notion of coverage since we would like to provide a good coverage of all subreddits the user is interested in. One of the popular approaches to solve this problem is to use a submodular function which exhibits a natural diminishing returns property [68, 69]. Next, we discuss a simple solution to substitute Equation (5.3).

5.2.2 Covering Subreddits

One popular approach especially used in document summarization tasks to make a function respect diversity, is to apply a concave function to the coverage [69],

$$F(\mathcal{S}) = \sum_{i=1}^I \sqrt{\operatorname{cover}_{\mathcal{S}}(i)}. \quad (5.5)$$

This objective function satisfies the *submodularity* property which exhibits a natural diminishing returns, that is, given two sets R and S from a finite ground set V where $R \subseteq S \subseteq V \setminus v$, the incremental *value* of an item v decreases as the context in which v is considered grows from R to S . More formally, submodularity is a property of set functions, *i.e.*, the class of functions $f : 2^V \rightarrow \mathbb{R}$ that map subsets $S \subseteq V$ to a value $f(S)$. The function f can be thought as a black box function which maps any given subset to a real number.

Definition 5.2.1 *The function f is called submodular if the following inequality holds for any S : $f(S \cup \{v\}) - f(S) \leq f(R \cup \{v\}) - f(R)$, where f is submodular if $R \subseteq S \subseteq V$*

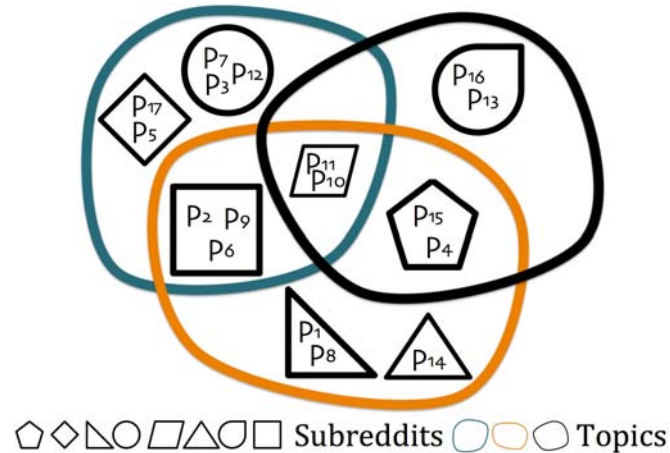


Figure 5.5. An illustration of how posts p_1, \dots, p_n are structured into different subreddits and topics.

and $v \in V \setminus S$. This form of submodularity directly satisfies the property of diminishing returns; the value of adding v never increases when the context gets larger. [67]

Even though finding the exact subset that maximizes Equation (5.5) is intractable, it has been shown that the maximization of a monotone submodular function under a cardinality constraint can be solved near-optimally using a *greedy* algorithm [67]. In particular, if a function f is submodular and monotone, and takes only non-negative values, then a greedy algorithm approximates the optimal solution of Equation (5.5) within a factor of $(1 - 1/e)$ [67]. Due to the large number of subreddits with hundreds of incoming posts every hour, the naive greedy approach is not efficient. Therefore, we use CELF [70] which provides an efficient way to optimize this function based on a lazy-forward optimization.

Before we move on, let us provide some intuition behind Equation (5.5). The idea here is to apply a diminishing return to reward of posts coming from the same subreddit. In other words, the gain of selecting posts from the same subreddit diminishes due to the concave function. By adapting the example of [69], let us demonstrate the intuition behind this concave function. Consider a user is subscribed to two subreddits; AskReddit and worldnews. Assume that candidate posts in AskReddit are p_1 and p_2 , having a hot-score of 5000 and 4000, respectively. Similarly, worldnews has one candidate post, p_3

with a hot-score of 3000. Evaluating this function for the first time on this set selects p_1 since it has the largest marginal gain. However, the next time we select p_3 even though the hot-score of p_2 is higher since $\sqrt{5000 + 4000} < \sqrt{5000} + \sqrt{3000}$. Intuitively, this means that selecting a post from a subreddit that is not yet explored has a higher gain than selecting a post from a subreddit that we already covered. Therefore, the objective function will reward diversity by having items chosen from different subreddits, and will not let popular subreddits to dominate.

However, even though this function respects diversity, it does not consider several important aspects. First, most of the subreddits are often related to each other. For instance, `Cooking`, `cookingvideos`, `cookingforbeginners`, `recipes` are among several cooking-related subreddits covering essentially the same interest. One way to alleviate this problem might be using a topic model on every post and identify similar posts in terms of content. Then, we could select posts covering different topics. However, given the dynamic nature of Reddit with thousands of posts, such approaches are prohibitively expensive. Finally, this framework does not allow users to *discover* new subreddits that are aligned with their interests. Next, we discuss an extension of this framework which addresses above problems.

5.2.3 Covering Topics

There exist a large number of subreddits that share similar characteristics, and they can be grouped into topics. Therefore, instead of covering individual subreddits, one might cover *topics* that a user is interested in. For this purpose, we used Latent Dirichlet Allocation (LDA) [71], a generative model that discovers the underlying topics in a text document by assuming that each document $d \in \mathcal{D}$ is associated with a K -dimensional topic distribution. In other words, each document covers K latent topics, where each topic is defined as a distribution over words drawn from a Dirichlet distribution. Given such a model, an ideal approach would be directly inferring the topics a user is interested in. However, we only have an explicit signal about the interests of the user by means of subscribed subreddits.

Therefore, directly inferring the latent interests of the user is a difficult task. We alleviate this problem as follows. We first construct a set of \mathcal{D} documents where each document d_i represents a subreddit $i \in \mathcal{I}$. The document d_i contains the text of all posts submitted to subreddit i in the dataset. After building a model using K topics, we obtain a topic distribution for a given subreddit i , and use cosine similarity to compute most similar subreddits for i . After that, given a subreddit i that the user is subscribed to, we simply instantiate a *pseudo-topic* which consists of the subreddit i and top m most similar subreddits to i . Intuitively, this means that Reddit is clustered into topics, and topics themselves consist of subreddits, as shown in Figure 5.5. Note that under this assumption, even though two instantiated topics might contain very similar subreddits, their coverage will be adjusted accordingly since our model allows *overlapping* subreddits. In other words, when we cover a post from a subreddit i , the diminishing return property will be reflected to all topics that i is participated in. Table 5.2 illustrates some topics obtained by using this approach. As can be seen from the table, we are able to discover similar subreddits such as `food`, `Cooking`, `cookingforbeginners`. On the other hand, we also discovered some niche subreddits. For instance, given that a user is subscribed to `food` subreddit, we can now recommend `ketorecipes`; a subreddit that shares low-carb keto recipes.

We now extend Equation (5.5) to consider the similarity between subreddits and to inject posts from related subreddits that the user is not subscribed. For this purpose, we formulate the following objective function,

$$F(\mathcal{S}) = \sum_{k \in \mathcal{K}} \sqrt{\sum_{i \in \mathcal{R}_k \cap \mathcal{I}} \varphi_i \text{cover}_{\mathcal{S}}(i)}, \quad (5.6)$$

where \mathcal{K} represents a set of topics, k represent an arbitrary subreddit from this set, \mathcal{R}_k represents the set of subreddits that belongs to topic k , \mathcal{I} represents the set of subreddits in the corpus, and $\text{cover}_{\mathcal{S}}(i)$ represents how much subreddit i is covered by posts in set \mathcal{S} , as in Equation (5.2). φ_i is simply added as a penalization term to prevent the framework favoring subreddits that belong to multiple topics, and chosen as $\varphi_i = 1/|\hat{\mathcal{K}}_i|$ where $\hat{\mathcal{K}}_i$ represent the set of topics the subreddit i belongs to.

Theorem 5.2.1 *Given two functions $\mathcal{F} : 2^V \rightarrow \mathbb{R}$ and $f : \mathbb{R} \rightarrow \mathbb{R}$, the composition $\mathcal{F}' = f \circ \mathcal{F} : 2^V \rightarrow \mathbb{R}$ is non-decreasing submodular, if f is non-decreasing concave and \mathcal{F} is non-decreasing submodular. [69]*

Claim 5.2.2 *The topic-based coverage function in Equation (5.6) is submodular.*

Proof $\text{cover}_{\mathcal{S}}(i)$ is a modular function with non-negative weights (hence, monotone). Similarly, sum of $\text{cover}_{\mathcal{S}}(i)$ function is also monotone. This monotone function is surrounded by a square root function, which is a non-decreasing concave function. Using a concave function to this monotone function yields a submodular function (see Theorem 5.2.1). Finally, the sum of a collection of submodular functions are submodular [72], thus $F(\mathcal{S})$ is submodular. ■

Equation (5.6) satisfies two important aspects we would like to emphasize. First, it respects the similarity between subreddits. In other words, subreddits that belong to the same topic get diminishing returns. This is an important feature since we do not want to overwhelm the frontpage by covering posts from similar topics. On the other hand, by using the notion of topics, we are able to consider subreddits that the user is not even subscribed. Thus, this function naturally encourages discoverability of new subreddits.

Note that in both Equation (5.5) and Equation (5.6), we used *hot-score* to assign a reward to the posts. This function is not an ideal scoring function that we desire, since it causes a strong power-law behavior between scores of popular and un-popular subreddits. A more sophisticated scoring algorithm can be designed by advancing probabilistic models such as Chinese Restaurant Processes [73]. However, as of 2014⁶, Reddit no longer shares the actual upvote and downvote counts, and only provides the computed hot-score value for individual posts. Therefore, we use hot-score to associate a reward to the posts.

⁶https://www.reddit.com/r/announcements/comments/28hjga/reddit_changes_individual_updown_vote_counts_no.

5.2.4 Personalization

In Equation (5.6), we introduced a topic-based coverage function that captures the desired properties of our framework. However, this function does not respect the personal preferences of the users. This is an important problem since users might be interested in certain topics more than others. Moreover, even if a user is interested in a particular topic, it might be the case that the user is interested in certain subreddits in that topic more than others. To address these problems, we first introduce a personalized weight for each topic in Section 5.2.4, and then extend this model by introducing a personalized weight for subreddits in a given topic in Section 5.2.4.

Personalizing topic weights To address the first problem, we now introduce a personalized weight for each topic and obtain a *personalized coverage function* for a fixed user u , as follows:

$$F_u(\mathcal{S}) = \sum_{k \in \mathcal{K}} w_k \sqrt{\sum_{i \in \mathcal{R}_k \cap \mathcal{I}} \varphi_i \text{cover}_{\mathcal{S}}(i)}, \quad (5.7)$$

where w_k represents the preference of the user on topic k . Our main goal is to learn a personalized coverage function $F_u(\mathcal{S})$ by learning the weight vector of the user for all topics. Given a topic k , the *marginal gain* of adding a new post v to set \mathcal{S} can be computed as follows,

$$\Delta F_u(v|\mathcal{S}) := F_u(\mathcal{S} \cup \{v\}) - F_u(\mathcal{S}), \quad (5.8)$$

which corresponds to,

$$\Delta F_u(v|\mathcal{S}) := \sqrt{\sum_{i \in \mathcal{R}_k \cap \mathcal{I}} \varphi_i \text{cover}_{\mathcal{S} \cup \{v\}}(i)} - \sqrt{\sum_{i \in \mathcal{R}_k \cap \mathcal{I}} \varphi_i \text{cover}_{\mathcal{S}}(i)}. \quad (5.9)$$

Our user-interaction model considers the current interface of Reddit, that is, the users interact with the posts by means of *upvotes* and *downvotes*. We adapt a similar scheme and present the set of posts \mathcal{S} to the user, sorted by submodular reward of each post in decreasing order. Then, for each post j in set \mathcal{S} , we receive a feedback of $f_j \in \{-1, 0, +1\}$, corresponding to *upvote*, *no-vote*, and *downvote*, respectively. After presenting a set of

posts \mathcal{S} to the user with weight vector w and getting feedback vector f , the total loss we get can be written as follows,

$$L_u(\mathcal{S}, w, f) = - \sum_{k \in \mathcal{K}} w_k \sum_{j \in \mathcal{S}} f_j \Delta F_u(j|\mathcal{S}). \quad (5.10)$$

Note that we included a negative sign in the formulation since we are considering the *loss* instead of a reward. The total loss we get by using the current weight vector w depends on the marginal gain of each post in set \mathcal{S} , and the type of the feedback we receive for that post.

Due to the dynamic setting of the problem, a natural way to learn the weights w is to use an online learning algorithm such as Stochastic Gradient Descent (SGD) [74] or Exponentiated Gradient Descent (EG) algorithm [75]. However, SGD is not a suitable algorithm for our application since (i) weights might become negative and hence, violate the submodular property, (ii) the weights do not adapt to the interest of the users quickly enough. On the other hand, EG preserves the non-negativeness property while adjusting the weights in a shorter time [76]. More formally, given an old weight vector w^t at time t , EG estimates the new vector w^{t+1} at time $t+1$ by minimizing a combination of KL-divergence [77] between the new and old weight vector, and the loss of the new weights,

$$\sum_{k \in \mathcal{K}} w_k^{t+1} \log \frac{w_k^{t+1}}{w_k^t} + \eta L_u(\mathcal{S}, w_k^{t+1}, f) \quad (5.11)$$

where η is the learning rate, balancing the two terms. By taking the gradient of Equation (5.11), we get the following update rule for the weights w_{t+1} ,

$$w_k^{t+1} = \frac{1}{Z_w} w_k^t \exp \left(\eta \sum_{j \in \mathcal{S}} f_j \Delta F_u(j|\mathcal{S}) \right) \quad (5.12)$$

where Z_w is the normalization factor that ensures that the new weights sum to one. Next, we address the second problem, and learn a personalized weight for individual subreddits associated with a topic.

Personalizing subreddit-topic weights By learning the weight vector w , we aim to learn a *personalized coverage function* for the user. However, learning only the topic weights is

not enough to achieve the desired notion of coverage. It might be the case that the user is interested in a given topic, however might not be interested in some of the subreddits that are presented in that topic. For instance, *make-up* topic includes `MakeupAddiction`, `makeupexchange` and `AustralianMakeup` subreddits among others. Even though the user is interested in the *make-up* topic, it might be the case that user is not interested in `AustralianMakeup` subreddit simply because it targets users in Australia. Or similarly, the user might like certain subreddits in a given topic more than others. Therefore, using pre-defined weights on the subreddits within a topic is not ideal, and we need to learn the personal preference of the user for individual subreddits in a given topic. Thus, we extend the personalized coverage function in Equation (5.7) by introducing topic-subreddit weights α_{ki} for a given topic k and subreddit i as follows,

$$\hat{F}_u(\mathcal{S}) = \sum_{k \in \mathcal{K}} w_k \sqrt{\sum_{i \in \mathcal{R}_k \cap \mathcal{I}} \alpha_{ki} \varphi_i \text{cover}_{\mathcal{S}}(i)}. \quad (5.13)$$

where $\sum_i \alpha_{ki} = 1$. Note that we have a similar loss function as in Equation (5.10), where the marginal gain of adding a new post v now includes the α_{ki} terms as follows,

$$\Delta \hat{F}_u(v|\mathcal{S}) := \sqrt{\sum_{i \in \mathcal{R}_k \cap \mathcal{I}} \alpha_{ki} \varphi_i \text{cover}_{\mathcal{S} \cup \{v\}}(i)} - \sqrt{\sum_{i \in \mathcal{R}_k \cap \mathcal{I}} \alpha_{ki} \varphi_i \text{cover}_{\mathcal{S}}(i)}. \quad (5.14)$$

and the new loss becomes,

$$\hat{L}_u(\mathcal{S}, w_k, \alpha, f) = - \sum_{k \in \mathcal{K}} w_k \sum_{j \in \mathcal{S}} f_j \Delta \hat{F}_u(j|\mathcal{S}). \quad (5.15)$$

Similar to Equation (5.11), we would like to minimize the KL-divergence between new and old weight vectors and a loss term,

$$\sum_{k \in \mathcal{K}} w_k^{t+1} \log \frac{w_k^{t+1}}{w_k^t} + \sum_{k \in \mathcal{K}} w_k^{t+1} \sum_{i \in \mathcal{R}_k} \alpha_{ki}^{t+1} \log \frac{\alpha_{ki}^{t+1}}{\alpha_{ki}^t} + \eta \hat{L}_u(\mathcal{S}, w_k^{t+1}, \alpha_{ki}^{t+1}, f).$$

Setting the derivatives of this function with respect to w_k^{t+1} and α_{ki} , we get the following updates:

$$w_k^{t+1} = \frac{1}{Z_w} w_k^t \exp \left(\eta \sum_{j \in \mathcal{S}} f_j \Delta \hat{F}_u(j|\mathcal{S}) \right), \quad (5.16)$$

$$\alpha_{ki}^{t+1} = \frac{1}{Z_{\alpha_k}} \alpha_{ki}^t \exp \left(\eta \sum_{j \in \mathcal{S}} f_j \frac{\partial}{\partial \alpha_{ki}^t} (\Delta \hat{F}_u(j|\mathcal{S})) \right) \quad (5.17)$$

where in Equation (5.17) we use the gradient evaluated at the current parameter α_{ki}^t , thus, the updates are explicit. Note that the subreddit-topic weights α reside inside of the concave function, and subject to diminishing returns. This is a desirable property, since we want the reward of covering posts from a given subreddit to diminish as we select more posts from this subreddit.

