WALKING FORWARD AND BACKWARD: TOWARDS GRAPH-BASED SEARCHING AND MINING

BY

YUAN FANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Associate Professor Kevin Chen-Chuan Chang, Chair
Professor Jiawei Han
Professor Chengxiang Zhai
Associate Professor Shonali Krishnaswamy, Institute for Infocomm Research

Abstract

Graphs are powerful vehicles to represent data objects that are interconnecting or interacting with each other. We explore random walks on various kinds of graph to address different searching and mining scenarios. This dissertation focuses on two symmetric forms of random walk called the *forward* walk and *backward* walk, which can be applied to enrich three key tasks in searching and mining, namely, extraction, ranking and classification, in novel ways. More specifically, we enhance extraction with the *metrics* of probabilistic precision and recall, ranking with the *senses* of importance and specificity, and classification with *heterogeneous contexts* in terms of relationship type and confidence level. We further study the underpinning principle of random walks on a graph, which is often known as the smoothness assumption. We argue that smoothness is a *pointwise* property and requires *probabilistic* modeling. Thus, we propose a new framework to define pointwise smoothness probabilistically on a graph, which unifies two different "modes" of smoothness corresponding to the forward and backward random walks, respectively. Finally, our graph-based random walk solutions have consistently demonstrated promising empirical results for a wide range of searching and mining applications.

Acknowledgements

Since I first started my career as a graduate student at University of Illinois at Urbana-Champaign, I have been indebted to numerous colleagues and friends who have always been inspirational and helpful.

First of all, I would like to express my most sincere gratitude and appreciation to my advisor Professor Kevin Chen-Chuan Chang, who was a great mentor in preparing me for a research career in computer science. Throughout my five years at the University, I was motivated, shaped and benefited by Prof. Chang's perspectives, including his pursuit of innovation in science and engineering, his insight into the big picture of the field, as well as his attention to detail in conducting research. I am especially thankful for not only his guidance and support, but also his flexibility in working with me.

I would also like to thank my doctoral committee members, including Prof. Jiawei Han, Prof. Chengxiang Zhai and Prof. Shonali Krishnaswamy. During our discussions, they brought up many constructive and mind-opening suggestions, which were critical and valuable to my dissertation work as well as future research directions.

Singapore's Agency for Science, Technology and Research (A*STAR) provided immense financial support for my entire Ph.D. candidature. The University's Advanced Digital Sciences Center (ADSC) also accommodated many of my financial and academic needs. Special thanks go to Dr. See-Kiong Ng from A*STAR, as well as Prof. Marianne Winslett and Prof. Douglas Jones from ADSC, who made my training as smooth as possible.

I wish to thank all of my fellow team members led by Prof. Chang. I consider myself extremely fortunate to be able to work with brilliant colleagues including Dr. Rui Li,

Dr. Mianwei Zhou and Dr. Vincent Zheng. I also greatly appreciate other team members, as well as colleagues and friends from the Database and Information Systems Laboratory, the Department of Computer Science, A*STAR and ADSC.

I also owe my gratitude to many collaborators who not only contributed to my dissertation research, but also broadened my horizon. The teamwork with Dr. Paul Hsu, Dr. Ming-Wei Chang and Dr. Kuansan Wang during my two internships at Microsoft Research sparked quite a few innovative ideas to explore in an industrial setting. The collaboration with Prof. Hady Lauw from Singapore Management University and Dr. Fanwei Zhu from Zhejiang University was fulfilling and delightful.

This journey could not have been possible without the help from our administrative staff, including Mary Beth Kelley, Rhonda McElroy and Donna Coleman from the Department of Computer Science, Siti Zuraidah and Aizhen He from A*STAR, Pauline Wee and Donna Foley from ADSC, among many others. Their professional assistance in one way or another is very much appreciated.

Last but not least, I am grateful to my parents and wife. In particular, my wife Dr. Sha Huang, who was a graduate student herself, was especially supportive and understanding. Their unwavering love and encouragement kept me moving forward. This dissertation is dedicated to them.

Table of Contents

List of Figures . . . . . **vi**

Chapter 1 Introduction **1**

Chapter 2 Extraction with Precision and Recall **11**

Chapter 3 Ranking with Importance and Specificity **42**

Chapter 4 Classification with Heterogeneous Contexts **79**

Chapter 5 Graph-based Smoothness Framework **111**

Chapter 6 Conclusion: Unification and Beyond **136**

References **140**

Appendix A Proofs of Propositions **150**

List of Figures

1.1 Searching and mining problems for a bibliographic knowledge base. 4

1.2 Toy graph for relation extraction. 5

1.3 Toy graph for object classification. 6

1.4 Toy graph for object ranking. 6

2.1 The dual problems of pattern search and tuple extraction. 13

2.2 Running example for relation extraction. 17

2.3 Tuple-pattern associations and graph. 22

2.4 Inference rules for probabilistic precision and recall. 25

2.5 PRDualRank algorithm: Ranking by PR Duality. 28

2.6 Interpreting probabilistic precision and recall as random walks. 29

2.7 Partially materialized tuple-pattern graph. 29

2.8 Snapshots of partial materialization. 32

2.9 PatternSearch algorithm: Finding good patterns. 33

2.10 TupleExtration algorithm: Extracting relevant tuples. 34

2.11 Target relations. 35

2.12 Comparison of precision and recall. 38

2.13 Comparison of optimal F-scores. 38

2.14 Comparison of execution time in seconds. 39

2.15 Using top K patterns by precision versus by recall. 40

2.16 Effect of parameters on PRDualRank. 40

3.1 Top venues for "spatio temporal data." . . . 43
3.2 Toy graph for a bibliographic network. . . . 46
3.3 Illustration of round trips. . . . 52
3.4 RoundTripRank for the bibliographic toy graph. . . . 53
3.5 2SBound algorithm: Two-Stage Bounding for RoundTripRank. . . . 58
3.6 NDCG@K of RoundTripRank and mono-sensed baselines. . . . 70
3.7 Ranking venues for "spatio temporal data." . . . 71
3.8 Ranking venues for "semantic web." . . . 71
3.9 Effect of the specificity bias. . . . 72
3.10 NDCG@K of RoundTripRank+ and existing dual-sensed baselines. . . . 73
3.11 NDCG@5 of RoundTripRank+ and customized dual-sensed baselines. . . . 74
3.12 Time and quality of 2SBound on BibNet under varying slacks. . . . 76
3.13 Active set size and query time on growing graphs. . . . 77
3.14 Rate of growth in snapshot, active set and query time. . . . 77

4.1 Toy graph for classification of shopping queries. . . . 84
4.2 Examples of different types of neighbor in graph regularization. . . . 101
4.3 Precision-recall plot for different relationship types. . . . 102
4.4 Accuracy comparison for different relationship types. . . . 102
4.5 Result analysis for queries with and without clicks. . . . 104
4.6 Accuracy comparison with regression node model. . . . 104
4.7 Accuracy with different confidence initializations. . . . 105
4.8 Accuracy versus number of queries. . . . 106
4.9 Accuracy versus the number of clickthroughs. . . . 108
4.10 Convergence of accuracy and minimum error. . . . 109
4.11 Execution time versus number of total queries. . . . 110

5.1 Toy problem on the two-spiral dataset. . . . 126

5.2 Summary of the datasets. 129
5.3 Performance comparison of SSL methods. 131
5.4 Effect of incorporating class priors in prediction. 133
5.5 Effect of unlabeled data. 134
5.6 Effect of graph perturbation. 135

Chapter 1

Introduction

We live in a world with the unprecedented availability of data. Graphs are powerful representations to capture not only individual data objects, but also their interconnections or interactions between one another. Random walk is an effective means to exploit the inherent structures in a graph, which is crucial to solving common searching and mining problems, such as *extraction*, *ranking* and *classification*, over data organized as a graph. In this dissertation, we study graph-based approaches for these problems, enriching them in various novel ways, which correspond to different forms of random walk. We also study the underlying principle governing these solutions, unified into a graph-based smoothness framework.

In this chapter, we first present an overview of our work in Sections 1.1–1.3, including a brief review of graph and random walk, novel graph-based approaches to various searching and mining problems, as well as an in-depth examination of their fundamental principle. Next, we outline the organization of the dissertation in Section 1.4.