The blessing of multiplicative updates is their speed, in the sense that they converge quickly to the *best* expert (i.e. topic) when the data generating process is static [76]. That is, the good weights grow exponentially, and the remaining weights rapidly get wiped out. However, this property becomes a caveat when the data generating process *changes* over time since the previously selected expert might not perform well on the new data. This phenomenon is often referred as the *curse* of the multiplicative updates [76]. This problem is particularly apparent in our application since the interest of the user might shift over time due to the dynamic nature of Reddit. For instance, the user might express a keen interest towards `nfl` and related subreddits due to the recent hype about the *Super Bowl* championship. This causes our framework to adjust the weight of the *sports* topic and promote *sports-related* subreddits and posts. However, even though we want to adapt to the new topic, it might not be reasonable to allow the weight of that topic to take over completely. In other words, instead of concentrating on one expert, we would like to have more *variety* in the long term. Note that the same observation also holds for the weights of the subreddits for a given topic. Decaying Past [78] is an elegant solution to make EG algorithm robust against data that changes over time by preventing the *currently* good weights from taking over completely. In an essence, this algorithm produces a new weight vector by taking a mixture of the past observations with weights *decaying backward* in time. Thus, the current observation always has the largest mixture coefficient. Let w^{t+1} represent the weight vector that we obtained from EG algorithm in Equation (5.12). Decaying Past algorithm transforms the weight of topic k as follows,

$$w_k^{t+1} \leftarrow (1 - \beta) w_k^{t+1} + \beta \frac{1}{Z_t} \sum_{q=0}^t \frac{w_k^q}{(t+1-q)^\gamma} \quad (5.18)$$

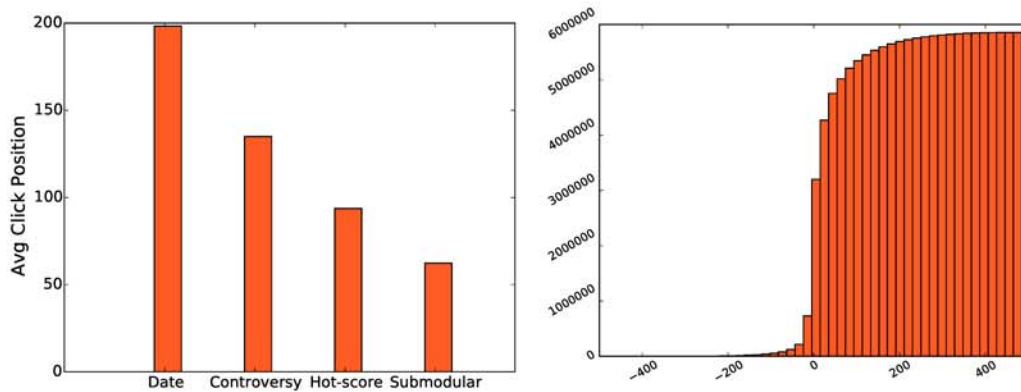


Figure 5.6. Average click positions out of all users for different baselines and our algorithm where lower is better (left). The histogram where x -axis represents the difference of average click position and our algorithm and y -axis represents number of links where higher is better (right).

where $\beta \in [0, 1]$ is the mixing proportion, $\gamma \geq 0$ is a parameter that determines the rate of decay, and $Z_t = \sum_{k \in \mathcal{K}} \sum_{0 \leq q < t} \frac{w_k^q}{(t+1-q)^\gamma}$ is a normalizing constant.

5.3 Related Work

The closest work to the proposed framework are [68] and [79]. Superficially, our framework shares several similarities with [68] and [79]. We also use a submodular function to select a small subset of items, and learn a profile vector from user interactions. However, there are several subtle but key differences, which we wish to highlight. First, the approach of [68] and [79] is text based, and the user profile vector is constructed using a bag of words representation of the documents a user has upvoted. [68] discuss an extension of their framework to topics. However, their approach requires a topic model to predict the latent topics of each post. Given the large number of posts, and the fact that they are usually short (typically less than 300 characters) it is unrealistic to run a topic model to infer topics per post on Reddit. Instead, we side step the issue by estimating the topics of a subreddit, and use the inferred latent subreddit relationships to estimate topics of a post. Note that this is done during the pre-processing step and does not incur a runtime overhead. More-

over, our submodular function is different that that proposed by [68] and [79] (see Section 5.2.4 for details). Another subtle but important difference is that [68] use Exponentiated Gradient (EG) of [75] to update the user profile weights, while [79] use stochastic gradient descent. As we argue in Section 5.2.4, both these algorithms are unable to capture drift in user interests. Therefore, we update the personalized weights of users with Decaying Past algorithm of [78]. Consequently, our framework is able to adapt the changes in user’s interests (see Section 5.4).

Despite being one of the most popular websites in the world, there has been few studies that focus on Reddit data. To best of our knowledge, this work is among the first to investigate relationships between communities and use it to design a recommendation system for subreddits. [80] studied how factors such as submission titles and submission times determine the popularity of social media content by studying re-submissions. [81] studied how user submissions have evolved and how the community’s perception of submissions changed over time. [82] studied comment threads on Reddit by investigating what extent discussion threads resemble a topical hierarchy and whether threads can be used to enhance Web search. [83] studied the problem of subreddit classification using graph kernels.

There has been also external websites such as `subredditfinder.com` to recommend subreddits. They provide a searchable database of subreddits based on activity, popularity or keywords. However, they do not curate a frontpage, nor do they perform any sort of personalization.

5.4 Experiments

We evaluate our framework with three different approaches. First, we perform an offline experiment using real user data from Reddit, and evaluate our framework in terms of *average click position* (see Section 5.4.1). We then perform two kinds of qualitative experiments; we simulate a hypothetical user, and investigate the quality of the recommendations (see Section 5.4.2). After that, we perform a user study and evaluate our framework in terms of *interestingness*, *diversity*, *discoverability*, and *personalization* (see Section 5.4.2).

5.4.1 Quantitative Experiments

In this section, we perform an offline experiment using data⁷ of actual Reddit users, who chose to share their information for research purposes between 2007-2010. The dataset is composed of unique Reddit id of the users, and corresponding links they upvoted or downvoted. Additionally, we used the public API⁸ of Reddit and collected *subreddit*, *creation date*, *hot-score value*, and *number of comments* for each link since such metadata was not included in the voting dataset. After removing deleted posts, the dataset includes 2,046,401 links from 31,452 users with 5,856,725 total number of votes. This dataset covers a diverse set of communities, having 6,497 different subreddits.

We evaluate our framework in terms of *average click-position*. Given a user u who casts a vote against link l , and a frontpage $f_{\mathcal{M}}$ generated by a method \mathcal{M} , the average click-position is simply defined as the average position of link l in $f_{\mathcal{M}}$. Thus, an ideal method should place the links that user likes to an upper position in the frontpage. Since we do not have access to the list of subreddits that the user has subscribed to, we split the data into training and testing (90% – 10%) and created a *user profile* using the training set. After that, we simply assume that a user is subscribed to a subreddit if there are at least five votes for this subreddit in the training set. Then, for each date the user casts a vote in the test set, we generate a frontpage of 500 posts from the subreddits that user is interested in. This information is then used to compute the average click position. Since we do not have access to a live environment where we learn personalized topic weights through the interactions the user has expressed, we only evaluate the core framework introduced in Section 5.2.2. We compare our method, *Submodular*, against four baselines, as follows:

- *Date*: In this method, we sort links by date in decreasing order, thus, users are always exposed to the *freshes*t content.

⁷Dataset can be accessed at: http://www.reddit.com/r/redditdev/comments/bubhl/csv_dump_of_reddit_voting_data.

⁸Public API of Reddit: <https://www.reddit.com/dev/api>.



Figure 5.7. Initial coverage of subreddits by hot-score algorithm for the simulated user (left). Initial coverage of subreddits by our framework (center). Coverage of subreddits after five epochs (right).

- *Controversy*: This is an alternate method that Reddit provides for its users⁹, and sorts links by taking a ratio between the total number of votes and the difference between upvotes and downvotes. Thus, a controversial post is defined as having a relatively equal number of upvotes and downvotes.
- *Hot-score*: This method simply uses the hot-score function in Equation (5.1), and sorts the set of links for each subreddit in decreasing order. It then generates the frontpage by selecting one link from each subreddit in a *round-robin fashion*, until the frontpage is filled.

Figure 5.6 (left) illustrates the average click positions for all methods. As can be seen from the figure, our method has a significantly better average click position than baseline methods, meaning that we are able to put the links that user likes to an upper position in the frontpage. Figure 5.6 (right) compares hot-score based ranking to our method. In this case, the x-axis represents the difference of average click position between hot-score and our algorithm. As can be seen from the figure, our method significantly outperforms the hot-score method. Moreover, our framework is not prohibitively expensive, and takes only an additional 0.25 seconds per frontpage generation comparing to baselines.

5.4.2 Qualitative Experiments

In the previous experiment, we quantitatively demonstrated our algorithm’s ability to rank posts in an upper position. However, this measure does not reflect how well our algorithm performs in terms of other important aspects such as *diversity*, *discoverability* and *personality*. In this section, we first demonstrate the quality of our framework from a simulated user’s point of view, and then perform a user study to test our framework.

In order to build subreddit similarities, we collected data using public API of Reddit between November 2014-2015, having a total of 6.2 million links from 225,785 subreddits. We removed all subreddits and links that are marked as NSFW (i.e. adult content) and removed subreddits that are having less than 1000 subscribed users. After this filtering

⁹<https://www.reddit.com/controversial>.

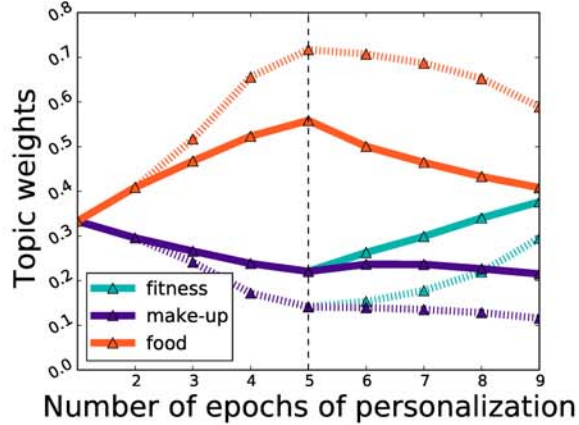


Figure 5.8. Topic weights for the simulated user over ten epochs where dashed-lines represent weights learned by EG, and non-dashed lines represent weights learned by Decaying Past algorithm.

step, our dataset has 4.2 million links from 4,100 subreddits. For each post, we removed stopwords and punctuation. After that, we build a corpus of 4,100 documents where each document contains the titles of all posts submitted to a given subreddit. We then build a topic-model using LDA with 300 number of topics, and we selected the number of subreddits in each topic as $m = 5$. We used Online LDA [84], an improvement over LDA which uses variational inference instead of a Collapsed Gibbs Sampler [85], and used default LDA parameters provided by the tool. Under these settings, LDA takes two hours to produce a topic-model. Finally, for our personalization component, we used a step size of $\eta = 0.5$ for EG, $\beta = 0.5$ and $\gamma = 1$ for Decaying Past updates.

Simulated user experiment In this experiment, we consider a hypothetical user who is interested in *food*, *make-up* and *fitness* topics. In particular, we assume that the user is subscribed to *loseit*, *Art*, *MakeupAddiction*, *aww*, *Cooking*, *pics*, *food* and *Fitness* subreddits. In order to evaluate the quality of our framework, we first want to show that it respects *diversity*, in the sense that it does not promote redundant content. For instance, note the similarity between *Cooking* and *food* subreddits, as well as *loseit* and *Fitness*. We hypothesize since these subreddits contain similar

content, the proportion of them should be smaller than subreddits with distinct topics, such as `MakeupAddiction`.

Figure 5.7 (left) illustrates the covered subreddits by the hot-score algorithm. Note that, since links are selected from each subreddit in a round-robin fashion, all subreddits are covered equally. However, this method does not consider the similarity between the subreddits, thus, it might repeatably cover posts from `food` and `Cooking` subreddits and overwhelm the user with similar content. Figure 5.7 (center) shows the coverage of subreddits from our framework. As we hypothesized, our framework demonstrates two important aspects. First, the coverage of similar subreddits diminish due to submodularity. On the other hand, in addition to the subreddits which the user is subscribed to, we also show posts from related subreddits, such as, `keto`, `painting`, `AnimalsBeingJerks` and `makeupexchange`. Therefore, our framework promotes discoverability.

Secondly, we would like to simulate the effect of personalization on this hypothetical user. In particular, we assume that a set of posts shown to the user every day for a period of ten days. In the first five days, the user only upvotes posts related to the *make-up* topic. We hypothesize that as user keeps expressing an interest towards this topic, the weight associated with the *make-up* topic will increase accordingly. Figure 5.7 (right) illustrates the coverage of the user after five epochs. As can be seen from the figure, the coverage of *make-up* related topics is higher than others. Moreover, given the intense interest of the user towards this topic, we inject more links related to *make-up* from subreddits user is not subscribed to, in particular, from `makeupexchange`, `MakeupAddicts`, `RedditLaqueristas`.

Finally, we would like to see how our system is able to adapt to the changes. Therefore, we assume that user has developed a sudden interest towards *fitness* topic for the remaining five epochs, and only upvotes posts related to this topic. Figure 5.8 illustrates how the weights change over time (updates by our algorithm denoted by non-dashed lines). Note that for the sake of brevity, we only illustrate the weight of three topics, *food*, *make-up*, and *fitness*. As can be seen from the figure, the weight of *make-up* topic grows over the first five epochs, indicating that the coverage of this particular topic will increase. On the other hand, during the next five days, the weight of *fitness* topic starts to overtake. Figure 5.8 also

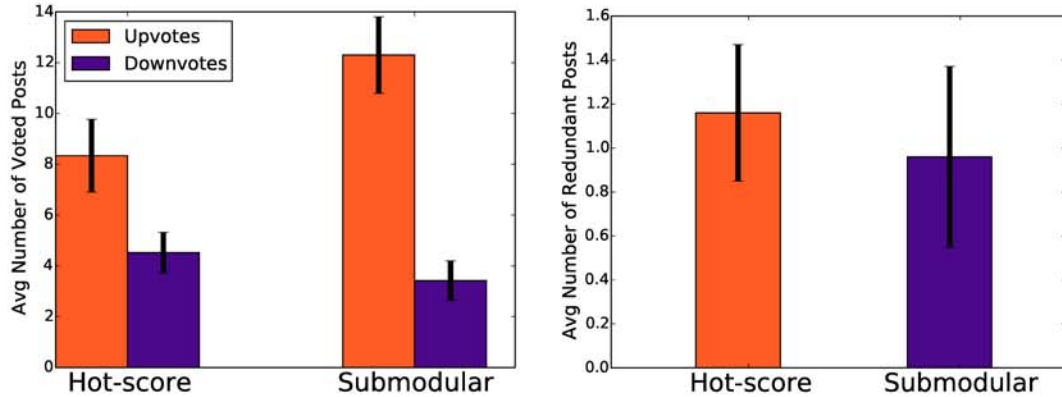


Figure 5.9. Results from user study. Interestingness, measured by average number of upvotes and downvotes (left). Diversity, measured by number of redundant posts (right).

illustrates the weights learned by EG updates (dashed-lines). In comparison to Decaying Past algorithm, we can see that EG quickly adapts to the first interest of the user during first five epochs, however, does not *recover* from this behavior after user shifts to the *fitness* topic.

User Study In this section, we show the evaluation of our framework with a user-study that consist of 13 volunteers from University of California, Santa Cruz. In particular, we would like to evaluate our framework in terms of *interestingness*, *diversity*, *discoverability* and *personality*. For this purpose, we designed a website that has a similar interface to Reddit. Users are first shown a list of subreddits, associated with a short description, and asked to subscribe for subreddits related to their interests. In order to not to overwhelm users, we only provided a list of top 125 subreddits, and asked users to subscribe at a maximum of five subreddits. We showed users the generated frontpages consisting of 25 posts, which is the default frontpage size of Reddit.

We showed users a set of posts over a period of five epochs and asked them to upvote or downvote links they were interested in. Users were shown the title and subreddit of each post, and had the ability to investigate the link in a pop-up window by clicking on each

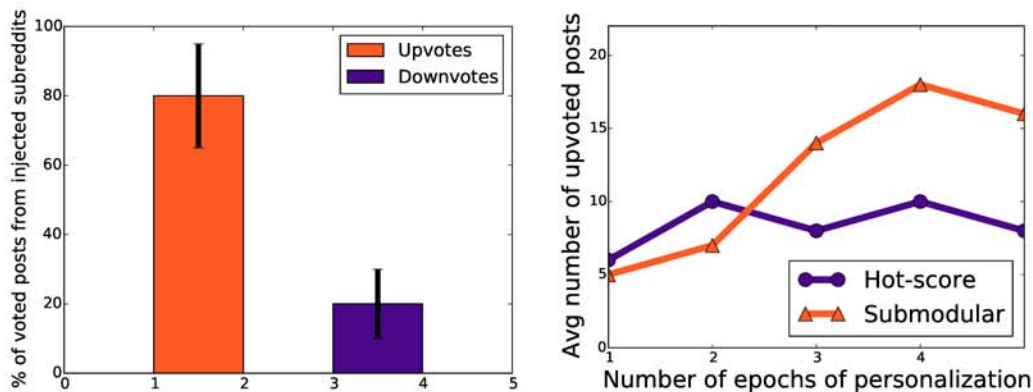


Figure 5.10. Results from user study. Discoverability, measured by the percentage of upvoted links from injected subreddits (left). Personalization, measured by number of upvoted links over time (right).

link. In addition to upvote and downvote buttons, we also provided a separate checkbox associated with each link which allows users to mark a content as *redundant*. Users were shown the frontpage that is generated by our method, or the hot-score method in random order, and they were not made aware of the source of the posts in order to not to bias their ratings. Under this setting, the generation of frontpage takes 1 second on average.

Figure 5.9 (left) illustrates the average number of upvotes and downvotes that each of the underlying method gets over five epochs from all users. As we can see from the figure, users upvoted more links in our framework. Figure 5.9 (right) illustrates the average number of redundant links found by both methods. Note that the results for hot-score and our method are similar, and users did not find many redundant content in shown frontpages. We conjecture that this is due to the fact that the version of hot-score algorithm we use in our comparison generates links in a round-robin fashion, thus shows content equally distributed among all subreddits user is subscribed to. However, this assumption does not hold when users are subscribed to multiple subreddits covering similar topics. As our third experiment, we investigated whether users liked the links that we promote from subreddits that they are not subscribed to. We simply computed the percentage of upvotes and downvotes for each inserted link. As can be seen from the Figure 5.10 (left), users

are interested in injected links 80% of the time. Therefore, our framework encourages discoverability. For our final experiment, we evaluated how well our algorithm learns under limited user feedback. Figure 5.10 (right) illustrates the average number of upvotes over five epochs and we can see that we are able to learn the personal preferences of the users since the number of upvotes increase over time. On the other hand, since hot-score method is not performing any personalization, the average number of liked posts are uniformly distributed.

5.5 Conclusions

In this chapter, we proposed a framework to tailor a *personalized* set of posts that takes *coverage*, *diversity* and *discoverability* into account. We formulated a simple and elegant submodular objective function for which we can efficiently provide a near-optimal solution. We then projected subreddits into topical space, and discovered novel relationships between communities. We then used this relationship to encourage discoverability of new content and communities. Finally, we extended our framework to learn a *two-step personalized coverage function*.

We evaluated our framework both quantitatively and qualitatively. Our offline evaluation with real user data from Reddit demonstrates that our framework ranks posts to an upper position, and significantly outperforms baseline algorithms. Moreover, our user-study shows that our that our algorithm provides better coverage and diversity while encouraging discoverability.

Finally, even though we demonstrated Reddit as a use-case, our framework is applicable to any setting where the items are distributed into *categories* (i.e. cuisines on Yelp), and the goal is to select a representative and diverse subset of *items* (i.e. restaurants on Yelp). Moreover, in cases where inferring to similarity between categories is not possible by building a topic-model, one can use *collaborative filtering* approaches to infer category similarities.

6 A SUBMODULAR FRAMEWORK TO SUMMARIZE TED TALKS

In this chapter, we propose a novel summarization framework to summarize TED talks. We formulate our objective function as a submodular framework that balances coverage and diversity. Our coverage function covers the ideas that speaker is promoting while also leveraging aspects that audience is focused on. Our diversity function measures the diversity of the summary in terms of latent dimensions, and incorporates state-of-the-art word embedding methods from deep learning and neural language models. Our experiments show that both our coverage function, and the diversity function outperforms the baseline methods. Moreover, our method is applicable to any setting where a feedback from the audience is available, such as videos on Youtube, or lectures in Coursera.

6.1 Motivation

TED is a non-profit organization devoted to “*Ideas Worth Spreading*”¹. Since 1984, TED invites “the world’s most fascinating thinkers and doers to give the talk of their lives”, and brings inspirational speakers from **T**echnology, **E**ntertainment, **D**esign areas together. As of today, more than 1,500 talks are available on TED.com, meaning over 1,500 ideas worth spreading. However, watching all videos on TED.com, and extracting useful ideas is prohibitively expensive. In fact, according to TED speaker Sebastian Wernicke’s ‘1000 TED Talks, 6 words’ talk, it would take over 250 hours to watch 1,000 TED talks which brings a total of \$15,000 cost per person².

In this work, we propose a novel framework to summarize TED talks. Unlike traditional document summarization tasks, an important feature of TED talks is the notion of an *audience*. TED.com makes the best talks and performances available by posting video

¹ TED Talks: <http://www.ted.com/pages/view/id/5>

²“1000 TED Talks, 6 words”: https://www.ted.com/talks/sebastian_wernicke_1000_tedtalks_6_words

recordings of the talks along with their subtitles. The users then can leave comments for individual talks, and criticize the points they agree or disagree. For instance, consider Ken Robinson’s infamous “*Do schools kill creativity?*” talk, where the description is given as follows³:

“Sir Ken Robinson makes an entertaining and profoundly moving case for creating an education system that nurtures (rather than undermines) creativity.”

Figure 6.1 illustrates the word cloud for the speaker where the most dominant words are **education**, **people** and **children**. Similarly, Figure 6.2 illustrates the word cloud for the audience which focuses on words such as **creativity**, **school**, **education**. One can see that even though there are common concepts such as *education* where both the speaker and the audience focus on, there are also concepts such as **creativity** where audience is more interested in. In this work, we conjecture that an ideal summarization framework should not only cover the diverse points the speaker is focuses on, but should also leverage the audience’s point of view.

Our second contribution in this work is to measure the diversity of the summary in terms of *latent dimensions*. Our framework uses the latest advancements in deep learning and neural language models, and incorporates the latent representations of the words to encourage diversity.

This chapter is structured as follows. In Section 6.3 we discuss related work. In Section 6.2, we propose the coverage and the diversity components of our framework. In Section 6.4 we compare our framework with several baseline methods. Section 6.5 concludes the chapter.

6.2 Methodology

In this study, we are interested in designing a system to perform extractive summarization on TED talks. In particular, we are interested in selecting a subset of sentences from

³“Do schools kill creativity?”: https://www.ted.com/talks/ken_robinson_says_schools_kill_creativity



Figure 6.1. Top words of speaker where words sizes are positively correlated with frequency of each word, and colors are assigned randomly. We can see that the speaker focuses on words such as **education, people, children**.



Figure 6.2. Top words of audience where words sizes are positively correlated with frequency of each word, and colors are assigned randomly. We can see that the audience focuses on words such as **education, school, creativity**.

the talk that covers important aspects the speaker is promoting, while also leveraging concepts that audience is focused on. Formally, let \mathcal{T} represent a TED talk that is divided into n units, and let $i \in \mathcal{T}$ represent a single unit in \mathcal{T} . Then, we are interested in selecting a subset $\mathcal{S} \subseteq \mathcal{T}$ to represent the entirety of ground set \mathcal{T} , subject to a cardinality constraint $k \leq n$. In other words, we would like to select k units from the ground set \mathcal{T} to form a summary. Let us assume that we have a set function $\mathcal{F} : 2^{\mathcal{T}}$ that measures the quality of set \mathcal{S} . Then, we are interested in finding the optimal solution to the following optimization task,

$$\mathcal{S}^* = \operatorname{argmax}_{\mathcal{S} \subseteq \mathcal{T} : |\mathcal{S}| \leq k} \mathcal{F}(\mathcal{S})$$

where \mathcal{S}^* represents the optimal solution subject to a cardinality constraint k which denotes the total number of units in the summary set \mathcal{S} . Thus, the summarization task we are interested in becomes a *budgeted maximum coverage* problem [86]. This is a well-known NP-hard problem [67], however, it has been shown that if \mathcal{F} is monotone submodular, then a greedy algorithm can solve the problem near-optimally with $(1 - 1/e)$ approximation factor [67]. In addition to providing an efficient solution with a constant factor guarantee, submodular algorithms are shown to naturally model *coverage* and *diversity* notions [69, 87]. Several existing frameworks for automatic summarization are instances of submodular functions [69], and they achieve state-of-the-art results for many summarization problems. Therefore, we formulate our problem as a submodular task for which we can efficiently find the optimal solution within $(1 - 1/e)$ by using a greedy algorithm.

An important aspect of a good summarization system is to balance *coverage* and *diversity*. Maximal Marginal Relevance (MMR) [88] is one of the most popular summarization frameworks that positively reward coverage while negatively penalizing the redundancy in the summary set \mathcal{S} . However, even though this function is submodular, the negativity factor violates monotonicity and constant-factor approximation of the greedy algorithm is not guaranteed [69]. Instead, [69] proposes a framework that positively reward diversity instead of negatively penalizing redundancy. In particular, their framework is defined as follows,

$$\mathcal{F}(\mathcal{S}) = \mathcal{F}_L(\mathcal{S}) + \lambda \mathcal{F}_R(\mathcal{S}),$$

where $\mathcal{F}_L(\mathcal{S})$ measures the *coverage* (i.e. how similar \mathcal{S} to the rest of the document), and $\mathcal{F}_R(\mathcal{S})$ measures the *diversity* (i.e. how diverse the sentences in set \mathcal{S}). The $\mathcal{F}_R(\mathcal{S})$ term can be interpreted as a regularization term that exists in traditional machine learning frameworks. Next, we discuss how to design the relevancy, and the diversity components.

6.2.1 Designing the Relevancy Component

Most of the existing summarization frameworks including [69,87] operate on sentence-level summaries where one aims to select a subset of sentences from the document. However, unlike traditional text, the transcript of the talks are directly derived from speech. This brings unique challenges when selecting individual sentences as the single units in the summarization task since sentences are often connected to each other via conjunctions. For instance, the following set of sentences from Ken Robinson’s “*Do schools kill creativity?*” talk should be considered as one unit since they are topically *coherent*:

“What TED celebrates is the gift of the human imagination. We have to be careful now that we use this gift wisely and that we avert some of the scenarios that we’ve talked about. And the only way we’ll do it is by seeing our creative capacities for the richness they are and seeing our children for the hope that they are. And our task is to educate their whole being, so they can face this future.”

As can be seen from this example, treating “*And the only way we’ll do it is by seeing our creative capacities for the richness they are and seeing our children for the hope that they are.*” sentence as a single unit will lower the quality of the summary since it is not clear what *it* refers to in the sentence. Therefore, instead of selecting individual sentences to add to the summary set \mathcal{S} , we treat the set of coherent sentences as a single unit. For this purpose, we use the segmentation that is manually generated by the TED.com staff.

An important aspect of the relevance component $\mathcal{F}_L(\mathcal{S})$ is to measure the similarity of the summary set \mathcal{S} to the overall talk. In particular, we would like to cover important aspects that speaker is focused on. For this purpose, we define the $\mathcal{F}_L(\mathcal{S})$ as follows,

$$\mathcal{F}_L(\mathcal{S}) = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} \langle \mathbf{w}_i, \mathbf{w}_j \rangle_{\alpha}. \quad (6.1)$$

Here, i represents an arbitrary unit from the summary set \mathcal{S} , j represents an arbitrary unit from the ground set \mathcal{T} , \mathbf{w}_i (and similarly, \mathbf{w}_j) is a vector of size $|\mathcal{V}|$ where \mathcal{V} is a vocabulary that is constructed as the set of all unique words in talk \mathcal{T} , and each element $w \in \mathbf{w}_i$ simply represents the number of times that word w appears in unit i . $\langle \cdot, \cdot \rangle$ represents a dot product, and α represents the weight vector of the speaker over the vocabulary \mathcal{V} that is simply generated by counting the frequency of each word $w \in \mathcal{V}$ over the entire transcript.

$\mathcal{F}_L(\mathcal{S})$ measures the similarity of the summary set \mathcal{S} to the talk \mathcal{T} . However, it only captures the aspects that the *speaker* is focused on. Ideally, we would also like to promote the aspects that the *audience* is interested in. Therefore, we define another coverage function similar to $\mathcal{F}_L(\mathcal{S})$,

$$\mathcal{F}_A(\mathcal{S}) = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} \langle \mathbf{w}_i, \mathbf{w}_j \rangle_{\beta}, \quad (6.2)$$

where β represents the weight vector of the audience over the vocabulary \mathcal{V} that is generated by counting the frequency of each word $w \in \mathcal{V}$ in the comments of the audience for the talk \mathcal{T} . However, both $\mathcal{F}_L(\mathcal{S})$ and $\mathcal{F}_A(\mathcal{S})$ are monotone functions that do not encourage diminishing returns. Thus, we need to design another component that will encourage the objective function to encourage diversity in the summary set \mathcal{S} .

6.2.2 Designing the Diversity Component

Diversity component should be designed to reflect the notion of *diminishing returns*, such that selecting units that are similar to the already selected sentences will have less gain. $\mathcal{F}_R(\mathcal{S})$ function of [69] is defined as follows,

$$\mathcal{F}_R(\mathcal{S}) = \sum_{k=1}^{\mathcal{K}} \sqrt{\sum_{i \in \mathcal{P}_k \cap \mathcal{S}} r_i}, \quad (6.3)$$

where $\mathcal{P}_k, k = 1, \dots, \mathcal{K}$ is a partition of the ground set \mathcal{T} into separate clusters that is generated by using K -means algorithm on the sentences, and r_i is the reward of adding item i to the empty set \mathcal{S} . This function rewards diversity since selecting a unit from a cluster that is not yet having one of its elements already chosen has more benefit. Then, then the cluster start to demonstrate diminishing returns effect as we start to select more units from it due to the concave square root function. Thus, the partitions \mathcal{P} of the ground set \mathcal{T} is simply performed to group similar items together under a concave function so that we can apply the diminishing returns effect.

One important observation we propose in this chapter is that the diversity can be measured by means of *latent dimensions* on the vocabulary set \mathcal{V} . Thus, selecting units that represent dimensions that are not yet covered will have more gain. Word embeddings are popular methods to generate distributed vector representations of words where each word $w \in \mathcal{V}$ is represented as a vector over \mathcal{D} dimensions. Unlike traditional language models, word embeddings [37] take advantage of the notion of a *context* that is defined as a fixed number of preceding words. Thus, similar words are mapped to similar positions in the vector space. The learned word vectors are empirically shown to preserve semantics, for instance, word vectors can be used to answer analogy questions using simple vector algebra where the result of a vector calculation $\mathbf{v}(\text{“Madrid”}) - \mathbf{v}(\text{“Spain”}) + \mathbf{v}(\text{“France”})$ is closer to $\mathbf{v}(\text{“Paris”})$ than any other word vector [35].

Let us assume that we have a word embedding model \mathcal{M} over \mathcal{D} dimensions. Then, we can define a diversity component as follows,

$$\mathcal{F}_R(\mathcal{S}) = \sum_{d=1}^{\mathcal{D}} \sqrt{\sum_{i \in \mathcal{S}} \varphi_{i,d}}, \quad (6.4)$$

where d is an arbitrary dimension from \mathcal{D} , and $\varphi_{i,d}$ represents how much unit i covers the dimension $d \in \mathcal{D}$. Thus, we measure the diversity in terms of latent dimensions that words are embedded in. Since the word representations do not depend on the dataset, this approach allows us to plug high quality word representations that are trained on external sources, such as Google’s pre-trained word vectors on Google News dataset that consists of 100 billion words⁴. Note that this approach does not require us to explicitly cluster sentences in a document as in [69], and thus, provides an alternative way to measure diversity even when the documents are too short and do not carry enough signals for clustering. Thus, our final framework becomes the following,

$$\mathcal{F}(\mathcal{S}) = \mathcal{F}_L(\mathcal{S}) + \mathcal{F}_A(\mathcal{S}) + \lambda \mathcal{F}_R(\mathcal{S}), \quad (6.5)$$

where $\mathcal{F}_L(\mathcal{S})$ is defined as Equation (6.1), $\mathcal{F}_A(\mathcal{S})$ is defined as Equation (6.2), $\mathcal{F}_R(\mathcal{S})$ is defined as Equation (6.4) and λ is a weight between $0 \leq \lambda \leq 1$ that balances the relevancy and the diversity.