1.1 Background on Graph and Random Walk

Different data objects in many domains often interconnect or interact with each other. Such inter-object interactions form a myriad of gigantic and complex networks, including for instance protein-protein interaction networks [86], highway transportation systems [62], the World Wide Web [19], bibliographic networks [49], and social networks [1, 40], with profound significance in scientific, engineering and social applications. Not surprisingly, graphs are natural vehicles to model data objects living on a network, for they are capable of concisely

representing not only the objects themselves but also the relationships between them, which traditional "flat" representations of data often fall short of. One example among the most influential graph-based applications is the famous PageRank algorithm [81], treating the webpages as an interconnected link graph instead of a simple flat collection. We formally define a graph in the following.

DEFINITION 1.1 (GRAPH). A *weighted* and *directed graph* $G = (V, E)$ comprises a set of nodes (or vertices) $V = \{1, \ldots, |V|\}$, and a set of directed edges E. Each directed edge $(i, j) \in E$ is an ordered pair of nodes such that $i, j \in V$, and is associated with a weight $W_{ij} \in [0, \infty)$, where W is an $|V| \times |V|$ weight matrix.

Although our definition specifies a weighted and directed graph, it subsumes an unweighted or undirected graph as well. For an *unweighted* graph, we simply have uniform edge weights, *i.e.*, $W_{ij} = 1, \forall ij$. For an *undirected* graph, we have symmetric directed edges, *i.e.*, $(i, j) \in E \Leftrightarrow (j, i) \in E$ and $W_{ij} = W_{ji}, \forall ij$. Moreover, by convention, $W_{ij} = 0$ if and only if $(i, j) \notin E$.

Note that each vertex in a graph represents an object, whereas each edge embeds some relationship between the two connecting objects. To address searching and mining problems, a key challenge lies in relating objects through their relationships or interactions. While earlier studies have employed simpler graph theoretic measures such as degree and betweenness centrality [46], more recent efforts have shifted to spectral techniques [17], and the related *random walk* or propagation-based methods [81, 67], for their ability to ultimately capture not only local interactions, but also global structures of the graph.

In this dissertation, we study random walk-based approaches and their principles. A random walk can be intuitively described by the behavior of a random surfer. Consider a surfer moving on a graph, which is initially residing at some node v. As her first step, she randomly moves to a neighboring node of v. In the subsequent steps, she repeats this process, randomly moving to a neighboring node of her current node. The sequence of nodes

visited by the random surfer forms an instance of the random walk. In fact, a random walk on a graph is a finite Markov chain, which is governed by V as its state space and P as its transition matrix [76], as formally defined below.

DEFINITION 1.2 (RANDOM WALK ON GRAPH). Given a graph $G = (V, E)$ and some starting node $v \in V$, a *random walk* is a sequence of random nodes $(V_t : t = 0, 1, \ldots)$ where $V_0 = v$, and the value of V_{t+1} depends on that of V_t according to the stochastic matrix P such that $p(V_{t+1} = j | V_t = i) = P_{ij}$. P is defined as follows:

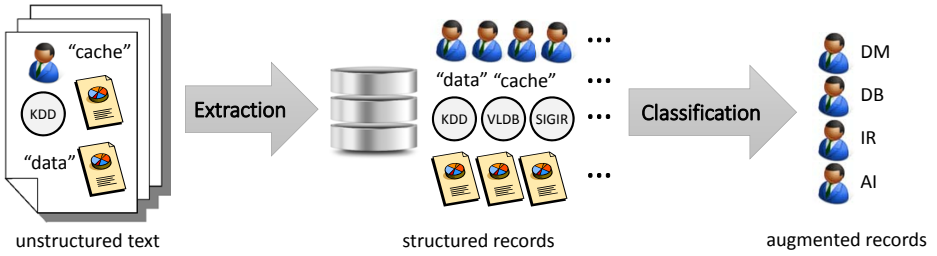
$$P_{ij} = \begin{cases} \frac{W_{ij}}{\sum_{k \in V} W_{ik}} & (i,j) \in E \\ 0 & \text{else.} \end{cases} \tag{1.1}$$

1.2 Graph-based Searching and Mining Problems

Data as a graph motivate a suite of exciting scenarios, calling for effective and efficient searching and mining techniques. Most of these scenarios could be cast into one of the typical problems in searching and mining. In this dissertation, we focus on three key problems, namely, *extraction*, *ranking*, and *classification*, which are illustrated in Figure 1.1 by the application scenarios involving a bibliographic knowledge base such as DBLP [31]. First, extraction and classification can help build and scale up a knowledge base by automatically extracting structured records from unstructured data, and augmenting the records with additional attributes. Second, ranking is a useful tool to answer end users' information needs by producing a ranked list of results with respect to some query. We further elaborate the three tasks in the following.

Relation extraction: *to extract tuples of a given relation from a corpus such as a text database or the Web.* To build a bibliographic knowledge base, we would be interested in finding the tuples of the form ($\langle$person$\rangle$, $\langle$paper$\rangle$) for the relation `author-of`. Extraction of

(a) Building knowledge base



(b) Using knowledge base

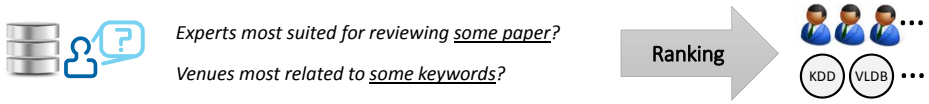


Figure 1.1: Searching and mining problems for a bibliographic knowledge base.

such tuples help build the bibliographic knowledge base automatically. As another example, imagine we are building a geographical information system, and intend to extract the tuples of the form (⟨city⟩, ⟨country⟩) for the relation `capital-city-of`, such as (Tokyo, Japan) and (Ottawa, Canada). In contrast, for the relation `largest-city-of`, a different set of tuples are expected, such as (Tokyo, Japan) and (Toronto, Canada).

While supervised natural language processing techniques [70, 42] have achieved considerable success in information extraction, pattern-based techniques [18, 4, 103] have also become popular as they generally require less supervision. Pattern-based approaches also naturally enable us to construct a graph, given that different patterns and their matching tuples can be interlinked to form a network, such as the one depicted in Figure 1.2. Starting from some sample patterns or tuples (known as the "seeds"), we could ultimately discover new ones though the links in such a network. For instance, given a seed tuple (Ottawa, Canada) belonging to the relation `capital-city-of`, we can also extract (Tokyo, Japan) for the same relation based on their common patterns "⟨city⟩ is the capital city of ⟨country⟩" and "⟨country⟩'s capital, ⟨city⟩."

Object classification: *to classify data objects into one of the predefined categories.* The classifications can conveniently augment the bibliographic knowledge base, such as assigning

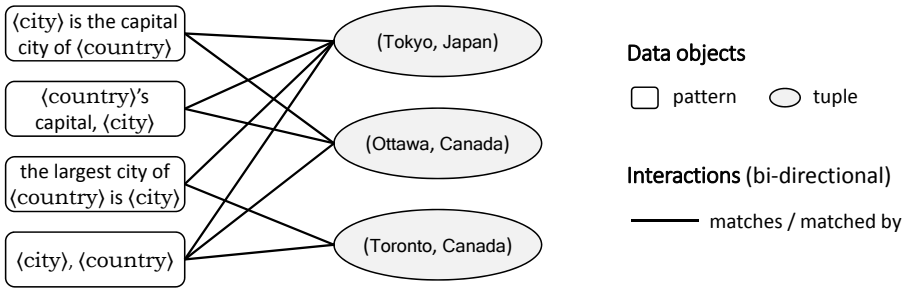


Figure 1.2: Toy graph for relation extraction.

each researcher to a primary field, which may not be directly available through extraction only. Alternatively, suppose we are building an e-commerce website such as Amazon [6], where users compose queries to search for products online. These queries are in fact data objects created by users, which can be classified into a product category (for instance mapping query "`canon sd1000`" to category `camera`). This enables redirecting the user to the appropriate product vertical for an improved shopping experience.