Claim 6.2.1 *The summarization algorithm in Equation (6.5) defined by Equation (6.1), Equation (6.2) and Equation (6.4) is submodular.*

Proof Equation (6.1) and Equation (6.2) are modular functions with non-negative weights (hence, monotone). Similarly, $\sum_{i \in \mathcal{S}} \varphi_{i,d}$ in Equation (6.4) is also monotone with non-negative weights. This monotone function is surrounded by a square root, which is a non-decreasing concave function. Applying a concave function to a monotone function yields a submodular function (see Theorem 5.2.1). The sum of a collection of submodular functions are submodular [72], thus $\mathcal{F}_R(\mathcal{S})$ is submodular. The summation of the modular functions $\mathcal{F}_L(\mathcal{S})$ and $\mathcal{F}_A(\mathcal{S})$ with submodular function $\mathcal{F}_R(\mathcal{S})$ preserves the submodular property, and hence $\mathcal{F}(\mathcal{S})$ is submodular. ■

⁴<https://code.google.com/archive/p/word2vec/>

6.3 Related Work

Automatic summarization is a well-studied problem in the literature. Several methods have been proposed for single and multi-document summarization, including [88] and [89]. There has been also extensive work in submodular summarization frameworks such as [69] and [87]. Our work is different from related works in the sense that our coverage function is based on both speaker and the audience, and our diversity function evaluates the diversity in terms of latent dimensions of the words.

There also has been several works that use TED data. [90] explores content-based and collaborative recommendation methods for TED talks. [91] introduces a model for multiple instance learning and uses the comments from TED talks to predict talk-level emotion dimensions. [92] studies TED talks to design a system that detects and enables the exploration of relevant fragments inside educational videos. [93] studies finding metadiscourse concepts in TED talks with a crowdsourced approach. [94] proposes a model to analyze memorable spoken quote corpora from TED talks. [95] proposes a system to create video digests to support browsing and skimming videos, and uses TED talks as one of the case studies. [96] analyzes comments of TED talks that are left on both TED.com and Youtube, and investigates whether there exists a significant difference in type of comments according to platform, and whether there exists significant differences in commenting observed according to presenter characteristics.

6.4 Experiments

Here, we first introduce our dataset in Section 6.4.1, then discuss how we evaluate our algorithms in Section 6.4.2. In Section 6.4.3, we compare our method with several baseline methods.

6.4.1 Datasets

We crawled TED.com website between February 24-25, 2016 and obtained raw HTML files of the talks, and top-level comments associated with each talk. We then parsed HTML files to extract metadata, and applied basic pre-processing steps such as stopword and punctuation removal. While extracting transcripts of the talks via *speech to text* methods is possible, TED.com provides robust transcripts for the talks along with the videos. Thus, we directly use the obtained transcripts from the collected HTML files. The collected TED talks cover a wide range of interests such as topics including *music*, *gaming*, *gender*, and themes including *How the Mind Works*, *Tales of Invention* and *Women Reshaping the World*. Figure 6.3 shows top topics (left), and top themes (right).

6.4.2 Evaluation

In order to evaluate the quality of the generated summaries, we use a benchmark evaluation metric called Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [97]. ROUGE is officially adopted as the evaluation metric for The Document Understanding Conference (DUC)⁵, the main forum that provides benchmarks for researchers on document summarization. This metric simply computes the recall between the automatically produced summary and a set of human-produced reference summaries. Let \mathcal{S} represent the candidate summary extracted from the ground set \mathcal{T} , and $c_i : 2^{\mathcal{T}} \rightarrow \mathbb{Z}_+$ represent the number of times n -gram i occurs in summary \mathcal{S} . Similarly, let \mathcal{R} represent a set of reference summaries where \mathcal{R}_j represent n -grams contained in the reference summary j , and $r_{j,i}$ represent number of times n -gram i occurs in reference summary j . Then, ROUGE score is defined as,

$$\mathcal{F}_{\text{rouge}}(\mathcal{S}) = \frac{\sum_{j=1}^{|\mathcal{R}|} \sum_{i \in \mathcal{R}_j} \min(c_i, r_{j,i})}{\sum_{i=1}^{|\mathcal{R}|} \sum_{i \in \mathcal{R}_j} r_{j,i}}.$$

Unlike traditional documents used in document summarization tasks, the transcript of the talks are often short documents. Therefore, we adapted ROUGE score with $n = 1$ for

⁵<http://duc.nist.org>



Figure 6.3. Top topics and themes out of 1,200 TED.com talks where words sizes are positively correlated with the frequency of each category.

n -grams. In other words, we measure the unigram recall between the automatic summary and the reference summaries. Thus, a high level of overlap between the shared concepts indicates a high quality of summary.

In order to evaluate ROUGE score, we need reference summaries. `tedsummaries.com` is a website that shares manually generated short summaries of TED talks. Thus, each reference summary collected from this website corresponds to a reference set \mathcal{R} when computing the ROUGE score.

6.4.3 Results

Our experiments are three-fold. First, we would like to understand how the speaker-based modular function in Equation (6.1) performs against a randomly generated summary. Then, we would like to understand how audience-based function in Equation (6.2) improves over the speaker-based function. Finally, we would like to see how results improve with the diversity component. In particular, we compare the diversity component of [69] in Equation (6.3) that is based on K -means, and our diversity component in Equation (6.4) that is based on word embeddings. In all experiments, we select $k = 5$ units to form a summary since the median number of segments for each talk is $n = 20$. Thus, we are selecting 25% of the talk to form a summary. More formally, the baseline methods we compare in this study is as follows:

1. **Random:** We randomly select k segments from the talk to form a summary.
2. **Speaker:** We use Equation (6.1) to select k segments from the talk to form a summary.
3. **Speaker + Audience:** We use Equation (6.1) and Equation (6.2) to select k segments from the talk to form a summary.
4. **Speaker + Audience + Diversity (K -means):** We used Equation (6.1) and Equation (6.2) as the relevancy component, and Equation (6.3) as the diversity component. Following [69], we selected the number of clusters for each talk $k = 0.2 \times n$ where

Table 6.1.

ROUGE measures on 80 TED talks for different methods. Numbers represent percentages.

| Method | ROUGE |
|---|--------------|
| Random | 24.69 |
| Speaker | 35.21 |
| Speaker + Audience | 36.29 |
| Speaker + Audience + Diversity (K-means) | 36.79 |
| Speaker + Audience + Diversity (Word2vec) | 38.23 |

n is the total number of segments in the talk. On median, the segments are clustered into $k = 4$ clusters. We used Scipy [98] library for K -means, and applied TF-IDF before feeding the segments into the clustering algorithm. Following the paper of [69], the reward r_i of adding a segment to the empty set is computed as $r_i = \frac{1}{n} \sum_j p_{i,j}$ where $p_{i,j}$ where $p_{i,j}$ is computed as follows,

$$p_{i,j} = \frac{\sum_{w \in i} \text{TF}_{w,i} \times \text{TF}_{w,j} \times \text{IDF}_w^2}{\sqrt{\sum_{w \in i} \text{TF}_{w,i}^2 \text{IDF}_w^2} \sqrt{\sum_{w \in j} \text{TF}_{w,j}^2 \text{IDF}_w^2}}.$$

5. **Speaker + Audience + Diversity (word embedding):** We used Equation (6.1) and Equation (6.2) as the relevancy component, and Equation (6.4) as the diversity component. We used Google’s pre-trained word vectors on Google News dataset that consists of 100 billion words⁶. This model is constructed using word2vec tool [37] with Negative Sampling method with Skip-gram representation on $d = 300$ latent dimensions. $\varphi_{i,d}$ in Equation (6.4) is simply defined as $\varphi_{i,d} = \sum_{w \in i} \mathcal{M}_{w,d}$ where \mathcal{M} corresponds to a word embedding model, and $\mathcal{M}_{w,d}$ corresponds to the weight of word w of the dimension d in model \mathcal{M} . Since the weights can be negative in word embedding models, we used a logistic function on weights where a weight $\mathcal{M}_{w,d}$ is transformed to $\mathcal{M}_{w,d} = \frac{1}{1+e^{-\mathcal{M}_{w,d}}}$.

⁶<https://code.google.com/archive/p/word2vec/>

Table 4 shows the obtained ROUGE statistics for different methods. As can be seen from the results, using a coverage function and selecting segments that covers the aspects the speaker is focused on performs significantly better than using a random model. Furthermore, using the audience weights in the coverage function improves the results. This is aligned with our intuition that considering the concepts the audience is engaged in addition to the speaker is important. Using the diversity component with K -means slightly improves the results. We conjecture this is due to the fact that TED talks are short comparing to traditional text in document summarization tasks, and clustering segments do not reflect our desired notion of diversity. Finally, we obtain the best results with the diversity component that is based on latent dimensions. This result is aligned with our intuition that evaluating diversity of the units in terms of latent dimensions performs well.

6.5 Conclusions

In this chapter, we proposed a novel summarization framework that balances coverage and diversity. Our coverage function not only takes the speaker’s aspect into account, but it also incorporates the important concepts that the audience is focused on. Our diversity component is not dataset independent, and measures the diversity in terms of latent dimensions of the words that forms the sentences. Our framework is flexible in the sense that any general or domain-specific word embedding model can be plugged to measure diversity. Moreover, this approach inherently allows us to incorporate external resources, such as pre-trained word embedding models that is trained on billions of documents. Finally, we demonstrate our framework to summarize TED talks as a novel application.

A similar setting to TED talks is to summarize Youtube videos by using the comments from the audience, summarizing movie scripts using IMDB comments, or summarizing lecture videos on Coursera using the questions and comments of the students in the discussion forum.

7 CROWDSOURCED RESOURCE-SIZING OF VIRTUAL APPLIANCES

In this chapter, we use a population of VMware Virtual Center Virtual Appliances (VCVA) and their respective workloads and describe techniques for constructing a model of their resource consumption by mining logs of application performance. We use our model to provide sizing recommendations for the virtual appliance and identify features that can be used to provide estimates of expected memory consumption. We show results of better than 70% prediction accuracy for predicting physical memory usage. We describe modeling techniques from statistical machine learning that are amenable to representing complex, non-linear systems.

7.1 Motivation

Virtual Appliances are pre-packaged virtual machine images that run on a specific virtualization platform, and they are common modalities of application deployment in public, private and hybrid cloud environments. These virtual machine images include the software components of the application along with meta data about their anticipated aggregate resource requirements such as amount of RAM and number of GHz desired for the virtual machine. Accurate estimates of resource requirements can influence the configured size such as number of virtual CPUs (vCPU), amount of RAM, and the settings of resource reservations or limits [99].

Specifying the appropriate resource size is a critical but challenging task. Allocating insufficient resources potentially impacts the performance, reliability and stability of the virtual appliance, while allocating too many resources is wasteful. In this chapter we conduct a case study of estimating the resource usage of the VCVA [100]. VCVA is a critical component in VMware's virtualized infrastructure since it is the administrative point of contact for managing hypervisors, virtual machines, virtual networks, and storage. A

straightforward strategy to respect the resource sizing would be to make a rough estimation of the resources needed by using domain expert knowledge, and then over provision the resources. However, the complexity of the VCVA, administrative requirements, provisioning and monitoring workflows make it difficult to reason about what to factor into the estimation.

One contribution of our work is to use data mining techniques to identify those factors. In particular, we analyze 200GB compressed profiler logs of VCVA that run on various platforms. Each profiler log consists of 12,000 features, and create a significant information overload for our estimates. Thus, we first employ feature selection and exploratory data analysis techniques to reduce the information overload in the profiler logs, and identify critical features that are associated with changes in memory consumption. The selected features are then corroborated and validated by domain experts.

Furthermore, while VCVA presented as a single monolithic entity, it is actually a collection of interacting services and components. These interactions result in a non-linear relationship between the infrastructure being managed by the VCVA, the active workflows and component interactions. Due to these dynamics, rough rules of thumb for sizing VCVA depend on the particulars of a given environment [101]. However, no guidance is given on how much these factors may influence the final setting, leaving it to the user to determine an appropriate sizing by trial and error. A second contribution of our work is to attempt to construct a resource estimation model in the presence of the service, component, workload and deployment dynamics.

Modeling the resource usage of a non-trivial application is a challenging task. Component interactions and application complexity can result in complex, non-linear relationships between application performance and resource usage. The novelty in our work stems from taking advantage of the deployment of multiple instances of an application in public, private or hybrid clouds and coping with the inevitable *noise* and disparities introduced by the different deployment conditions. We use data from multiple instances of the same virtual appliance or application to detect and diagnose performance anomalies [102] and estimate

Table 7.1.

List of environments and their characteristics. The data for each sample comes from multiple log statements that share the same time interval in the VCVA's profiler log.

| Environment | Windows Samples | Linux Samples | Number of Runs |
|---------------|-----------------|---------------|----------------|
| Hands on lab | N/A | 39414 | 24 |
| Nimbus | N/A | 10619 | 1 |
| Onecloud | 10554 | N/A | 8 |
| Perf | 8999 | 9704 | 200 |
| Private | N/A | 28823 | 2 |
| Featurestress | 3088 | 695 | 10 |
| VoV | 11107 | N/A | 8 |

resource usage and application performance to make better provisioning and consolidation decisions.

The remainder of this chapter is organized as follows. Section 7.2 briefly describes the seven environments where we collect data from the deployed instances of the VCVA. Section 7.3 outlines our data processing, feature-selection, visualization and model pipeline. Section 7.4 presents our evaluation results. Section 7.5 describes related work and Section 7.6 presents our conclusions and future work.

7.2 Datasets

VMware vCenter Server (vCenter or VC) provides centralized management of virtualized hosts and virtual machines from a single console. VCVA is a self-contained virtual machine image that can be deployed and run as a virtual machine on the VMware ESXi hypervisor. VCVA contains all of the components used by or needed by VC. Each instance of the VCVA produces *profiler logs*, which provide some insight into the activities taking place inside the appliance. The profiler logs contain, among other things, performance met-

Table 7.2.

List of example categories, ProcessStats, InventoryStats, SessionStats and RateCounter with associated features from the profiler logs.

| Category | Description | Features |
|----------------|---------------------------------------|----------|
| ProcessStats | physical memory usage, User Cpu Usage | 10 |
| InventoryStats | Number of Clusters, Virtual Machines | 10 |
| SessionStats | Number of Sessions | 1 |
| RateCounter | FilterCreates, FilterDestroys | 30 |

rics such as memory, cpu usage, information on the inventory such as the number of virtual machines, operation activities such as the frequency of powering on or cloning virtual machines. The VCVA is packaged as a pre-configured Linux or Windows virtual machine where we consider each operating system platform separately during the modeling phase.

We collected and analyzed profiler logs for the VCVA from seven different deployments with instances of VC running on both Windows and Linux. Each profiler log bundle includes a set of workloads that are recorded for different runs of the VC application. This gives us the ability to observe different workloads on different operating systems for the VCVA. We collect profiler logs from the VCVA instances in the environments listed below:

Hands on Lab (Hol): Collected from Hands on Lab at VMworld 2012. During the HoL customers try out new or soon-to-be-released versions of VMware products and familiarize themselves by doing exercises that focus on key activities, operations or workflows against virtualized infrastructure. We collect logs from the vCenter instances that manage the deployments of virtual infrastructure that support the exercises.

Performance (Perf): Collected from the Performance Group at VMware during their nightly tests of vCenter performance. This particular workload includes several high-load tests to ensure the performance of vCenter.

VMware on VMware (VoV): This environment is used internally for providing services within VMware.

Table 7.3.

The list of features that are discovered by mutual information test. All features are common to both Linux and Windows platforms.

| | |
|--------------------|--------------------|
| IncomingCalls | VpxdCache/Size |
| NumSessions | NumPoweredOnVMs |
| NumVirtualMachines | RetrieveContents |
| HostSync | HostSyncVpxaCalls |
| TaskInfoCalls | TotalOutgoingCalls |
| PropertyAssigns | PropertyRemoves |

Private: Small single-developer instances of the VCVA used for managing vSphere workloads such as performance testing and debugging applications.

Nimbus: This environment is an internal cloud for developers inside VMware.

Featurestress: This environment is used for running stress tests of various Virtual Center features.

Onecloud: This environment is VMware's internal cloud to run applications.

Table 7.1 shows the number of samples per deployment and operating system platforms. In some cases, VCVA deployment is operated only on a single operating system platform. Table 7.2 shows a sample list of features from the profiler logs. The total size of compressed collected profiler logs are 200GB and include 12,000 such features.

7.3 Methodology

In this section, we first select informative and non-redundant features (Section 7.3.1). Then, we use exploratory data visualization methods to investigate the relationship between selected features and performance metric of interest (Section 7.3.2).

Table 7.4.

Top triple features with high mutual information. All features are validated by domain experts.

| |
|--|
| NumSessions + IncomingCalls + PropertyAssings |
| NumSessions + IncomingCalls + TotalOutgoingCalls |
| NumSessions + IncomingCalls + VpxdCacheSize |

7.3.1 Feature Selection

Feature Selection component will help us to identify important and intuitive features that have an impact on a particular performance metric of interest and help us analyze how these features are related. Original feature space has approximately 12,000 features and many of them are redundant or irrelevant to the performance metrics of interest. By performing feature selection on the data, we expect to select relevant and informative features and improve the interpretability of the model.