Inspired by the recent developments in graph-based regularization, classification accuracy can be substantially improved under the key assumption of smoothness: similar objects are likely to belong to similar categories [113, 107]. In other words, on a graph where similar objects are connected as neighbors, the classification of an object can be "helped" or regularized by its neighbors on the graph. To construct this graph, while the objects naturally constitute the set of nodes, the edges shall connect similar objects. Taken query classification as an example, we can link queries sharing one or more words (similar by lexicon), or queries with with user clicks landing on the same page (similar by click), or queries issued in the same user session (similar by session), as illustrated in Figure 1.3.

Object ranking: *to return a ranked list of data objects by their semantic relatedness to some given query object.* Given various academic objects like authors, papers, keyword terms and publication venues in a bibliographic knowledge base, some interesting ranking questions could be raised: (a) Given a paper as the query, who are the experts (authors) most suited for peer reviewing it? (b) Given some keyword terms as the query, what are the

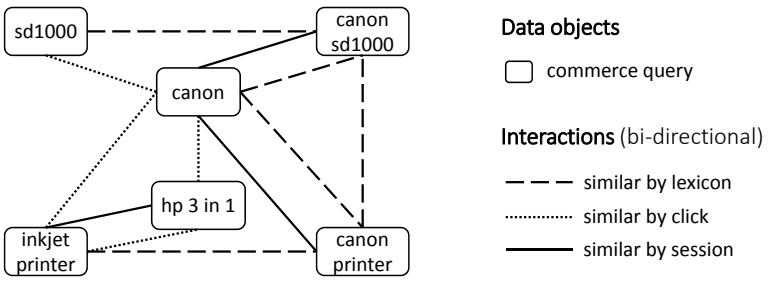


Figure 1.3: Toy graph for object classification.

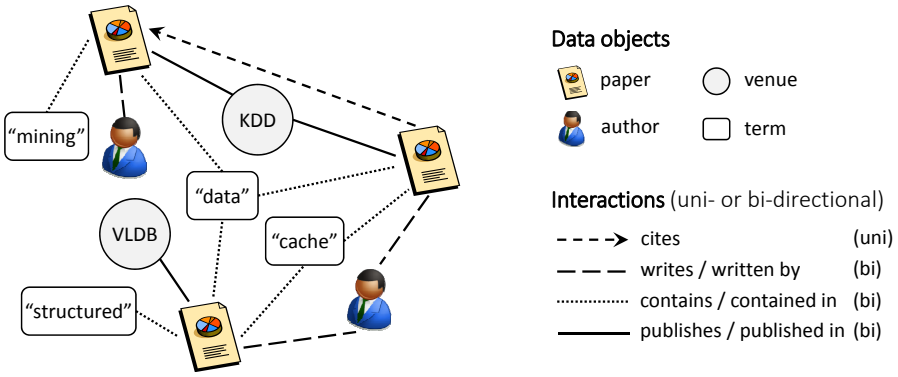


Figure 1.4: Toy graph for object ranking.

most matching venues? (c) Given an author as the query, which are the papers most related to his or her research? These questions could be answered by a ranking over the objects in terms of their semantic relatedness to the query.

In particular, the academic objects in the bibliographic example can be naturally organized into a graph, as depicted in Figure 1.4. Given that each edge encodes certain inter-object interactions, through these edges it is possible to quantify the semantic relatedness between different objects [10, 80, 23]. Subsequently, we can rank objects by their semantic relatedness to the query object.

Proposed agenda. Although many graph-based techniques have been developed for extraction, ranking and classification, in this dissertation we enrich them with various novel aspects overlooked by existing methods. Interestingly, all of our enriching solutions can be expressed by two symmetric and "opposite" forms of random walk, called the *forward* and *backward* random walks [2].

First, most applications of relation extraction fundamentally seek to optimize the metrics of precision or recall with respect to a ground truth set. However, these two objective metrics are not explicitly modeled by previous pattern-based extraction methods [18, 4]. Thus, we propose to support relation extraction with the *objective metrics* of probabilistic *precision* and *recall*, which improves not only the accuracy but also the interpretability of the extracted patterns and tuples. As the inference process turns out, probabilistic precision and recall can be captured by the backward and forward random walks, respectively.

Second, a ranking in terms of a single sense of semantic relatedness is often insufficient, given that different users often have disparate information needs. Unfortunately, most existing approaches only focus on the sense of importance (*i.e.*, authoritativeness) [81, 10, 80, 23] or a vague sense of "proximity" without a finer interpretation [1, 58, 68, 102]. Only until recently the complementary sense of specificity [56] emerged in a few heuristic forms. Thus, we propose to generalize the notion of importance to also capture specificity, and further integrate the two senses in one coherent process to achieve a customizable balance between them. That is, we diversify ranking with the complementary *senses* of *importance* and *specificity*, as well as their customizable trade-off. Interestingly, importance and specificity also correspond to the forward and backward random walks on the graph, respectively.

Third, while the classification of an object can be influenced or regularized by its contexts (*i.e.*, its similar objects), the contexts may be heterogeneous and should not be treated uniformly. Although some existing methods have considered some form of heterogeneous relationship types between neighboring objects on the graph [61, 100, 60], little research has been devoted to a second dimension of heterogeneity—the varying confidence levels of the contexts. Thus, we propose to guide the classification of an object by discriminating its *heterogeneous contexts*, in terms of both the *relationship type* and the *confidence level*. In particular, our framework, which incorporates both dimensions of heterogeneity, is a special form of the backward random walk.

Apart from these three tasks, other major tasks include object resolution [7] and clus-

tering [45] on the graph. Such tasks are the so called object-centric problems [47]. There also exist edge-centric problems, in particular link prediction [75]. Potentially, our random walk-based techniques can be applied to these tasks as well. However, we also note that there are other graph-based problem formulations focused on the subgraph level [57], or even on the graph level (within a collection of graphs) [66]. These formulations are different from the node or edge-level interactions explored in this dissertation, and they beyond the scope of this dissertation.

1.3 Principle behind Graph-based Approaches

While random walks are capable of solving various graph-based searching and mining problems, as the second part of this dissertation, we will investigate the fundamental principle governing and unifying the forward and backward random walks on graphs. In particular, we attempt to answer the following questions.

First, *what is the underpinning mechanism that random walks boil down to*? At the core of random walks on graphs lies the notion of *smoothness* [111, 24], which is the principle governing the interactions between neighboring nodes on the graph. Simply put, smoothness means that two close nodes on the graph are likely to behave in the same way. Random walks in fact implicitly exploit this assumption to solve various graph-based problems. Thus, it is important to examine smoothness on a graph, *i.e.*, the explicit form of interaction between nodes. In particular, we study the principle of smoothness in the classic setting of graph-based semi-supervised learning, which can be extended to other searching and mining settings in this dissertation.

Second, *what are the requirements of realizing smoothness on a graph*? We identify that smoothness is *pointwise* and *probabilistic*. It is inherently a property that occurs "everywhere" on the graph and thus applies to every node, which we call the pointwise nature of smoothness. Furthermore, it is only meaningful to model the pointwise nature of smoothness

probabilistically. While pointwise smoothness has been studied in non-graph based settings [84, 71, 96], previous graph-based methods [113, 107, 14, 98], whether probabilistic or not, only support aggregate, but not pointwise, smoothness.

Third, *is there a unifying framework to capture different "modes" of smoothness?* As we will see, different forms of random walk are employed to address searching and mining of varying needs. However, they are still underpinned by the same principle of smoothness, although there might be different modes of smoothness. Interestingly, these different modes can be unified under a smoothness framework consisting of the same foundational submodels of *data closeness* and *label coupling*. It turns out that different modes of smoothness are simply the results of different probabilistic formulations under the same framework.

1.4 Organization

The rest of this dissertation is organized into the following chapters, where Chapters 2–4 cover graph-based solutions to the three key problems in searching and mining, and Chapter 5 explores the principle of smoothness to unify graph-based approaches.

In Chapter 2, we propose to support pattern-based relation extraction with the objective metrics of probabilistic precision and recall [37]. In particular, our pattern-based approach entails a graph structure based on a network of interconnected patterns. Building upon this graph, we realize the metrics of precision and recall in a probabilistic manner, which can be interpreted as two symmetric forms of random walks on the graph. Another challenge lies in the incompleteness of the graph—given an enormous corpus such as the Web, we can only partially materialize the graph, which inevitably calls for a sampling guideline.