Mutual Information (MI) measures how much knowing a particular feature reduces uncertainty about the performance metric of interest. Our main intuition is that we can eliminate the features that do not contribute to making a decision towards the performance metric of interest. Similarly, we can treat the features with high MI as intuitive and important since they significantly reduce the uncertainty about the performance metric of interest. Mutual Information has been shown [103] to be a submodular function which intuitively agrees with the diminishing return property since selecting an informative feature reduces the uncertainty.

We computed MI between features extracted from the logs, and performance metric of interest (e.g. physical memory usage). Then, we identified top features with highest MI out of 12,000 features. Figure 7.1 visualizes a subset of top features for both Linux and Windows platforms. After identifying top features with MI, we conduct additional experiments to investigate whether there is an agreement or overlap in the top features between different platforms such as Linux and Windows. Table 7.3 shows a list of features

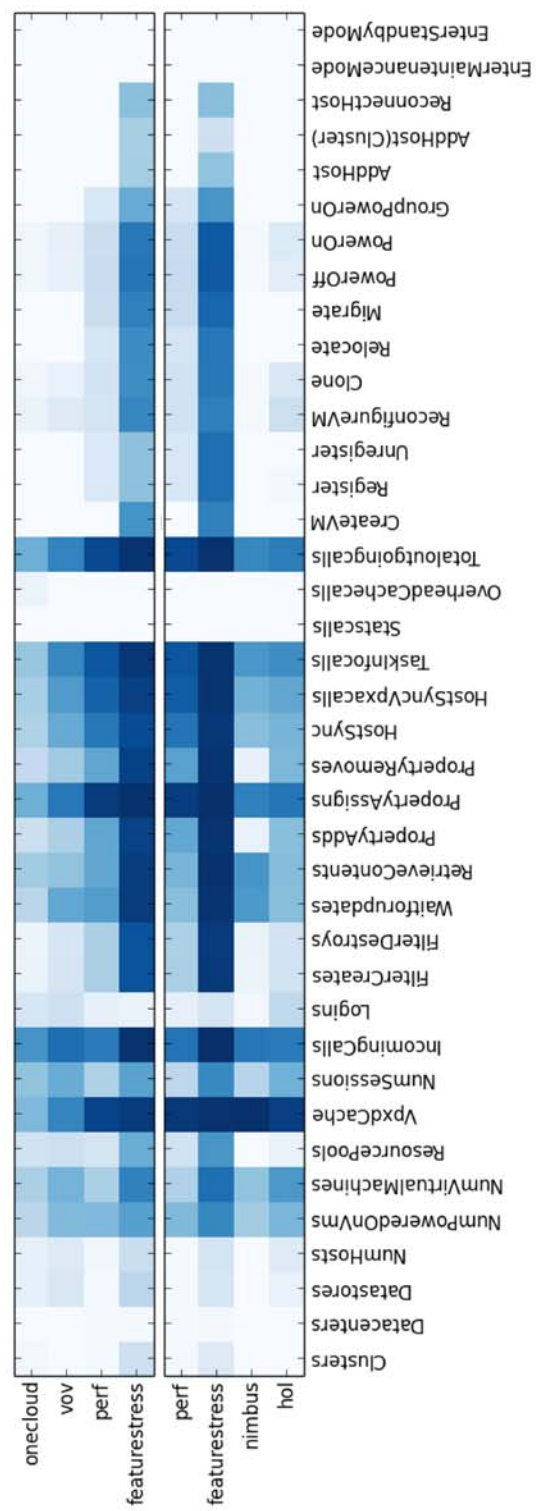


Figure 7.1. Mutual Information Matrix for different datasets on different platforms (darker color means high MI where lighter color means low MI). Top section represents Windows platform, and bottom section represents Linux platform.

which are identified as common features across both platforms. Finally, we computed MI between a linear combination of triple features and the performance metric of interest and identified top features. Our main intuition for this experiment is to see whether we can come up with a set of three features that have an impact on the performance metric of interest and whether domain experts can validate top triple-features as intuitive (see Table 7.4).

After performing MI experiments, we validate whether identified feature sets are intuitive and informative by consulting to domain experts. Domain experts identified **NumSessions**, **IncomingCalls** and **VpxdCache** as top three features that have an impact on physical memory usage, which is a subset of features we identify automatically in Table 7.4.

7.3.2 Exploratory Data Analysis

Since our goal is to construct a model from a set of heterogeneous sources, we would like to determine (1) the diversity of the sources used for training and predicting and (2) understand the limitations of our model when attempting to generalize to new or previously unseen instances of the VCVA. We use a number of visualizations to highlight both issues. We first perform a 2-D projection of the data by using Principal Component Analysis to identify whether there is a structural consistency between different environments on both platforms. We then perform a 3D heatmap analysis based on the top 3 features identified in previous section to see whether there is a similarity in terms of Physical Memory Usage between different platforms.

Principal Component Analysis We perform Principal Component Analysis (PCA) on the dataset to provide a simpler representation of the data and to understand the relationship between different environments on both platforms. PCA allows us to project the data from a higher dimension to a lower dimensional manifold such that each feature is represented by its projection along the line. We decided to perform a 2D projection of the data due to visualization purposes.

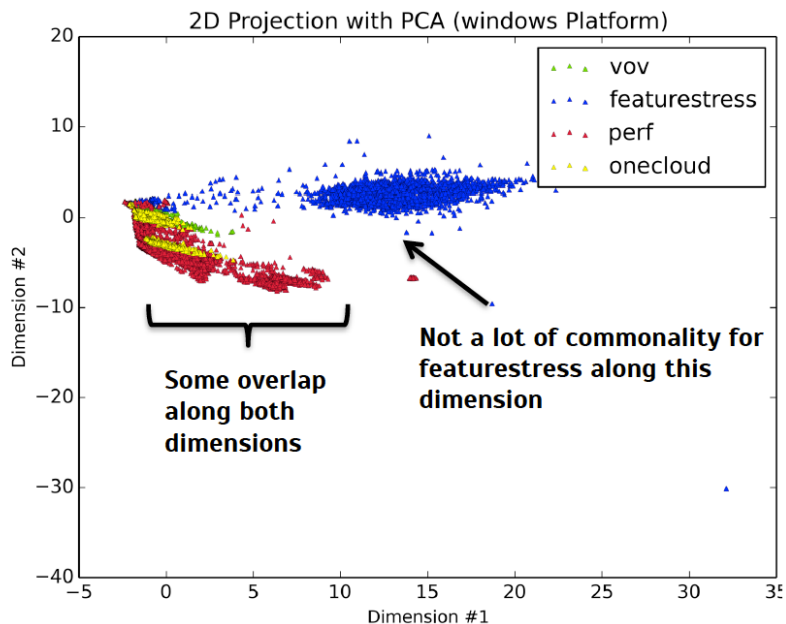


Figure 7.2. 2D projection via PCA Analysis on Windows platform. All environments on Windows platform except *Featurestress* have some overlap along both dimensions.

Figure 7.2 shows 2D projection for different environments on Windows platform. We can see that all environments on Windows platform except *Featurestress* have some overlap along both dimensions. However, *Featurestress* environment have almost no overlap along Dimension 1 and very little overlap along Dimension 2. On the other hand, there is a structural similarity along both dimensions between *Onecloud*, *Vmware on Vmware* and *Perf* datasets, which means that one should expect to get reasonable results if we perform a holdout experiment. However, a good generalization for *Featurestress* dataset does not seem to be possible since it has a quite different characteristic than the other observed datasets.

Figure 7.3 shows 2D projection for different environments on Linux platform. Similarly to the previous case, we observe an agreement among different environments on Linux platform except *Featurestress* which only has some overlap along first dimension. There is structural similarity along both dimensions between *Hol*, *Nimbus*, *Perf* and *Private* datasets. We expect to get reasonable results if we perform a holdout experiment

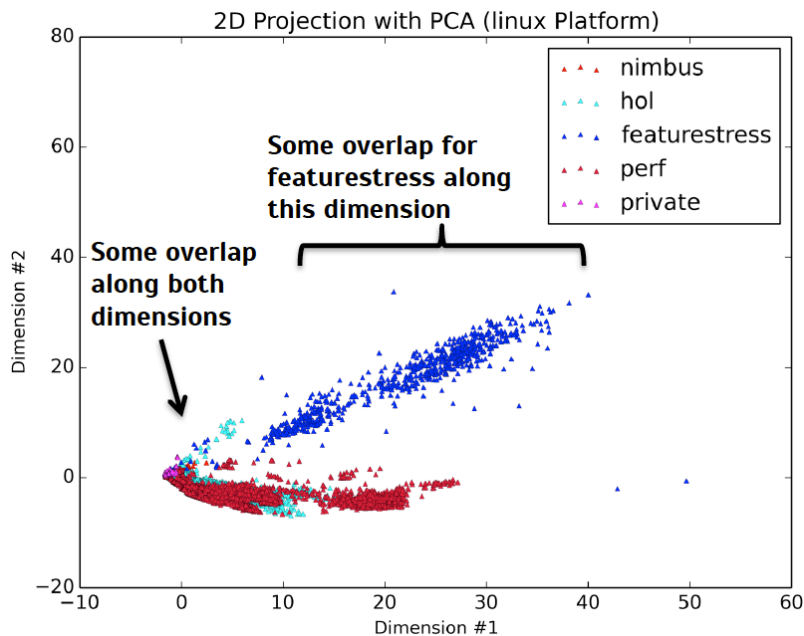


Figure 7.3. 2D projection via PCA Analysis (Linux Platform). We observe an agreement among different environments on Linux platform except *Featurestress* which only has some overlap along first dimension.

within these datasets. Even though we do expect to get a good generalization for Featurestress environment, we expect to get a better performance than Featurestress in Windows platform due to the commonality along the first dimension.

3D Heatmap Analysis For this component, we plot 3D visualization of the environments using the top three features that selected by Mutual Information experiment and validated by domain experts. We further applied a heatmap coloring to the points such that each point in 3D space takes its color based on the actual physical memory usage value it corresponds.

Our main intuition in this experiment is to see whether there is a color-based consistency between nearest neighbors. If two points are close to each other in 3D space, we expect them to share similar shades of the same color tone since three coordinates have a significant impact on the value of physical memory usage. Figure 7.4 shows the heatmap-based 3D plot for Linux platform. We can see that there are many consistent color-based clusterings in the plot which indicates that if two points are close to each other in 3D

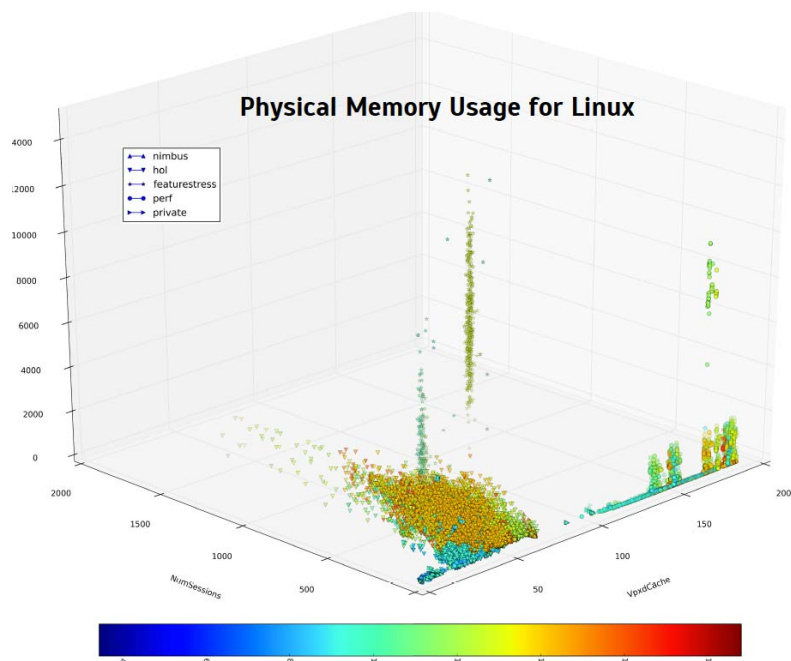


Figure 7.4. 3D plot with a heatmap based on physical memory usage on Linux platform.

space, then they tend to share similar physical memory usage values. Figure 7.5 shows the heatmap-based 3D plot for Windows platform and we can see that most of the environments have consistent color-clusterings except Featurestress environment.

7.4 Experiments

Model Selection We first would like to identify a model family to perform the prediction task we are interested in. PCA-based visualizations in the previous section demonstrates various level of disparities in the characteristics of individual datasets. For instance, datasets such as Featurestress can significantly bias the predictions on function-based families since the characteristics of the dataset is quite different than others. On the other hand, using a similarity-based model, such as K-Nearest Neighbors, where a prediction is made based on the closest datapoints can lessen the extent of skewness in the predictions. Futhermore, 3D heatmap visualization in the previous section also suggests that a near-

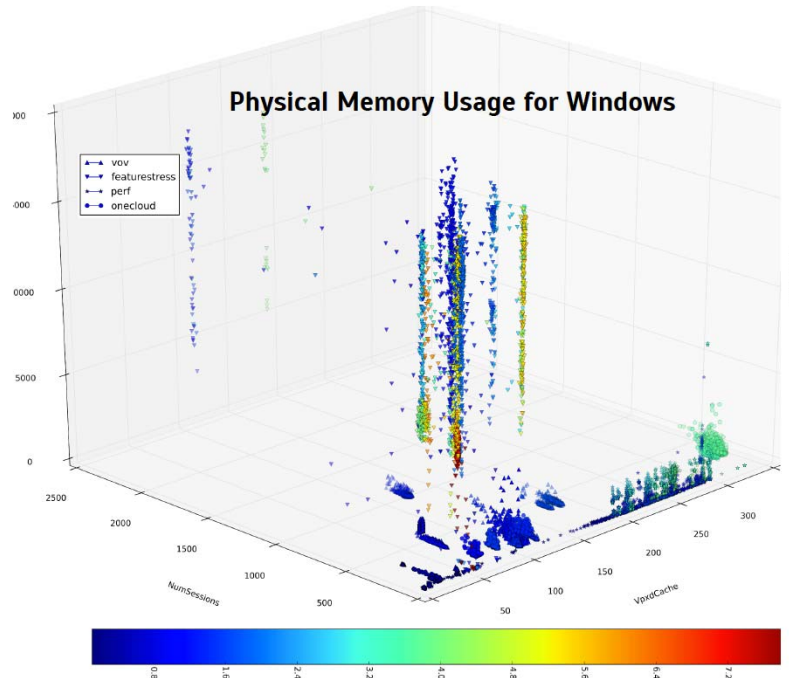


Figure 7.5. 3D plot with a heatmap based on physical memory usage on Windows platform.

est neighbor based model would be a good fit for our problem since there is a reasonable consistency for physical memory usage within neighborhoods.

Evaluation Metrics We use Recall as our main metric of merit which is defined as

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (7.1)$$

We consider a true positive as one where our predicted Physical memory usage is greater than or equal to (\geq) the observed value and a false negative as one where our predicted resource usage is less than ($<$) the observed value. We also pay attention to the Root Mean Squared Error (RMSE) of our predictions – we prefer smaller RMSE values – cognizant of the fact that predicting an extremely high value for Physical Memory Usage would mislead the model into thinking the results were very positive when in fact gross over estimates for Physical Memory Usage would be wasteful and overly conservative.

Table 7.5.

Physical memory usage on Linux platform where we trained on 70% of the data and tested on 30% of the data.

| Source | RMSE (GB) | Recall |
|---------------|-----------|--------|
| Featurestress | 0.66 | 0.77 |
| HoL | 1.56 | 0.76 |
| Nimbus | 0.45 | 0.74 |
| Perf | 2.33 | 0.75 |
| Private | 0.23 | 0.75 |

Results The first question we seek to answer is whether our nearest neighbor model works within a dataset, e.g., train on a portion of the Perf dataset on Windows and predict the physical memory usage of the unseen portion of the Perf dataset on Windows. Tables 7.5 and 7.6 show that we can obtain good predictions ($> 70\%$ Recall and within 3GB RMSE) within each dataset. For our evaluation we repeatedly partition each dataset (10 times/folds) into a randomly selected 70% of the datapoints for training and 30% for testing. We use the max of the nearest $k = 3$ neighbors as the prediction of physical memory usage.

The second question we seek to answer is whether hold-out experiments are possible, i.e., training a model by holding out a portion of the dataset, and predicting the resource usage on the left-out part. This allows us to reason about whether the behavior of the VCVA on a given platform is consistent enough given the disparities in the environments and workloads in different environments.