In Chapter 3, we propose to diversify the ranking of interconnected data objects with semantic senses, particularly in terms of importance and specificity, as well as their customizable trade-off [38]. It turns out that importance and specificity can be respectively captured by two symmetric forms of random walk in "opposite" directions, which can be

further unified into a coherent "round trip" random walk. In fact, the two forms of random walks are common techniques employed in a wide range of scenarios, including those studied in other chapters. Lastly, to enable online processing in real time, we also discuss the efficient and scalable implementation of both forms of random walk as well as their integration as a round trip.

In Chapter 4, we propose to guide object classification with *heterogeneous contexts*, in terms of the relationship types and confidence levels [41]. While most previous studies focus on heterogeneous graphs focus on the first dimension of relationship types, we address both dimensions of heterogeneity through modeling every node (or object) on the graph using a Dirichlet prior. Moreover, we also propose to learn parameters for the relationship types with very limited labeled data, whereas in many previous approaches, such parameters are often manually selected. Interestingly, our classification model can be perceived as a special form of random walk on the object graph.

In Chapter 5, we investigate the principle underpinning graph-based random walk approaches, which boil down to the concept of *smoothness* between neighboring nodes on the graph [39]. In particular, using a graph-based semi-supervised setting, we study two complementary dimensions of smoothness: its *pointwise* nature and *probabilistic* modeling. While no existing graph-based work exploits them in conjunction, we encompass both in a novel and unifying framework for realizing different modes of pointwise smoothness on a graph in a probabilistic manner.

Lastly, Chapter 6 concludes this dissertation and discusses some future research directions to improve graph-based searching and mining.

Chapter 2

Extraction with Precision and Recall

In this chapter, we propose to enrich the task of extraction with the metrics of probabilistic *precision* and *recall*. We exemplify this enrichment by the problem of *relation extraction*, where the aim is to extract entity tuples of a given relation from different webpages on the Web. In particular, recent pattern-based approaches for relation extraction enable us to capture the network of interconnected patterns and tuples as a graph. Building upon this graph, we formally define precision and recall in a probabilistic manner, which can be captured by backward and forward random walks on the graph, respectively. Furthermore, as it is only feasible to obtain an incomplete graph due to the scale of the Web, we also propose some guiding principles to handle the partial materialization of the graph.

2.1 Introduction

The Web has evolved into our ultimate information repository. To unleash data inside the unstructured texts online, we are facing the immense challenge of information extraction (IE), to convert unstructured contents to structured information. While the barrier is daunting, there also come novel opportunities. The massive contents on the Internet offer both *redundancy* and *diversity*, both of which inspire new techniques. In particular, to scale up to the Web, IE has recently moved toward search-based, which is consistent with the emergence of Web-based Query Answering (QA), by sending keyword queries (*e.g.*, "capital city") to a general search engine, retrieving top pages, and processing them to extract desired information (*e.g.*, the relation `capital-city-of`).

This search-and-extract approach has been recently studied in several Web-based IE [35, 5] and QA [69, 28, 34] systems, which reveal consistent interesting observations:

- *Size is good: Redundancy has fundamentally changed the problem* from document-centered to corpus-centered. The size of the Web ensures that, when the corpus is considered as a whole, the "voting" effect beyond each individual document will contribute significantly to surface good results.

- *Diversity is good: Simple patterns can outperform* sophisticated NLP-based techniques. The diversity of the Web ensures that simple patterns, while inexpensive to execute, will match some documents to extract good results.

The insights amount to the approach of *pattern-based relation extraction*, to extract a relation R of tuples t by matching some textual patterns p. For instance, to extract tuples of the form ($\langle$city$\rangle$, $\langle$country$\rangle$) for the relation `capital-city-of`, we can use pattern p_1 = "$\langle$city$\rangle$ is the capital ... of $\langle$country$\rangle$" or p_2 = "$\langle$city$\rangle$ is ... city of $\langle$country$\rangle$", both of which match the text "Paris is the capital city of France" and extract the tuple (Paris, France). Note that we use $\langle$E$\rangle$ to denote an entity type, which will be further explained in Section 2.3.

Unfortunately, while *pattern* is at the heart of pattern-based IE, it remains an open issue as how to find good patterns in a principled and systematic manner. How do we find patterns like p_1 and p_2? Which one is better and Why? (Presumably, p_1 is more accurate to indicate the desired capital relationship; however, p_2 can likely return more complete answers.) For finding patterns to use, the existing works fall into two categories: On the one hand, patterns may be manually specified (by inspecting the corpus), which is the approach from early pioneering work [55] to recent ones [35, 109]. On the other hand, patterns can be learned implicitly in an iterative process, as in DIPRE [18] and Snowball [4]. As Section 2.2 will contrast, in these iterative learning approaches, patterns are lacking principled quality metrics, and they are often hidden as a by-product in the iterative process.

In this chapter, as our first goal, we propose and address the problem of *pattern search*—

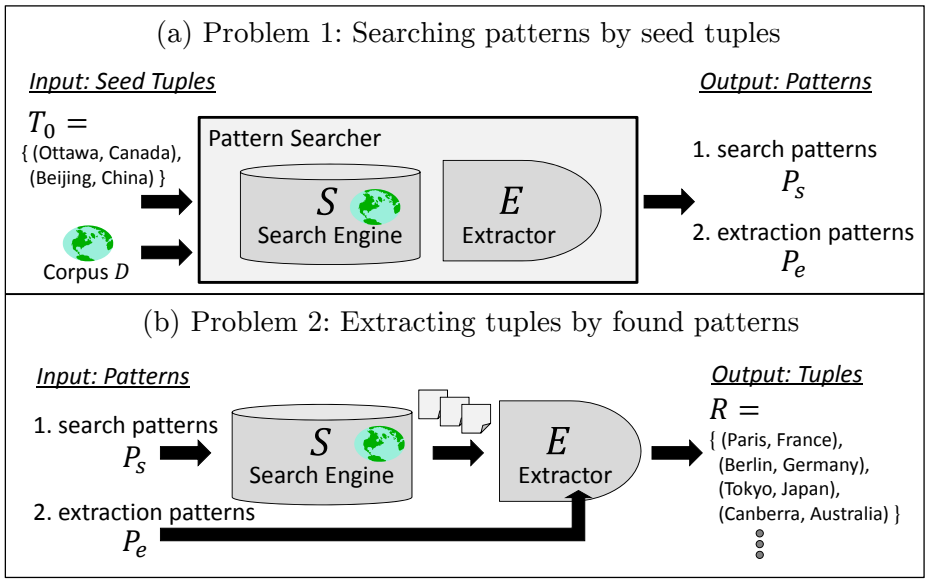


Figure 2.1: The dual problems of pattern search and tuple extraction.

to find and rank reusable patterns in a principled way. As Figure 2.1(a) illustrates, given a small number of seed examples, such as (Ottawa, Canada) and (Beijing, China), we wish to search systematically in the space of candidates to return good patterns in an order ranked by their quality. Such explicit and principled discovery of patterns not only enhances extraction effectiveness because patterns are now of high quality, but also enables new scenarios. First, with reusable patterns, we can execute and re-execute these patterns on demand to extract new tuples over an evolving corpus like the Web, without relearning the patterns as a part of the extraction process. Second, with such patterns, we can also execute focused retrieval by combining them with ad hoc keywords (*e.g.*, use "ancient history" + "⟨city⟩ is the capital ... of ⟨country⟩" to find capital cities that are also historical), as recently explored in ad hoc content querying [109].

As our second goal, upon the patterns found we study the subsequent problem of scalable *tuple extraction*, to form a complete framework for relation extraction shown in Figure 2.1(b). Our proposed solution relies on the ranked patterns of various granularities to match different scopes of the corpus, enabling the extraction of new tuples in a scalable manner. It is also important to return tuples of high quality in a principled way, parallel to the problem of

pattern search. Thus, the ultimate purpose of the patterns can be fulfilled—to efficiently and effectively find more tuples for a given relation.

As it turns out, both problems boil down to developing effective *ranking* of patterns and tuples. Our goal is thus to study the principles for such ranking. While no formal principles exist, the informal insight of *Pattern-Relation Duality* (or PR Duality) has long been observed since DIPRE [18]. It is essentially stated as follows:

PR Duality (Original): *Given a good set of patterns, we can build a good set of tuples. Given a good set of tuples, we can build a good set of patterns.*

While the insight has been widely used in existing studies, it falls short of providing a formal principle. Subsequent works [4, 3] continue to leverage the insight, but mostly resort to heuristics for modeling the iterative reinforcement. First, what does "good" mean? We need to formalize a set of metrics—precision and recall—for capturing the quality of patterns as well as tuples. Second, how exactly do patterns and tuples interrelate? We need to develop a probabilistic inference framework that captures the propagation of metrics between them. As the foundation of our approach, we address these questions and propose the conceptual model PRDualRank, and thus "rediscover" PR Duality as the underlying principle for ranking both patterns and tuples.

PR Duality (Rediscovered): *Patterns and tuples reinforce each other under the quality metrics of precision and recall. That is, the quality of a pattern, both precision and recall, can be determined by the tuples it extracts. Likewise, the quality of a tuple can be determined by the patterns it matches.*

We further develop a concrete framework to realize PRDualRank. Conceptually, PRDualRank assumes the complete knowledge of all possible patterns and tuples, which is infeasible on a large corpus. Hence, in this concrete framework, we introduce a sampling process to partially materialize PRDualRank.

In summary, this chapter makes the following contributions:

- **Problems**: We propose the dual problems of *pattern search* and *tuple extraction*, which boils down to the metrics of precision and recall.
- **Conceptual model**: We present the formal *inference mechanism* for probabilistic precision and recall, which correspond to two different forms of random walk.
- **Concrete framework**: We develop a framework to partially materialize PRDualRank under our sampling guideline.
- **Empirical results**: We conduct *extensive experiments* over the real Web. In particular, we improved optimal F-score by a factor of up to 1.64.

2.2 Related Work

In terms of problem. Extracting tuples of a given relation from a text corpus has long been studied [18, 4, 35]. However, its dual problem of searching textual patterns only exists implicitly as an intermediate step of tuple extraction.

In contrast, this chapter treats patterns as first class citizens. We first stress their *reusability*. We explicitly search and rank patterns in a principled way, such that they can be reused for future extraction (*e.g.*, over the evolving Web at a later point), or to be used by other systems like DoCQS [109]. Second, we stress their *scalability*, which hinges on a principled framework for finding out "searchable" patterns to narrow the scope of extraction. Most previous works but QXtract [5] have not discussed the automatic discovery of searchable patterns. QXtract generates queries (*i.e.*, searchable patterns) to retrieve relevant documents that may contain tuples in a preprocessing step, after which existing methods like Snowball [4] is used on the retrieved documents separately. We instead identify the principles behind the ranking of patterns, and thus integrate the ranking of such searchable patterns into the same framework PRDualRank as other kinds of patterns.

For the quality metrics, none of the previous approaches explores a principled framework to derive the quality of the tuples and patterns. In particular, none develops a complete set of metrics covering both precision and recall. Instead, most emphasize "confidence" or "probability assessment," which captures precision only in an informal and heuristic way. For instance, in Snowball [4], confidence estimation is based on the assumption of independent patterns, resulting in excessively high confidence for all tuples. In QA [83], the relation must have a key attribute. For example, ⟨country⟩ is a key attribute of `capital-city-of`, as it cannot associate with more than one ⟨city⟩ as its capital. In KnowItAll [35, 36], it is assumed that tuples of one relation do not satisfy other relations. These assumptions are not always valid. Some other studies [91, 95, 11] estimate the confidence based on some simple statistics.

For the problem setting, we rely on bootstrapping using seed tuples, similar to some previous work [18, 4].

In terms of techniques. We propose a semi-supervised approach based on iterative inference on tripartite graphs constructed from tuples, patterns and their contexts. Such graph-based inference propagation is widely reported in the literature [3, 2, 106]. Specifically, our inference involves rewriting of a node's precision and recall in terms of its neighbors on the graph, which maps to different forms of random walk as inspired by an earlier study [2]. In particular, they study the problem of query template mining, where a tripartite graph can be constructed from queries, templates, and sites. The precision and recall of queries can be inferred from its neighboring sites through clickthroughs and its neighboring templates through instantiation. We share their insight that the quality of tuples and patterns can be inferred from their neighboring nodes. However, we are addressing a distinct problem of relation extraction, where patterns and tuples serve different purposes from queries, templates and sites in the problem of query template mining. In addition, in their setting, all queries and clickthroughs are available from a query log. All possible templates can also

(a) Example snippets from corpus

Id	Doc	Snippet	Freq.
s_1	d_1, d_3, d_4, d_5	**Paris** is the capital city of **France**...	4
s_2	d_1	**Paris** is the largest city of **France**...	1
s_3	d_2, d_3	**Ottawa** is the national capital city of **Canada**...	2
s_4	d_4, d_6	**Toronto** is the largest city of **Canada**...	2

(b) Tuples

t_1 = (Paris, France) , t_2 = (Ottawa, Canada), t_3 = (Toronto, Canada)

(c) Patterns

Search Pattern P_s (not necessarily from the above snippets)
p_a = "tourist | government"
p_b = "capital city | congress"

Extraction Patterns P_e (from the above snippets)
p_1 = "⟨city⟩ is the national capital ... of ⟨country⟩"
p_2 = "⟨city⟩ is the ... capital city of ⟨country⟩"
p_3 = "⟨city⟩ is the largest city of ⟨country⟩"
p_4 = "⟨city⟩ is the ... city of ⟨country⟩"

Figure 2.2: Running example for relation extraction.

be enumerated from this log. But in our problem, we are only given a few seeds as input, without the complete spaces of tuples and patterns. Thus, we must resort to the partial materialization of the graph, based on our sampling guideline.

2.3 Problems: Pattern Search and Tuple Extraction

Towards scalable and reusable pattern-based relation extraction, we must address two dual problems, in which the concept of pattern is central. As Figure 2.1 shows, with respect to a corpus D of documents (say, the Web), we are interested in developing a mechanism that can extract a target relation R from D, by way of searchable and reusable patterns.

2.3.1 Data Model

We start with defining our data model. Consider a running example in Figure 2.2 over a toy corpus D (say, some web pages), for a target relation R = `capital-city-of`.

A *corpus* D is a collection of documents, where some "snippets" may contain the relation

of interest. Figure 2.2 shows some example snippets. Each snippet may appear multiple times, as its *frequency*. For example, s_1 occurs 4 times. Since our extraction is to recognize a span of text (such as "Paris" in s_1) as a certain semantic role (such as a capital city), we denote the *vocabulary*, *i.e.*, all the tokens (which can be words, phrases and symbols depending on the application) in D, by Ω.

As our ultimate target, a *relation* contains a set of *tuples* of the form $(e_1, \ldots, e_n)$, where e_i is an attribute or *entity*. Each entity e of a type $\langle \mathsf{E} \rangle$ is an instance of its type, which draws value from its domain $\Omega\langle \mathsf{E} \rangle$. In other words, if e is of type $\langle \mathsf{E} \rangle$, then $e \in \Omega\langle \mathsf{E} \rangle$. For example, $e =$ "France" is of type $\langle \mathsf{country} \rangle$, where $\Omega\langle \mathsf{country} \rangle = \{$"France", "USA", "China", $\ldots\}$. Note that the recognition of entities from text (*e.g.*, "France" in snippet s_1 is a $\langle \mathsf{country} \rangle$) is itself an active research area. As this chapter focuses on relation extraction, we assume that the extractor is equipped with an entity tagger; many off-the-shelf tools are available today. In Figure 2.2, we thus recognize three tuples t_1, t_2 and t_3 of the form $(\langle \mathsf{city} \rangle, \langle \mathsf{country} \rangle)$. While our discussion in this chapter assumes a binary relation R of the form $(\langle \mathsf{E}_1 \rangle, \langle \mathsf{E}_2 \rangle)$, the framework is general with respect to the arity.

At the core of pattern-based extraction lies the notion of pattern. Generally, a *pattern* is a syntactic structure with tokens and entity types that describe a "presentation" of a desired tuple—how a tuple may appear in D. Over a large corpus, to speed up extraction, we will need patterns of different granularities to match various scopes like documents, passages, and sentences.