Results in Tables 7.7 and Table 7.8 suggests that we are able to perform hold-out experiments with reasonable performances. In particular, on Linux platform, we achieve good recall results ($> 80\%$ with a RMSE within 5GB) for three of the five datasets (Featurestress, Perf and Private). However, we perform poorly on Nimbus and HoL when we try to predict their memory usage by using only the left-out portion of the datasets. In the case

Table 7.6.
Physical memory usage on Windows platform where we trained on 70% of the data and tested on 30% of the data.

| Source | RMSE (GB) | Recall |
|---------------|-----------|--------|
| Featurestress | 1.33 | 0.78 |
| OneCloud | 0.58 | 0.76 |
| Perf | 0.39 | 0.77 |
| VoV | 0.01 | 0.74 |

of Nimbus, we consistently *underestimate* the physical memory usage. These underestimates are due to the narrow range of physical memory usage values in the Nimbus dataset itself, the minimum value observed in our Nimbus data is 12GB, the max is 15.5GB and the mean is 14.2GB with a standard deviation of 0.3GB. The relatively narrow band of physical memory usage values in Nimbus means that regardless of the variations in the key features we have identified the physical memory usage remains relatively flat, i.e., their influence of our key features on the physical memory usage in Nimbus is very different from the influence these same features exert in the other datasets. In the case of HoL the RMSE of our estimates is off by ~ 4 GB, which may be a result of the very different workloads that are executed in the HoL as compared to the other environments. For instance, the use of pre-prepared pools of virtual infrastructure such as VMs and datastores to support the exercises may result in fewer VC interactions related to managing virtual infrastructure, and triggered by administrators while customers are running exercises. Further, administrative and management activities may be deferred to times before or after peak demand for exercises which would result in a different profile of interactions with Virtual Center, influencing the contents of the profiler logs.

Results on Windows platform results in smaller RMSE values. The model performs the worst on Featurestress as a function of the other datasets. This is an expected behavior (see Figure 7.2) since Featurestress has little in common with the other datasets. Thus, building

Table 7.7.

Physical memory usage on Linux platform where we train on k-1 datasets and predict the left-out dataset.

| Prediction target | RMSE (GB) | Recall |
|-------------------|-----------|--------|
| Featurestress | 4.80 | 0.97 |
| HoL | 4.19 | 0.23 |
| Nimbus | 3.78 | 0.00 |
| Perf | 3.46 | 0.93 |
| Private | 0.983 | 0.82 |

a model from the others and trying to predict Featurestress consistently underestimates the physical memory usage. Whereas the Recall for OneCloud is relatively low (20%), and our estimate of its physical memory usage results in a small underestimate, e.g. ~ 500 MB. Similarly, our estimates for Perf have an RMSE of less than 1 GB.

7.5 Related Work

The work most similar to ours is [102]. In that work the authors crowdsource telemetry data to automatically identify virtual machines that run the same application and then use that for collective/collaborative debugging. In our work, the application of interest is fixed (we only focus on the VCVA) and we focus on gathering information from profiler log-data from different deployments of this application and trying to construct a resource usage model that is robust and accurate despite the differences in workloads, number of VMs managed, the complexity of the VCVA etc. in each of the deployments. Our results show that for the VCVA we can identify a consistent set of features that describe its memory usage across environments and operating system platforms.

The idea of using group behavior to reason about the behavior and characteristics of an application deployed in various settings was discussed in [104] in the context of security and dependability – collections of independent instances of the same application coopera-

Table 7.8.
Physical memory usage on Windows platform where we train on k-1 datasets and predict the left-out dataset.

| Prediction target | RMSE (GB) | Recall |
|-------------------|-----------|--------|
| Featurestress | 3.38 | 0.09 |
| OneCloud | 0.55 | 0.20 |
| Perf | 0.93 | 0.69 |
| VoV | 0.30 | 0.91 |

tively monitor their execution for flaws and attacks and notify the community when such events are detected. In our work, we focus on resource usage and performance rather than reliability.

Carat [105] crowdsources a model of the factors influencing energy usage from a population of mobile handsets running various applications. The authors distinguish between *energy bugs* – specific instances of an application that drain the battery much faster than other instances of that same application – and *energy hogs* – an application that drains the battery faster than the average application.

Building a model of a complex application is easier if the application has purposefully been constructed and instrumented in a way that makes it easy to collect the relevant data/metrics [106]. Our experience with the VCVA highlighted the richness of the data contained in the profiler logs (we consider the detailed logs a strength in light of our ability to employ statistical techniques to select critical features and build accurate models).

7.6 Conclusions

In this chapter, we developed techniques for sizing Virtual Center Virtual Appliances (VCVAs) by mining the profiler log data from multiple deployments. We use feature selection techniques to automate the identification of key features – our automatically selected features agree with features selected by experts. We use various data visualization tech-

niques to understand the data and differences between the platforms. Finally we use a structural model to account for the disparities of different VCVA deployments and the relative complexity of the VCVA. We show results of better than $\sim 70\%$ prediction accuracy for predicting physical memory usage.

8 FUTURE WORK

In this chapter, we discuss two future directions of our research. In particular, we first present a submodular framework for graph comparison. Our framework is inspired by latest state-of-the-art submodular document summarization models and extends the graphlet kernel to encourage *diversity* while avoiding redundancy. Our experiments on several benchmark datasets show that our framework outperforms the graphlet kernel in terms of classification accuracy by using 50% less samples.

Then, we present a submodular framework to automatically generate color palettes from an image. We formulate color palette generation problem as a *coverage* task and propose a submodular framework for which we can provide an efficient solution with near-optimal approximation guarantees. Our method balances the *coverage* and the *diversity* of colors presented in an image, and exploits the notion of *diminishing returns*.

8.1 A Submodular Approach for Graph Sampling

A commonly used paradigm for representing graphs is to use a vector that contains normalized frequencies of occurrence of certain motifs or sub-graphs. The graphlet kernel of [16] uses induced sub-graphs of k nodes (christened as *graphlets* by [15]) as motifs in the vector representation, and computes the kernel via a dot product between these vectors. However, existing sampling methods for graphlets suffer from a few drawbacks.

Redundancy: The graphlet frequency distribution exhibits a power-law behavior, that is, the frequency of certain graphlets grows as a power of others. This becomes a significant problem when the frequency of informative graphlets are overwhelmed by graphlets that do not carry any discriminating power across graphs. Figure 8.1 illustrates such an example where graphlet G_{20} occurs significantly higher than more informative graphlets such as G_{32}



Figure 8.1. A sample graph from MUTAG [46] dataset that illustrates the decomposed graphlets where size of each graphlet is positively correlated with its frequency in the graph.

and G_{34} . An ideal framework should take this observation into account and avoid selecting redundant graphlets.

Diversity: It can be observed that graphlets of a given size k are related to each other [83]. For instance, graphlets G_{36} , G_{37} and G_{40} only differ by one node and one edge. Therefore, a sampling scheme that does not take the inherent similarity between graphlets does not respect the diversity among the samples. An ideal framework should encourage diverse graphlets when performing sampling.

8.1.1 Proposed Solution

Our algorithm takes advantage of the inherent similarity between graphlets of size $\leq k$ and $k + 1$. The key observation of our proposal is that, one can use this relationship to represent a graphlet of size $k+1$ as a *probability distribution* over size $\leq k$ graphlets. Let G_i represent a graphlet of size $k+1$, G_j represent a graphlet of size k and n_{ij} denote the number of times G_j occurs as a sub-graph of G_i . Computation of n_{ij} is done by deleting a node of G_i and counting how many times graphlet G_j is produced as a result [107]. Repeating the same process for all graphlets of size $1 \leq l \leq k$ and normalizing the frequencies, we obtain a distribution for graphlet G_i by means of smaller-sized graphlets (see Figure 8.2 for an example decomposition).













| |  G_2 |  G_3 |  G_5 |  G_6 |  G_7 |  G_{15} |  G_{10} |  G_{11} |  G_{14} |  G_{18} |
|--|---|---|---|---|---|--|--|--|--|--|
|  G_{35} | 0.35 | 0.35 | 0.09 | 0.09 | 0.02 | 0.02 | 0.02 | 0.0 | 0.01 | 0.0 |
|  G_{41} | 0.21 | 0.49 | 0.07 | 0.07 | 0.09 | 0.03 | 0.0 | 0.01 | 0.0 | 0.01 |

Figure 8.2. An example for graphlets G_{35} and G_{41} that shows the decomposed graphlets of size $\leq k$, associated with normalized frequencies. As can be seen from the table, both graphlets cover graphlets $G_2, G_3, G_5, G_6, G_7, G_{15}$ with different weights.

An alternate way to interpret this probability distribution is as follows: each graphlet G_i of size $k+1$ *covers* graphlets of size $\leq k$ with a certain amount. Thus, whenever we sample a graphlet G_i , we also *cover* some portion of other graphlets. For instance, whenever we pick graphlet G_{41} , we cover the following graphlets with associated probabilities: $G_{41} = \{G_2 : 0.21, G_3 : 0.49, G_5 : 0.07, G_6 : 0.07, G_7 : 0.09, G_{11} : 0.01, G_{15} : 0.03, G_{18} : 0.01\}$. Therefore, one can view this as a *maximum coverage* problem where our main objective is to select m subsets from a collection $S = \{S_1, S_2, \dots, S_n\}$ of n subsets such that the union of the selected sets $\bigcup_{i=1}^m S_i$ has the maximum coverage. In other words, we would like to maximize the following objective function:

$$A^* = \operatorname{argmax}_{A \subseteq S: |A| \leq m} f(A)$$

This problem is well-known to be NP-hard [108]. However, it can be formulated in terms of *submodular* functions where a greedy algorithm is guaranteed to approximate the optimum solution within a factor of $(1 - 1/e) \approx 0.63$ [67] (see Chapter 2).

We formulate our task as a submodular optimization problem and adapt our framework from document summarization task of [69]. In order to apply our framework, we changed the random sampling approach slightly as follows: whenever we sample a graphlet, we also sample its immediate neighbors. Thus, our sampling procedure is still random, but we also have a chance to capture the graphlets within similar neighborhoods. Let S represent a set of neighborhoods of size k graphlets, $P_{<k}$ represent the set of unique graphlet types

of size $< k$, p represent an arbitrary graphlet in $P_{<k}$, i represent an arbitrary neighborhood and j represent a graphlet in that neighborhood. We then define our submodular objective function as follows:

$$F(S) = \sum_{p \in P_{<k}} \left(\sum_{i \in S} \sum_{j \in i} r_{pj} \right)^\alpha \quad (8.1)$$

where r_{pj} is the weight of graphlet p in j and α is curvature parameter that determines the rate that reward diminishes over time. Interpretation of this framework is as follows: we measure the diversity of the selected set S in terms of graphlets of size $< k$. For each graphlet p in the set of unique graphlets of size $< k$, we quantify the amount that is already covered by the selected neighborhoods in S . Thus, for each selected neighborhood i in the set S , we sum how much we already covered the $< k$ -sized graphlet p .

Proof Equation 8.1 is submodular. $(x)^\alpha$ is a non-decreasing concave function. Inside of $(x)^\alpha$, we have a modular function with non-negative weights, thus monotone. Applying $(x)^\alpha$ to such a monotone submodular function yields a submodular function, and summing submodular functions retains submodularity property. Therefore, $F(S)$ is submodular. ■

8.1.2 Preliminary Experiments

We compare our framework against graphlet kernel. Both kernels are coded in Python and normalized to have a unit length in the feature space. Moreover, we use 10-fold cross validation with a binary C -Support Vector Machine (SVM) where the C value for each fold is independently tuned using training data from that fold. In order to exclude random effects of the fold assignments, this experiment is repeated 10 times and average prediction accuracy of 10 experiments with their standard deviations are reported.

In order to achieve a fair comparison, we first sampled 100 neighborhoods of graphlets from a given dataset. Then, we feed exactly the same set of graphlets into our framework and submodularly selected 50 of them. Thus, our framework uses 50% less information than the graphlet kernel in the following experiments.

Table 8.1.
Comparison of classification accuracy for the graphlet kernel with our method where **STD** standard deviation, and **SE** represents standard error.

| DATA SET | GRAPHLET KERNEL | SUBMODULAR GRAPHLET KERNEL |
|----------|-----------------------------|------------------------------------|
| MUTAG | 77.11 (STD: 1.54, SE: 0.48) | 80.22 (STD: 1.08, SE: 0.34) |
| PTC | 55.82 (STD: 1.10, SE: 0.35) | 57.14 (STD: 1.35, SE: 0.42) |
| ENZYMES | 23.35 (STD: 1.30, SE: 0.41) | 25.10 (STD: 0.92, SE: 0.29) |
| NCI109 | 62.15 (STD: 0.28, SE: 0.09) | 62.28 (STD: 0.22, SE: 0.07) |

Datasets We used the following benchmark datasets used in graph kernels: MUTAG, PTC, ENZYMES and NCI1 (see Section 3.4.2 for a detailed description of the datasets).

We compare graphlet kernel with our method (see Table 8.1). As can be seen from the results, our framework is able to outperform base kernel with statistically significant improvements (shown in bold) while achieving a smaller standard error on most of the datasets.

In this section, we propose a novel framework to incorporate diversity when performing graph comparison. Even though we restricted ourselves to graph kernel literature in this study, our framework introduces a new perspective to graph sampling and summarization. For instance, our framework can be easily adoptable to summarize diverse aspects of a given graph for exploration or visualization purposes.

8.2 A Submodular Approach for Color Palette Generation

A color palette is defined as a set of color swatches which can be used to represent an image, and can be utilized in various areas. For instance, the usage of color palettes in web design is a common practice to assign consistent but attractive colors to website layouts [109].

Color palette generation has been extensively studied in the past. [118] proposes a framework to generate harmonized colors from an image by utilizing harmonious color

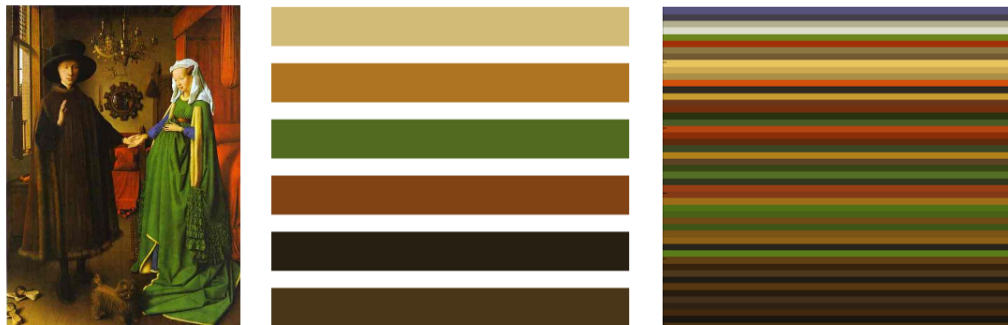


Figure 8.3. “The Arnolfini Portrait” by Jan van Eyck (left). Top $k = 6$ colors extracted using k-means (center). Top $k = 50$ colors extracted using k-means (right).

templates by the color theorist Yutaka Matsuda [119]. This work is followed by [120, 121] which also utilizes Matsuda’s harmonious color templates. [117] proposes a framework by using linguistic concepts associated with color moods. [122] proposes a framework to generate color palettes to enhance images by utilizing aesthetically popular color themes from Adobe Kuler <https://color.adobe.com>. [110] proposes a framework based on a regression model and utilizes six types of features among which saliency is used as the main feature. Jahanian [111] proposes a framework that automatically extracts colors from an image by using saliency maps.

In this work, we conjecture that the notion of diminishing returns naturally arise in color palette generation problem, and we approach the color palette generation as a *coverage* task. We propose a submodular framework that generates a color palette with near-optimal approximation guarantees $(1 - 1/e)$. To best of our knowledge, our work is the first to formulate color palette generation problem as a submodular optimization task. Moreover, our method is complimentary to the previously proposed approaches since our framework can be configured to work with saliency maps as in [111], bag of colors [114] or SIFT [115] to select a diverse set of colors.

One of the most common approaches to extract colors from an image is to use traditional clustering methods such as k -means [112, 113] and cluster the pixels in an image in terms of colors. However, color palettes that are obtained by such techniques suffer from

several drawbacks. First, the obtained color palettes are often dominated by background and foreground colors of the image. Second, the selected colors are often similar to each other, thus the color palette suffers from the *redundancy* problem. Figure 8.3 illustrates top colors extracted from the “The Arnolfini Portrait” painting by Jan van Eyck using k -means. As can be seen from the figure, diverse colors such as *blue* and *white* are dominated by background and foreground colors, and do not appear in the color palette. An ideal color extraction algorithm should cover dominant colors in the image while also encouraging the selection of diverse colors. There has been recent approaches to produce a better representation of the colors in an image. [110] proposes a regression model that is trained on 1600 color palettes for 40 images from 160 human participants, and studies the concept of *saliency*. However, their approach requires supervision through a training dataset, and is not completely autonomous. Recently, [111] proposes an automated method that is based on the saliency map of a given image in order to extract salient colors.

In this work, we approach color palette generation problem as a *coverage* task. Given an image \mathcal{I} and a budget k , we are interested in extracting a color palette \mathcal{S} that consist of k swatches such that the palette *covers* the entire image as *relevant* and as *diverse* as possible. In particular, our method takes a set of candidate colors \mathcal{V} extracted from the image \mathcal{I} , and the palette size k as input. Then, in each step, it greedily selects a color from the set \mathcal{V} by maximizing the overall similarity of the color to the image, while positively rewarding the diversity. *Diversity* is an important component of our notion of coverage, that is, while we want to select colors that are similar to the image as much as possible, we do not want to select colors that are too similar to each other. Therefore, we formulate our framework as a submodular optimization problem for which we can provide an efficient and near-optimal solution within $(1 - 1/e)$. Moreover, our method is completely automated, and *complementary* to previous approaches. For instance, instead of using a color histogram to generate the ground set, our framework can be configured to cover saliency maps [111], bag of colors [114] or SIFT [115].