For our discussion, we assume two classes of patterns, each with a rather simple form. However, we stress that the framework is general to handle any number of various classes of patterns, which can be defined in arbitrarily complex forms. Our first class of patterns aims at retrieving relevant documents from the corpus by a search engine: a *search pattern* is a set of searchable keyword phrases matching a relevant document d. Figure 2.2 shows some example search patterns. For example, $p_b =$ "capital city | congress" will match documents with two phrases "capital city" and "congress" in no particular order. Since a search pattern

is to be executed by a search engine, its expressiveness is limited by the chosen engine. Our second class of patterns will locate actual slots where desired entities occur to form a tuple: an *extraction pattern* further matches a text snippet s within a relevant document d. In Figure 2.2, p_1 = "⟨city⟩ is the national capital ... of ⟨country⟩" specifies that the entities and the keywords appear in that order, with some optional text "..." in between. It will match s_3—by aligning the words and recognizing "Ottawa" as a ⟨city⟩ and "Canada" a ⟨country⟩—and extract (Ottawa, Canada) as a matching tuple. Formally, we adopt the following two simple forms of patterns.

DEFINITION 2.1 (SEARCH PATTERN). A *search pattern* is a set of phrases, written as "$g_1 \mid \ldots \mid g_m$". Each *phrase* g_i is an n-gram of words $(w_1 \ldots w_n)$, where $w_i \in \Omega$, and $|g_i| = n \leq L_{\max}$, for some choice of phrase length $L_{\max}$.

DEFINITION 2.2 (EXTRACTION PATTERN). An *extraction pattern* for $(\langle \mathsf{E}_1 \rangle, \langle \mathsf{E}_2 \rangle)$ is a list of two phrases, "$g_1 \ldots g_2$" or "$g_2 \ldots g_1$", where "..." is an optional wildcard. Each g_i is a phrase containing a reference to $\langle \mathsf{E}_i \rangle$, *i.e.*, $g_i = l_1, \ldots, l_{n_1}, \langle \mathsf{E}_i \rangle, r_1, \ldots, r_{n_2}$ where $l_i, r_j \in \Omega$ and $|g_i| \leq L_{\max}$, for some choice of phrase length $L_{\max}$.

2.3.2 Problems

Our approach consists of two dual stages, as Figure 2.1 illustrates. As input, like many existing pattern-based extraction efforts [18, 4], we assume a small number of seed tuples such as (Ottawa, Canada) and (Beijing, China). As output, our ultimate goal is to find additional tuples for the matching relation, which is `capital-city-of` in this example. However, unlike existing studies, we emphasize the role of patterns, and stress the needs for scalability and reusability. For *scalability*, as we just explained, we support two (or more) classes of patterns, where search patterns can quickly retrieve relevant documents, and extraction patterns can accurately identify tuples from text. For *reusability*, we systematically search

for good patterns as a first-class citizen instead of a by-product hidden in the extraction process, and rank them with quality metrics in a principled framework. Overall, we must develop effective techniques for two problems.

Problem 1: Pattern search. Given a small number of seed tuples from relation R, search the corpus to find and rank patterns for R.

Problem 2: Tuple extraction. Given discovered patterns for R, execute them over the corpus to find and rank tuples for R.

2.4 Pattern-Relation Duality

The key challenge lies in finding good patterns (for pattern search) and good tuples (for tuple extraction). The principle of ranking, as it turns out, is a formalization of the long-existing insight of PR Duality. Our study attempts to formalize PR Duality, leading to the PRDualRank framework for the dual ranking of patterns and tuples. While our discussion mentions only extraction patterns, the same framework can be applied to search patterns (and, in principle, any definition of patterns matching certain scope for extraction).

Pattern and tuple spaces. To begin with, we examine our search space. That is, what is the set of candidate patterns or tuples we should consider?

Let us look at extraction patterns first. By Definition 2.2, a pattern is a combination of two phrases g_1 and g_2, each of which can be at most $L_{\max}$ in length. Each occurrence e_i of an entity type $\langle \mathsf{E}_i \rangle$ (say, "Paris" as a $\langle \mathsf{city} \rangle$ in snippet s_1) can form such a phrase with its surrounding tokens (say "$\langle \mathsf{city} \rangle$ is", "$\langle \mathsf{city} \rangle$ is the", and so on). Given the corpus D, we thus consider the space of extraction patterns as $P = \{p \mid p = \text{“}g_1 \ldots g_2\text{”}$, g_i is a phrase containing a reference to $\langle \mathsf{E}_i \rangle$, g_i occurs in D, $|g_i| \leq L_{\max}\}$. For our example in Figure 2.2, the space would include p_1, p_2, p_3, p_4, and many that are not listed, such as "$\langle \mathsf{city} \rangle$ is ... of $\langle \mathsf{country} \rangle$".

Similarly, for ranking tuples, there is also a potentially large space of candidates. For a relation R of the form $(\langle \mathsf{E}_1 \rangle, \langle \mathsf{E}_2 \rangle)$, such as $(\langle \mathsf{city} \rangle, \langle \mathsf{country} \rangle)$, the complete space is the Cartesian product of the two entity domains $\Omega\langle \mathsf{E}_1 \rangle \times \Omega\langle \mathsf{E}_2 \rangle$, such as any combination of city and country. As our goal is to extract such tuples from corpus D, we should only consider those pairs that actually appear in D. In fact, since entities that are far apart are unlikely to be semantically associated, we need only consider those $\langle \mathsf{E}_1 \rangle$ and $\langle \mathsf{E}_2 \rangle$ that are close within some given window. Thus, the space of tuple to consider is $T = \{t \mid t = (e_1, e_2),\ e_i \in \Omega\langle \mathsf{E}_i \rangle,\ e_1, e_2$ occur in D within x words$\}$. For our example in Figure 2.2 (assuming $x = 10$), if the entities only occur in the listed snippets, then the space is $T = \{t_1, t_2, t_3\}$.

Tuple-pattern association. To formally develop PR Duality, we further capture the associations between patterns and tuples, which enable us to systematically relate them and induce the duality between them.

Intuitively, each co-occurrence of a tuple t and pattern p in some snippet forms an *association* that is "intended" by the author of the document. We denote such an association as (t, p). Since each association (t, p) can occur multiple times in different snippets and documents, it has a *frequency* equaling to its total count of occurrences. For example, Figure 2.3(a) lists the associations in the snippets of Figure 2.2(a). t_1 co-occurs with p_2 and thus forms an association (t_1, p_2) with a frequency of 4, since snippet s_1 occurs 4 times. As another example, t_1 and p_4 co-occur at both s_1 and s_2, forming an association (t_1, p_4) with a total frequency of $4 + 1 = 5$.

While an association binds a specific pair of tuple and pattern, a pattern can associate with multiple tuples, and vice versa. Let us denote the multiset of tuples associated with p as $\tau(p)$, and similarly the multiset of patterns associated with t as $\pi(t)$. (They are multisets since an association can have multiple occurrences.) For instance, $\tau(p_2) = \{t_1 : 4, t_2 : 2\}$ and $\pi(t_1) = \{p_2 : 4, p_3 : 1, p_4 : 5\}$, according to Figure 2.3(a). Furthermore, let W_{ij} be the frequency of the association (t_i, p_j).

(a) Example associations

Association	Where	Freq.
(t_1, p_2)	s_1	4
(t_1, p_3)	s_2	1
(t_1, p_4)	s_1, s_2	5
(t_2, p_1)	s_3	2
(t_2, p_2)	s_3	2
(t_2, p_4)	s_3	2
(t_3, p_3)	s_4	2
(t_3, p_4)	s_4	2

(b) Example graph

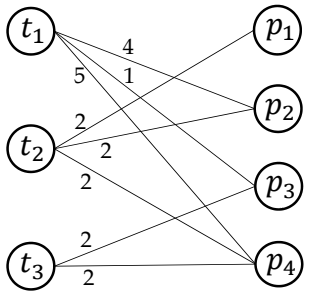


Figure 2.3: Tuple-pattern associations and graph.