8.2.1 Proposed Solution

Our main goal is to generate a color palette that covers the image while at the same time respecting the diversity of the colors that form the palette. More formally, let the ground set \mathcal{V} represent a set of n colors that is extracted from an image \mathcal{I} . We are interested in selecting a subset $\mathcal{S} \subseteq \mathcal{V}$ to represent the entirety of ground set \mathcal{V} , subject to a cardinality constraint $k \leq n$. Let us assume that we have a set function $\mathcal{F} : 2^{\mathcal{V}}$ that measures the quality of set \mathcal{S} . Then, we can define the color palette problem as a combinatorial optimization task as follows:

$$\mathcal{S}^* = \operatorname{argmax}_{\mathcal{S} \subseteq \mathcal{V}: |\mathcal{S}| \leq k} \mathcal{F}(\mathcal{S})$$

where \mathcal{S}^* represents the optimal solution subject to a cardinality constraint k which denotes the total number of swatches in the color palette \mathcal{S} . Thus, finding a set of colors that covers the image \mathcal{I} becomes a *budgeted maximum coverage* problem [86]. This is a well-known NP-hard problem [67], however, it has been shown that if \mathcal{F} is monotone submodular, then a greedy algorithm can solve the problem near-optimally with $(1 - 1/e)$ approximation factor [67].

Submodular algorithms are widely used in document summarization area [69,87] where the task is to select a subset of sentences from a document, similar to our color palette problem. One option to obtain a coverage function that satisfies the desired properties of is to design a framework similar to Maximal Marginal Relevance (MMR) [88]. MMR is one of the most popular document summarization frameworks that greedily selects the most relevant sentences to a document while at the same negatively penalizing the sentences that are too similar to the ones already selected in the summary. In particular, the objective function to add element s_i from the ground set \mathcal{V} to set \mathcal{S} is defined as,

$$\lambda \operatorname{Sim}_1(s_i, q) - (1 - \lambda) \max_{s_j \in \mathcal{S}} \operatorname{Sim}_2(s_i, s_j)$$

where Sim_1 measures the similarity of element s_i to the query or user profile q , Sim_2 measures the similarity between unit s_i and unit s_j , and $0 \leq \lambda \leq 1$ is the trade-off coefficient. Thus, one can design a similar objective function that maximizes the similarity between

a color $c_i \in \mathcal{V}$ and the image \mathcal{I} while minimizing the similarity between $c_i \in \mathcal{V}$ and already selected colors $c_j \in \mathcal{S}$. However, even though MMR-type functions are known to be submodular [69], they are not monotone due to the negative penalization term and do not satisfy the constant-factor approximation guarantee of the greedy algorithm [69]. Since we would like to design an objective function that produces a color palette with near-optimal solution $(1 - 1/e)$, this option is not viable to us. Therefore, we adapted the submodular framework of [69] from document summarization task to color palette generation problem. The framework of [69] balances relevance and non-redundancy,

$$\mathcal{F}(\mathcal{S}) = \mathcal{F}_L(\mathcal{S}) + \lambda \mathcal{F}_R(\mathcal{S}) \quad (8.2)$$

where $\mathcal{F}_L(\mathcal{S})$ measures the *coverage* (i.e. how similar \mathcal{S} to the rest of the document), and $\mathcal{F}_R(\mathcal{S})$ measures the *diversity* (i.e. how diverse the sentences in set \mathcal{S}). Thus, instead of negatively penalizing redundancy, they positively reward diversity.

Designing the relevancy component The first component of Equation (8.2) is to design an appropriate function that captures the similarity of selected color palette to the image \mathcal{I} . We define the coverage component as follows:

$$\mathcal{F}_L(\mathcal{S}) = \sum_{c_i \in \mathcal{V}} \sum_{c_j \in \mathcal{S}} w_{c_i} \sigma(c_i, c_j) \quad (8.3)$$

where \mathcal{V} represents the color histogram extracted from the image \mathcal{I} , c_i represents an arbitrary color from the set \mathcal{V} , w_{c_i} represents the weight of color c_i in the histogram, c_j represents a color from the selected color palette \mathcal{S} and $\sigma(c_i, c_j)$ measures the similarity of color c_i to color c_j .

Generation of ground set \mathcal{V} : Given an image \mathcal{I} , we computed the histogram \mathcal{V} by using k -means algorithm where we cluster the pixels the image based on their color, and created a histogram based on the number of pixels assigned to each cluster. Then, the weight w_{c_i} for each color $c_i \in \mathcal{V}$ simply becomes a normalized frequency distribution, representing how many pixels fall into the cluster $c_i \in \mathcal{V}$. Note that the number of clusters when generating the ground set \mathcal{V} should be large enough to capture a diverse range of

colors present in the image in order to prevent foreground and background colors to dominate as illustrated in Figure 8.3. Therefore, we extracted $n = 50$ colors as the ground set \mathcal{V} where each color $c_i \in \mathcal{V}$ serves as a *candidate* color for the color palette \mathcal{S} . Figure 8.3 (right) illustrates the extracted $n = 50$ colors from the “The Arnolfini Portrait”. As can be seen from the figure, using a larger number of clusters allow k -means algorithm to capture a diverse set of colors such as *blue* and *white*. Note that even though we are able to capture diverse colors with a large number of clusters, the task we are interested in is to select a small subset $\mathcal{S} \subseteq \mathcal{V}$ to generate a compact color palette.

Similarity function $\sigma(c_i, c_j)$: This function measures the similarity between two colors c_i and c_j . For this purpose, we used *CIE94* [116] based on $L * a * b^*$ colorimetric system, and takes the differences in lightness, chroma and hue into account. Similar to the generation of ground set \mathcal{V} , this component is flexible and can be configured to work with any type of similarity function as long as weights remain non-negative.

Designing the diversity component The other important aspect of our objective function is to prevent redundant colors to dominate the color palette by encouraging *diversity*. Thus, the design of $\mathcal{F}_R(\mathcal{S})$ component should respect diversity of the colors in the selected set \mathcal{S} ,

$$\mathcal{F}_R(\mathcal{S}) = \sum_{m=1}^{\mathcal{P}} \sqrt{\sum_{c_i \in \mathcal{S}} \mathbf{1}[c_i \in \mathcal{P}_m]} \quad (8.4)$$

where \mathcal{P} represents the set of n clusters of the ground set \mathcal{V} , m represent an arbitrary cluster from this set, and \mathcal{P}_m represents the set of colors that belong to cluster \mathcal{P}_m . The cluster set \mathcal{P} is simply generated by using k -means on the ground set \mathcal{V} . Similar to the previous cases, the generation of \mathcal{P} is flexible in the sense that the set of colors can be distributed into clusters using different methods. For instance, in cases where linguistic content is available on colors [117], one can cluster the colors in ground set \mathcal{V} based on linguistic features. The main intuition behind using a square root function in $\mathcal{F}_R(\mathcal{S})$ is to apply diminishing returns to the currently selected colors in \mathcal{S} . For instance, whenever we select a color that belongs to a particular cluster, the gain of adding colors from the same cluster diminishes due to the concavity of the square root function. Thus, the objective

function is encouraged to select colors from other clusters that not yet contributed to the color palette \mathcal{S} .

Proof of submodularity After defining the components of the objective function, we now prove that $\mathcal{F}(\mathcal{S})$ defined by Equation (8.3) and Equation (8.4) is submodular.

Claim 8.2.1 *The color-palette selection algorithm in Equation (8.2) defined by Equation (8.3) and Equation (8.4) is submodular.*

Proof Equation (8.3) is a modular function with non-negative weights (hence, monotone). Similarly, sum of $\sum_{c_i \in \mathcal{S}} \mathbf{1}[c_i \in \mathcal{P}_m]$ in Equation (8.4) is also monotone. The monotone function in Equation (8.3) is surrounded by a square root function, which is a non-decreasing concave function. Using a concave function to this monotone function yields a submodular function [69]. The sum of a collection of submodular functions are submodular [72], thus $\mathcal{F}_R(\mathcal{S})$ is submodular. The summation of the modular function $\mathcal{F}_L(\mathcal{S})$ with submodular function $\mathcal{F}_R(\mathcal{S})$ does not violate the submodularity, and hence $\mathcal{F}(\mathcal{S})$ is submodular. ■

8.2.2 Preliminary Experiments

We qualitatively compared our method with two baseline methods as follows:

- **Submodular:** This approach uses $\mathcal{F}(\mathcal{S})$ defined by Equation (8.3) and Equation (8.4) to select a color palette.
- **Modular:** This approach greedily selects top k colors from the histogram \mathcal{H} to maximize the similarity of the set \mathcal{S} to the image \mathcal{I} . In other words, this approach only uses $\mathcal{F}_L(\mathcal{S})$ component of our framework.
- **Top-k:** This approach uses k -means algorithm to extract k clusters from the image \mathcal{I} where the center of each cluster is used as a color.

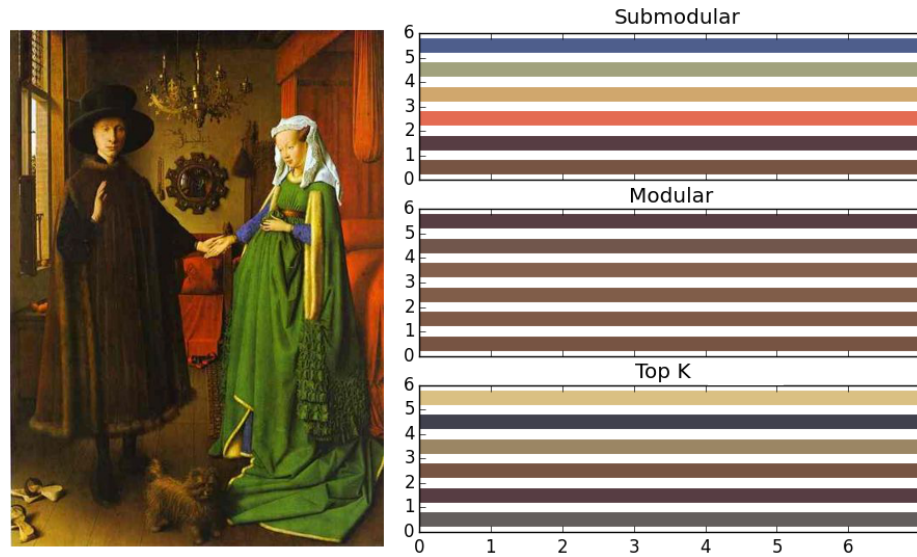


Figure 8.4. “The Arnolfini Portrait” by Jan van Eyck (left) and the extracted color palettes by **Submodular**, **Modular** and **Top-k** approaches (right).

Figure 8.4 illustrates the obtained color palettes for “The Arnolfini Portrait” by Jan van Eyck. As discussed earlier, **Top-k** method fails to capture diverse colors, and dominated by background and foreground colors of the painting. **Modular** approach completely dominates the color palette with *brown-ish* colors since it maximizes the similarity of the selected colors to the overall image. **Submodular** approach is able to generate a balanced color palette since the gain of selecting colors that are similar to already selected ones diminishes and allows diverse colors such as *blue* to get captured.

Color palettes are important aspects that are utilized in various areas in design, visual media and image retrieval. In this chapter, we approached the color palette generation problem as a coverage task, and proposed a submodular framework for automatic extraction of color palettes from images. Our method balances *relevancy* and *diversity*, and provides a near-optimal solutions with theoretical guarantees. Moreover, our approach is flexible and can be configured to use different notions of relevancy, and diversity.

9 CONCLUSIONS

In the recent years, there has been a significant increase in the amount of content generated daily, which forces both users and systems to cope with information overload. In this dissertation, we address challenges of information overload in two separate domains: (1) information overload in graphs and (2) information overload in text.

In the first part of the dissertation, we focused on information overload problem in graphs. In particular, we developed two frameworks to address existing problems in graph kernels, and demonstrate that both studies improve over popular graph kernels, namely Graphlet kernels, Weisfeiler-Lehman subtree kernels, and Shortest-Path kernels. In the first framework, we propose a novel study that *learns* the latent representations of sub-structures by leveraging the co-occurrence relationship of the features. In the second framework, we propose a general smoothing framework for graph kernels by taking *structural* similarity into account. Our framework is inspired by state-of-the-art smoothing techniques used in natural language processing. However, unlike NLP applications that primarily deal with strings, we show how one can apply smoothing to a richer class of inter-dependent sub-structures that naturally arise in graphs.

In the second part of the dissertation, we focused on information overload problem in text-related data. First, we focus on information overload in social news aggregation websites and propose a framework that tailors a *personalized* frontpage for individual users. Second, we focus on information overload arise in documents and video transcripts. In particular, we propose a novel summarization framework to summarize TED talks and formulate our objective function as a submodular framework that balances coverage and diversity. Third, we focus on information overload in system logs. In particular, we use a population of virtual machines from VMware [4], and demonstrate how to apply feature filtering and selection techniques to reduce the information overload and identify important features. Our contributions in this work are as follows.

Finally, we discuss two possible future directions of our research. In particular, we designed two submodular frameworks where information overload is present: we design a submodular graph sampling framework, and we design a submodular color palette selection algorithm. Both of these frameworks have promising directions that can be utilized in graph summarization tasks or color palette generation tasks.

LIST OF REFERENCES

LIST OF REFERENCES

- [1] <http://tinyurl.com/hvm34lr>.
- [2] <http://expandeddrablings.com/index.php/reddit-stats>.
- [3] <http://www.internetlivestats.com/twitter-statistics>.
- [4] <http://www.vmware.com>.
- [5] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The Anatomy of the Facebook Social Graph. *arXiv preprint arXiv:1111.4503*, 2011.
- [6] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the Spread of Influence Through a Social Network. *International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.
- [7] Mark Newman. The Structure of Scientific Collaboration Networks. *National Academy of Sciences*, pages 404–409, 2001.
- [8] Karsten Borgwardt, Cheng Soon Ong, Stefan Schnauer, S. V. N. Vishwanathan, Alex Smola, and Hans-Peter Kriegel. Protein Function Prediction via Graph Kernels. *Intelligent Systems in Molecular Biology*, pages 47–56, 2005.
- [9] Andrew Leach and Valerie Gillet. *An Introduction to Chemoinformatics*. Springer Science & Business Media, 2007.
- [10] Jeanne Ferrante, Karl Ottenstein, and Joe Warren. The Program Dependence Graph and its Use in Optimization. *Association for Computing Machinery Transactions on Programming Languages and Systems*, pages 319–349, 1987.
- [11] S. V. N. Vishwanathan, Nicol Schraudolph, Imre Risi Kondor, and Karsten Borgwardt. Graph Kernels. *Journal of Machine Learning Research*, pages 1–41. 2008.
- [12] Karsten Borgwardt. Graph Kernels. *PhD Thesis, Ludwig Maximilian University of Munich*, 2007.
- [13] Horst Bunke. Graph-based Tools for Data Mining and Machine Learning. *Machine Learning and Data Mining in Pattern Recognition*, pages 7–19, 2003.
- [14] David Haussler. Convolution Kernels on Discrete Structures. *Technical Report UCS-CRL-99-10, University of California Santa Cruz*, 1999.
- [15] Natasa Przulj. Biological Network Comparison Using Graphlet Degree Distribution. *European Conference on Computational Biology*, pages 177–183, 2006.

- [16] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient Graphlet Kernels for Large Graph Comparison. *International Conference on Artificial Intelligence and Statistics*, pages 488–495, 2009.
- [17] Nino Shervashidze and Karsten Borgwardt. Fast Subtree Kernels on Graphs. *Advances in Neural Information Processing Systems*, pages 1660–1668, 2010.
- [18] Gobinda Chowdhury. Natural Language Processing. *Annual Review of Information Science and Technology*, pages 51–89, 2003.
- [19] Paul Resnick and Hal Varian. Recommender Systems. *Communications of the Association for Computing Machinery*, pages 56–58, 1997.
- [20] Maeve Duggan and Aaron Smith. Online Adults are Reddit Users. *Pew Internet and American Life Project*, 2013.
- [21] <http://www.ted.com>.
- [22] <http://expandeddrablings.com/index.php/youtube-statistics>.
- [23] Gregory Conti. Countering Security Analyst and Network Administrator Overload Through Alert and Packet Visualization. *PhD Thesis, Georgia Institute of Technology*, 2006.
- [24] Nikil Wale, Ian Watson, and George Karypis. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. *Knowledge and Information Systems*, pages 347–375, 2008.
- [25] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An Open Source Software for Exploring and Manipulating Networks. *International Conference on Web and Social Media*, pages 361–362, 2009.
- [26] Hofmann, Thomas, Bernhard Scholkopf, and Alex Smola. Kernel Methods in Machine Learning. *The Annals of Statistics*, pages 1171–1220, 2008.
- [27] Tamas Horvath, Thomas Gartner, and Stefan Wrobel. Cyclic Pattern Kernels for Predictive Graph Mining. *International Conference on Knowledge Discovery and Data Mining*, pages 158–167, 2004.
- [28] Jan Ramon and Thomas Gartner. Expressivity Versus Efficiency of Graph Kernels. *First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74, 2003.
- [29] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized Kernels Between Labeled Graphs. *International Conference on Machine Learning*, pages 321–328, 2003.
- [30] Karsten Borgwardt and Hans-Peter Kriegel. Shortest-path Kernels on Graphs. *International Conference on Data Mining*, pages 74–81, 2005.
- [31] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten Borgwardt. Weisfeiler-Lehman Graph Kernels. *The Journal of Machine Learning Research*, pages 2539–2561, 2011.