Given the tuple space T and pattern space P, we can construct a *tuple-pattern graph* G as a convenient way to visualize associations. $G = (T, P)$ is a bipartite graph where T and P form two disjoint sets of nodes, as illustrated in Figure 2.3(b). The set of edges models the associations, such that an association (t_i, p_j) implies that t_i and p_j are connected by an edge weighted by the frequency W_{ij}. For example, (t_1, p_2) connects t_1 and p_2 with an edge of weight 4.

2.4.1 Metrics: Precision and Recall

Given the space of possible patterns P, we need formal metrics to measure their quality. As our search and ranking spaces encompass interrelated T and P, we aim to develop metrics that will uniformly apply to both patterns and tuples, in order to unify and integrate their mutual reinforcement, which is the intuition of PR Duality.

To determine the right metrics, we investigate the purpose of patterns, which is to extract, or "retrieve", tuples relevant to the target relation R. With an *oracle* who examines every snippet, we would be able to judge the relevance of each tuple with respect to R, and thus determine R. In order to capture the frequency of the member tuples, R is modeled as a multiset. The frequency of a tuple t is defined as the total number of times t can be extracted in the corpus (by any possible pattern), *i.e.*, $|\pi(t)|$. In our example, $R = \{t_1 : 10, t_2 : 6\}$.

Intuitively, a good pattern should match more relevant tuples, and fewer irrelevant ones.

In other words, the purpose of a pattern is to retrieve those relevant tuples in R. We can thus measure the quality of a pattern p by its "retrieval effectiveness" for finding R from the entire space T. The objectives of set retrieval naturally translate to the need of both *precision* $\mathcal{P}$ and *recall* $\mathcal{R}$, as in standard information retrieval evaluation. Consider pattern p_1 in our example, which retrieves $\tau(p_1) = \{t_2 : 2\}$. With respect to R, we thus have precision $\mathcal{P}(p_1) = \frac{|R \cap \tau(p_1)|}{|\tau(p_1)|} = \frac{2}{2} = 1.0$, and recall $\mathcal{R}(p_1) = \frac{|R \cap \tau(p_1)|}{|R|} = \frac{2}{10+6} = 0.125$. As another example, pattern p_4 that retrieves $\tau(p_4) = \{t_1 : 5, t_2 : 2, t_3 : 2\}$ will have $\mathcal{P}(p_4) = \frac{|R \cap \tau(p_4)|}{|\tau(p_4)|} = \frac{5+2}{5+2+2} = 0.78$ and $\mathcal{R}(p_4) = \frac{|R \cap \tau(p_4)|}{|R|} = \frac{5+2}{10+6} = 0.44$. Apparently, while some patterns like p_1 have a higher precision but a lower recall, other patterns like p_4 have a higher recall but a lower precision. Formally, letting $|\cdot|$ denote the sum of frequencies of all members in a set, we have

$$\mathcal{P}(p) \triangleq |R \cap \tau(p)|/|\tau(p)|, \tag{2.1}$$

$$\mathcal{R}(p) \triangleq |R \cap \tau(p)|/|R|. \tag{2.2}$$

The two metrics can also be applied to tuples, as we can view each tuple t as retrieving a multiset $\phi(t)$ that only contains t itself (with its frequency). Subsequently,

$$\mathcal{P}(t) \triangleq |R \cap \phi(t)|/|\phi(t)|, \tag{2.3}$$

$$\mathcal{R}(t) \triangleq |R \cap \phi(t)|/|R|. \tag{2.4}$$

The uniform definitions of precision and recall on both patterns and tuples enable us to unify their mutual reinforcement, as we will see next.

2.4.2 Inference: Mutual Reinforcement

In our problem setting, we are only give a few seed tuples for the target relation. In other words, R is unknown (or rather, not completely known), and thus the deterministic definitions of precision and recall in Eq. 2.1–2.4 cannot be directly implemented. It is only feasible to redefine and infer precision and recall in a probabilistic manner.

2.4.2.1 Probabilistic Definitions

As the foundation, we need to generalize the deterministic definitions of precision and recall to probabilistic ones. Let us start with patterns and consider Eq. 2.1. As its numerator is the count of $R \cap \tau(p)$ and its denominator is the count of $\tau(p)$, it represents the conditional probability that a tuple x belongs to R given that x is drawn from $\tau(p)$, which we call the *probabilistic precision* of p. We can similarly generalize Eq. 2.2 to define the *probabilistic recall* of p. Thus, we have

$$\mathcal{P}(p) \triangleq p\left(x \in R | x \in \tau(p)\right), \tag{2.5}$$

$$\mathcal{R}(p) \triangleq p\left(x \in \tau(p) | x \in R\right). \tag{2.6}$$

For a tuple $t \in T$, we can generalize in a similar fashion. In particular, for precision, we can rewrite Eq. 2.3 as $p\left(x \in R | x \in \phi(t)\right)$. Since $\phi(t)$ only contains t, the expression can be simplified to just $p\left(t \in R\right)$, which intuitively measures the probability of t being relevant. Likewise, for recall, we can rewrite Eq. 2.4 as $p\left(x \in \phi(t) | x \in R\right)$, which simplifies to $p\left(x = t | x \in R\right)$ as $\phi(t)$ only contains t. Thus, we define that

$$\mathcal{P}(t) \triangleq p\left(t \in R\right), \tag{2.7}$$

$$\mathcal{R}(t) \triangleq p\left(x = t | x \in R\right). \tag{2.8}$$

2.4.2.2 Probabilistic Inference

The inference of probabilistic precision and recall enables the dual ranking of tuples and patterns. We summarize the main results in Figure 2.4. The results describe the inference rules P1 and P2 for probabilistic precision, as well as R1 and R2 for probabilistic recall, which are mutually reinforcing between tuples and patterns.

We derive these rules in the following. Note that our derivations are inspired by the QueST framework [2] for the problem of query template mining, but we are addressing a distinct problem of relation extraction, as Section 2.2 explained.

(a) Inference of probabilistic precision	(b) Inference of probabilistic recall
P1 $\mathcal{P}(p_j) = \sum_{t_i \in \tau(p_j)} \mathcal{P}(t_i) \cdot \frac{W_{ij}}{\lvert\tau(p_j)\rvert}$	**R1** $\mathcal{R}(p_j) = \sum_{t_i \in \tau(p_j)} \frac{W_{ij}}{\lvert\pi(t_i)\rvert} \mathcal{R}(t_i)$
P2 $\mathcal{P}(t_i) = \begin{cases} 1 & \text{if } t_i \in T_0; \\ \sum_{p_j \in \pi(t_i)} \mathcal{P}(p_j) \cdot \frac{W_{ij}}{\lvert\pi(t_i)\rvert} & \text{else.} \end{cases}$	**R2** $\mathcal{R}(t_i) = \sum_{p_j \in \pi(t_i)} \frac{W_{ij}}{\lvert\tau(p_j)\rvert} \mathcal{R}(p_j)$

Figure 2.4: Inference rules for probabilistic precision and recall.

Inference of precision. For a pattern p_j, We rewrite $\mathcal{P}(p_j)$ into a function of $\mathcal{P}(t_i)$, for those $t_i \in \tau(p_j)$. The derivation leads to rule P1 in Figure 2.4.

$$
\begin{aligned}
\mathcal{P}(p_j) &\triangleq p\left(x \in R | x \in \tau(p_j)\right) \\
&\overset{1}{=} \sum_{t_i \in \tau(p_j)} p\left(x \in R, x = t_i | x \in \tau(p_j)\right) \\
&\overset{2}{=} \sum_{t_i \in \tau(p_j)} p\left(x \in R | x = t_i, x \in \tau(p_j)\right) p\left(x = t_i | x \in \tau(p_j)\right) \\
&\overset{3}{=} \sum_{t_i \in \tau(p_j)} p\left(x \in R | x = t_i\right) p\left(x = t_i | x \in \tau(p_j)\right) \\
&\overset{4}{=} \sum_{t_i \in \tau(p_j)} p\left(t_i \in R\right) p\left(x = t_i | x \in \tau(p_j)\right) \\
&\overset{5}{=} \sum_{t_i \in \tau(p_j)} \mathcal{P}(t_i) \frac{W_{ij}}{\lvert\tau(p_j)\rvert}
\end{aligned} \tag{2.9}
$$

Step 1 expands with joint distributions with every $t_i \in \tau(p_j)$. Step 2 uses Bayes' rule. Step 3 removes the condition $x \in \tau(p_j)$, of which $x \in R$ is conditionally independent, given the stricter condition $x = t_i$. Step 4 simply rewrites $p\left(x \in R | x = t_i\right)$ as $p\left(t_i \in R\right)$. Then, in step 5, we recognize the first term as simply $\mathcal{P}(t_i)$, while the second term can be calculated as $\frac{W_{ij}}{\lvert\tau(p_j)\rvert}$, which is the proportion of t_i among the tuples associated with p_j.