- [32] Hui Lin and Jeff Bilmes. A Class of Submodular Functions for Document Summarization. *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510-520, 2011.
- [33] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A Combinatorial Strongly Polynomial Algorithm for Minimizing Submodular Functions. *Journal of the Association for Computing Machinery*, pages 761–777, 2001.
- [34] George Nemhauser, Laurence Wolsey, and Marshall Fisher. An Analysis of Approximations for Maximizing Submodular Set Functions. *Mathematical Programming*, pages 265–294, 1978.
- [35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.
- [36] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, pages 1137–1155, 2003.
- [37] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and Their Compositionality. *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.
- [38] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.
- [39] Aasa Feragen, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten Borgwardt. Scalable Kernels for Graphs with Continuous Attributes. *Advances in Neural Information Processing Systems*, pages 216–224, 2013.
- [40] Thomas Gartner, Peter Flach, and Stefan Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. *Annual Conference on Computational Learning Theory*, pages 129–143, 2003.
- [41] Fabrizio Costa and Kurt De Grave. Fast Neighborhood Subgraph Pairwise Distance Kernel. *International Conference on Machine Learning*, pages 255–262, 2010.
- [42] A. J. Smola and R. Kondor. Kernels and Regularization on Graphs. *Annual Conference on Computational Learning Theory*, pages 144–158, 2003.
- [43] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The Spectrum Kernel: A String Kernel for SVM Protein Classification. *Pacific Symposium on Biocomputing*, pages 564–575, 2002.
- [44] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online Learning of Social Representations. *International Conference on Knowledge Discovery and Data Mining*, pages 701-710, 2014.
- [45] Chang, Chih-Chung, and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *Association for Computing Machinery in Transactions on Intelligent Systems and Technology*, 2001.

- [46] Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan Shusterman, and Corwin Hansch. Structure-activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro Compounds Correlation with Molecular Orbital Energies and Hydrophobicity. *Journal of Medical Chemistry*, pages 786–797, 1991.
- [47] Hannu Toivonen, Ashwin Srinivasan, Ross D. King, Stefan Kramer, and Christoph Helma. Statistical Evaluation of the Predictive Toxicology Challenge. *Bioinformatics*, pages 1183–1193, 2003.
- [48] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs Over Time: Densification Laws, Shrinking Diameters and Possible Explanations. *International Conference on Knowledge Discovery and Data Mining*, pages 177–187, 2005.
- [49] Anshumali Shrivastava and Ping Li. A New Space for Comparing Graphs. *Advances in Social Networks Analysis and Mining*, pages 62–71, 2014.
- [50] Antonina Andreeva, Dave Howorth, Steven Brenner, Tim Hubbard, Cyrus Chothia, and Alexey Murzin. Scop Database in 2004: Refinements Integrate Structure and Sequence Family Data. *Nucleic Acids Research*, pages 226–229, 2004.
- [51] S. V. N. Vishwanathan, Karsten Borgwardt, Omri Guttman, and Alex Smola. Kernel Extrapolation. *Neurocomputing*, pages 721–729, 2006.
- [52] Taishin Kin, Tsuyoshi Kato, Koji Tsuda, and Kiyoshi Asai. Protein Classification via Kernel Matrix Completion. *Kernel Methods in Computational Biology*, pages 261–274, 2004.
- [53] Naomi Fox, Steven Brenner, and John-Marc Chandonia. Scope: Structural Classification of Proteins-extended, Integrating Scop and Astral Data and Classification of New Structures. *Nucleic Acids Research*, pages 304–309, 2014.
- [54] Brendan D McKay. Nauty User’s Guide. *Computer Science Department, Australian National University*, 2007.
- [55] Jaz Kandola, Thore Graepel, and John Shawe-Taylor. Reducing Kernel Matrix Diagonal Dominance Using Semi-definite Programming. *Annual Conference on Computational Learning Theory*, pages 288–302, 2003.
- [56] Yee Whye Teh. A Hierarchical Bayesian Language Model Based on Pitman-Yor Processes. *International Conference on Computational Linguistics*, pages 985–992, 2006.
- [57] Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *Association for Computing Machinery Transactions on Information Systems.*, pages 179–214, 2004.
- [58] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. *Cambridge University Press*, 2008.
- [59] Hermann Ney, Ute Essen, and Reinhard Kneser. On Structuring Probabilistic Dependencies in Stochastic Language Modeling. *Computer Speech and Language*, pages 1–38, 1994.
- [60] Reinhard Kneser and Hermann Ney. Improved Backing-off for M-gram Language Modeling. *International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184, 1995.

- [61] Sharon Goldwater, Thomas Griffiths, and Mark Johnson. Producing Power-law Distributions and Damping Word Frequencies with Two-stage Language Models. *Journal of Machine Learning Research*, pages 2335–2382, 2011.
- [62] Sharon Goldwater, Tom Griffiths, and Mark Johnson. Interpolating Between Types and Tokens by Estimating Power-law Generators. *Advances in Neural Information Processing Systems*, pages 459–466, 2006.
- [63] Jim Pitman and Marc Yor. The Two-parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator. *Annals of Probability*, pages 855–900, 1997.
- [64] Danilo Croce, Alessandro Moschitti, and Roberto Basili. Structured Lexical Similarity via Convolution Kernels on Dependency Trees. *Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046, 2011.
- [65] Aliaksei Severyn and Alessandro Moschitti. Fast Support Vector Machines for Convolution Tree Kernels. *Data Mining and Knowledge Discovery*, pages 325–357, 2012.
- [66] Marion Neumann, Roman Garnett, Plinio Moreno, Novi Patricia, and Kristian Kersting. Propagation Kernels for Partially Labeled Graphs. *Workshop on Mining and Learning with Graphs at International Conference on Machine Learning*, 2012.
- [67] George Nemhauser, Laurence Wolsey, and Marshall Fisher. An Analysis of the Approximations for Maximizing Submodular Set Functions. *Mathematical Programming*, pages 265–294, 1978.
- [68] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning Down the Noise in the Blogosphere. *Conference on Knowledge Discovery and Data Mining*, pages 289–298, 2009.
- [69] Hui Lin and Jeff Bilmes. A Class of Submodular Functions for Document Summarization. *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520, 2011.
- [70] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective Outbreak Detection in Networks. *Conference on Knowledge Discovery and Data Mining*, pages 420–429, 2007.
- [71] David Blei, Andrew Y. Ng, and Michael Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.
- [72] Peter Stobbe and Andreas Krause. Efficient Minimization of Decomposable Submodular Functions. *Advances in Neural Information Processing Systems*, pages 2208–2216, 2010.
- [73] David Aldous. *Ecole D’ete de Probabilites de Saint-flour*. 1985.
- [74] Léon Bottou. Large-scale Machine Learning with Stochastic Gradient Descent. *International Conference on Computational Statistics*, pages 177–186, 2010.
- [75] Jyrki Kivinen and Manfred Warmuth. Exponentiated Gradient Versus Gradient Descent for Linear Predictors. *Technical Report UCSC-CRL-94-16, University of California, Santa Cruz*, 1994.

- [76] Manfred Warmuth. The Blessing and the Curse of the Multiplicative Updates. *Discovery Science*, page 382, 2010.
- [77] Solomon Kullback and Richard Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, pages 79–86, 1951.
- [78] Olivier Bousquet and Manfred Warmuth. Tracking a Small Set of Experts by Mixing Past Posteriors. *Annual Conference on Computational Learning Theory*, pages 31–47, 2001.
- [79] Amr Ahmed, Choon Hui Teo, S. V. N. Vishwanathan, and Alex Smola. Fair and Balanced: Learning to Present News Stories. *Association of Computing Machinery International Conference of Web Science and Data Mining*, pages 333-342, 2012.
- [80] Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. What’s in a Name? Understanding the Interplay Between Titles, Content, and Communities in Social Media. *International Conference on Web and Social Media*, 2013.
- [81] Philipp Singer, Fabian Flöck, Clemens Meinhart, Elias Zeitfogel, and Markus Strohmaier. Evolution of Reddit: From the Front Page of the Internet to a Self-referential Community? *International Conference on World Wide Web Companion*, pages 517–522, 2014.
- [82] Tim Weninger, Xihao Avi Zhu, and Jiawei Han. An Exploration of Discussion Threads in Social News Sites: A Case Study of the Reddit Community. *Advances in Social Networks Analysis and Mining*, pages 579–583, 2013.
- [83] Pinar Yanardag and SVN Vishwanathan. Deep Graph Kernels. *International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.
- [84] Matthew Hoffman, Francis Bach, and David Blei Online Learning for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, pages 856–864, 2010.
- [85] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. *Workshop on New Challenges for Natural Language Processing Frameworks*, pages 45–50, 2010.
- [86] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The Budgeted Maximum Coverage Problem. *Information Processing Letters*, pages 39–45, 1999.
- [87] Hui Lin and Jeff Bilmes. Multi-document Summarization via Budgeted Maximization of Submodular Functions. *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920, 2010.
- [88] Jaime Carbonell and Jade Goldstein. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. *Annual international Association for Computing Machinery Conference on Research and Development in Information Retrieval*, pages 335–336, 1998.
- [89] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. Multi-document Summarization by Sentence Extraction. *Workshop on Automatic Summarization at Association for Computational Linguistics*, pages 40–48, 2000.

- [90] Nikolaos Pappas and Andrei Popescu-Belis. Combining Content with User Preferences for TED Lecture Recommendation. *International Workshop on Content-Based Multimedia Indexing*, pages 47–52, 2013.
- [91] Nikolaos Pappas and Andrei Popescu-Belis. Explaining the Stars: Weighted Multiple-instance Learning for Aspect-based Sentiment Analysis. *Conference on Empirical Methods in Natural Language Processing*, 2014.
- [92] José Luis Redondo García, Mariella Sabatino, Pasquale Lisena, and Raphaël Troncy. Detecting Hot Spots in Web Videos. *International Conference on Posters & Demonstrations Track*, pages 141–144, 2014.
- [93] Rui Correia, Nuno Mamede, Jorge Baptista, and Maxine Eskenazi. Using the Crowd to Annotate Metadiscursive Acts. *Special Interest Group on Computational Semantics*, page 102, 2014.
- [94] Fajri Koto, Sakriani Sakti, Graham Neubig, Tomoki Toda, Mirna Adriani, and Satoshi Nakamura. Memorable Spoken Quote Corpora of TED Public Speaking. *Coordination and Standardization of Speech Databases and Assessment Techniques*, pages 1–4, 2014.
- [95] Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. Video Digests: A Browsable, Skimmable Format for Informational Lecture Videos. *Annual Association of Computer Machinery Symposium on User Interface Software and Technology*, pages 573–582, 2014.
- [96] Andrew Tsou, Mike Thelwall, Philippe Mongeon, and Cassidy Sugimoto. A Community of Curious Souls: An Analysis of Commenting Behavior on TED Talks Videos. *Public Library of Science*, 2014.
- [97] Chin-Yew Lin. Rouge: A Package for Automatic Evaluation of Summaries. *Association of Computer Linguistics Workshop on Text Summarization*, 2004.
- [98] Eric Jones, Travis Oliphant, and Pearu Peterson. Scipy: Open Source Scientific Tools for Python. 2014.
- [99] Ajay Gulati, Ganesha Shanmuganathan, Anne Holler, Carl Waldspurger, Minwen Ji, and Xiaoyun Zhu. VMware Distributed Resource Management: Design, Implementation, and Lessons Learned. *VMware Technical Journal*, 2012.
- [100] <http://tinyurl.com/vmware-external-link1>.
- [101] <http://tinyurl.com/vmware-external-link2>.
- [102] Dragos Ionescu and Rean Griffith. The Wisdom of Virtual Crowds: Mining Datacenter Telemetry to Collaboratively Debug Performance. *Annual Symposium on Cloud Computing*, pages 32-33, 2013.
- [103] Andreas Krause and Daniel Golovin. Submodular Function Maximization. *Tractability: Practical Approaches to Hard Problems*, 2012
- [104] Michael Locasto, Stelios Sidiroglou, and Angelos Keromytis. Application Communities: Using Monoculture for Dependability. *Conference on Hot Topics in System Dependability*, pages 7-9, 2005.

- [105] Adam Oliner, Anand Iyer, Eemil Lagerspetz, Sasu Tarkoma, and Ion Stoica. Collaborative Energy Debugging for Mobile Devices. *Conference on Hot Topics in System Dependability*, pages 6-9, 2012.
- [106] Eno Thereska, Bjoern Doebel, Alice X. Zheng, and Peter Nobel. Practical Performance Models For Complex, Popular Applications. *Association for Computing Machinery International Conference on Measurement and Modeling of Computer Systems*, pages 1–12, 2010.
- [107] Pinar Yanardag and SVN Vishwanathan. A Structural Smoothing Framework for Robust Graph Comparison. *Advances of Neural Information Processing Systems*, pages 2125-2133, 2015.
- [108] Uriel Feige. A Threshold of LN-N for Approximating Set Cover. *Journal of the Association for Computing Machinery*, pages 634–652, 1998.
- [109] Jason Beard. *The Principles of Beautiful Web Design*. Sitepoint, 2010.
- [110] Sharon Lin and Pat Hanrahan. Modeling How People Extract Color Themes from Images. *Association for Computing Machinery Conference on Human Factors in Computing Systems*, pages 3101–3110, 2013.
- [111] Ali Jahanian, SVN Vishwanathan, and Jan Allebach. Autonomous Color Theme Extraction from Images Using Saliency. *International Society for Optics and Photonic Electronic Imaging*, 2015.
- [112] Michael Orchard and Charles Bouman. Color Quantization of Images. *Transactions on Signal Processing*, pages 2677–2690, 1991.
- [113] Martin Solli and Reiner Lenz. Color Semantics for Image Indexing. *Conference on Colour in Graphics, Imaging, and Vision*, pages 353–358, 2010.
- [114] Christian Wengert, Matthijs Douze, and Hervé Jégou. Bag-of-colors for Improved Image Search. *Association for Computing Machinery International Conference on Multimedia*, pages 1437–1440, 2011.
- [115] Alaa E Abdel-Hakim and Aly A Farag. Csift: A Sift Descriptor with Color Invariant Characteristics. *Computer Vision and Pattern Recognition Conference*, pages 1978–1983, 2006.
- [116] Roderick McDonald and Kenneth Smith. CIE94: A New Colour-difference Formula. *Journal of the Society of Dyers and Colourists*, pages 376–379, 1995.
- [117] Naila Murray, Sandra Skaff, Luca Marchesotti, and Florent Perronnin. Toward Automatic and Flexible Concept Transfer. *Computers and Graphics*, pages 622–634, 2012.
- [118] Masataka Tokumaru, Noriaki Muranaka, and Shigeru Imanishi. Color Design Support System Considering Color Harmony. *International Conference on Fuzzy Systems*, pages 378–383, 2002.
- [119] Yutaka Matsuda. *Color Design*. Asakura Shoten, 1995.
- [120] Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. Color Harmonization. *Association for Computing Machinery Transactions on Graphics*, pages 624–630, 2006.

- [121] Lujin Wang, Joachim Giesen, Kevin McDonnell, Peter Zolliker, and Klaus Mueller. Color Design for Illustrative Visualization. *Visualization and Computer Graphics*, pages 1739–1754, 2008.
- [122] Peter O’Donovan, Aseem Agarwala, and Aaron Hertzmann. Color Compatibility from Large Datasets. *Association for Computing Machinery Transactions on Graphics*, page 63, 2011.

VITA

VITA

Pinar Yanardag Delul hails from Istanbul, Turkey. She received her MSc in Computer Engineering from Bogazici University, and was awarded a Fulbright fellowship to pursue her PhD at Purdue University, Department of Computer Science. Pinar worked for TUBITAK UEKAE (The Turkish Research Institute of Electronics and Cryptology) for two years as a Security Team Leader and FLOSS developer for the Pardus GNU/Linux project, and as a mentor for Google Summer of Code. She has also worked at VMware's Distributed Resource Scheduler team (two summers) and Amazon's Personalization team (one summer) as a machine learning scientist.