To enable the mutual reinforcement between patterns and tuples, we also rewrite $\mathcal{P}(t_i)$ in terms of $\mathcal{P}(p_j)$, for those $p_j \in \pi(t_i)$, resulting in rule P2 in Figure 2.4. Together, rules P1

and P2 form the inference mechanism for probabilistic precision.

$$\begin{aligned}
\mathcal{P}(t_i) &\triangleq p\left(t_i \in R\right) \\
&\overset{1}{=} \sum_{p_j \in \pi(t_i)} p\left(t_i \in R, t_i \in \tau(p_j)\right) \\
&\overset{2}{=} \sum_{p_j \in \pi(t_i)} p\left(t_i \in R | t_i \in \tau(p_j)\right) p\left(t_i \in \tau(p_j)\right) \\
&\overset{3}{=} \sum_{p_j \in \pi(t_i)} \mathcal{P}(p_j) \frac{W_{ij}}{|\pi(t_i)|}
\end{aligned} \tag{2.10}$$

Consider an arbitrary extraction of t_i by some pattern. In step 1, we expand to different $p_j \in \pi(t_i)$ such that the extraction of t_i is done by p_j, *i.e.*, $t_i \in \tau(p_j)$. Note that an extraction of t_i is obtained through only one p_j, meaning that $t_i \in \tau(p_j)$ are disjoint events for different p_j's. Step 2 applies the Bayes' rule. In step 3, we recognize that the first term is $\mathcal{P}(p_j)$; the second term is the probability that t_i is extracted by p_j (given that t_i has been extracted by some pattern), which can be computed as $\frac{W_{ij}}{|\pi(t_i)|}$.

Note that the rewriting does not apply to seed tuples $t_i \in T_0$, which are positive examples and should maintain their values at 1. More general forms of seed tuples can also be adopted, such as negative examples which should maintain their values at 0.

Inference of recall. The probabilistic recall of a pattern p_j (Eq. 2.6) can also be inferred from those of tuples t_i (Eq. 2.8), and vice versa, leading to rules R1 and R2 in Figure 2.4.

$$\begin{aligned}
\mathcal{R}(p_j) &\triangleq p\left(x \in \tau(p_j) | x \in R\right) \\
&\overset{1}{=} \sum_{t_i \in \tau(p_j)} p\left(x \in \tau(p_j), x = t_i | x \in R\right) \\
&\overset{2}{=} \sum_{t_i \in \tau(p_j)} p\left(x \in \tau(p_j) | x = t_i, x \in R\right) p\left(x = t_i | x \in R\right) \\
&\overset{3}{=} \sum_{t_i \in \tau(p_j)} p\left(x \in \tau(p_j) | x = t_i\right) p\left(x = t_i | x \in R\right) \\
&\overset{4}{=} \sum_{t_i \in \tau(p_j)} \frac{W_{ij}}{\pi(t_i)} \mathcal{R}(t_i)
\end{aligned} \tag{2.11}$$

Step 1 expands to joint probabilities with $t_i \in \tau(p_j)$. In step 2, we apply Bayes' rule. In step 3, as $x = t_i$ is a stricter condition, $x \in R$ is removed. In step 4, the first term can be computed as $\frac{W_{ij}}{\pi(t_i)}$, and the second term is simply $\mathcal{R}(t_i)$.

$$\begin{aligned}
\mathcal{R}(t_i) &\triangleq p\left(x = t_i | x \in R\right) \\
&\overset{1}{=} \sum_{p_j \in \pi(t_i)} p\left(x = t_i, x \in \tau(p_j) | x \in R\right) \\
&\overset{2}{=} \sum_{p_j \in \pi(t_i)} p\left(x = t_i | x \in \tau(p_j), x \in R\right) p\left(x \in \tau(p_j) | x \in R\right) \\
&\overset{3}{=} \sum_{p_j \in \pi(t_i)} p\left(x = t_i | x \in \tau(p_j)\right) p\left(x \in \tau(p_j) | x \in R\right) \\
&\overset{4}{=} \sum_{p_j \in \pi(t_i)} \frac{W_{ij}}{|\tau(p_j)|} \mathcal{R}(p_j)
\end{aligned} \tag{2.12}$$

Consider an arbitrary extraction of t_i by some pattern. Step 1 expands to $p_j \in \pi(t_i)$ such that t_i is extracted by p_j. Step 2 applies Bayes' rule. In step 3, given some extraction by the pattern p_j, the extraction is only determined by p_j and t_i's relevance does not matter. In step 4, the first term can be computed as $\frac{W_{ij}}{|\tau(p_j)|}$, and the second term is $\mathcal{R}(p_j)$.

Overall algorithm: PRDualRank. We now introduce the overall algorithm called PRDualRank in Figure 2.5, to rank both tuples and patterns by precision as well as recall. Given the seed tuples and the tupe-pattern graph, we first initialize precision and recall for tuples (line 1–3). Next, we update the precision and recall of patterns and tuples iteratively until some stopping condition (line 4–7). Finally, ranked patterns and tuples are outputted (line 8). Note that the stopping condition can be specified in different ways, *e.g.*, by a fixed number of iterations, or until convergence (which may require certain adjustment of the updating rules to ensure that non-trivial convergence occurs).

As a concluding remark, PRDualRank not only exemplifies the original PR Duality in [18], but also formally quantifies and thus "rediscovers" it, as first stated in Section 2.1. In particular, both precision and recall of a pattern can be expressed in terms of its associated

Input: Tuple-pattern graph $G = (T, P)$; seed tuples $T_0 \subset T$
Output: P and T ranked by probabilistic precision/recall

foreach $t \in T_0$ **do**
  $\mathcal{P}(t) \leftarrow 1$; $\mathcal{R}(t) \leftarrow \frac{1}{|T_0|}$;
end
repeat
  update $\mathcal{P}$ by rules P1 and P2;
  update $\mathcal{R}$ by rules R1 and R2;
until termination condition;
return P and T ranked by $\mathcal{P}$ or $\mathcal{R}$.

Figure 2.5: PRDualRank algorithm: Ranking by PR Duality.

tuples, by rule P1 and R1 in Figure 2.4. In duality, the quality of a tuple can be expressed in terms of its associated patterns, by rule P2 and R2. In other words, our inference process boils down to the mutual reinforcement between patterns and tuples.

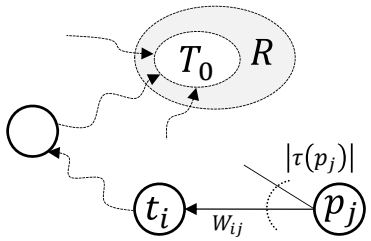
2.4.3 Interpretation: Forward and Backward Random Walks

The resulting inference framework for precision and recall, exhibits a rather interesting form of duality—they are symmetric and "opposite" to each other. Similar to what has been observed in the QueST framework [2], our inference for precision and recall can be interpreted as two different forms of random walks on the tuple-pattern graph.

On the one hand, probabilistic precision is a random walk *backward* as depicted in Figure 2.6(a). That is, starting from a pattern p_j on the tuple-pattern graph, what is the probability that we will reach a seed tuple in T_0 (a subset of the target relation R)? Informally, it measures the proportion of tuples associated with p_j that will reach the seed tuples, which captures the intuition of precision.

On the other hand, recall is a random walk *forward* as depicted in Figure 2.6(b): Starting from a seed tuple in $T_0 \subset R$ on the tuple-pattern graph, what is the probability that we will reach p_j? Intuitively, it is consistent with probabilistic recall of p_j, *i.e.*, the proportion of tuples in R that will reach p_j.

(a) Precision by backward random walk



(b) Recall by forward random walk

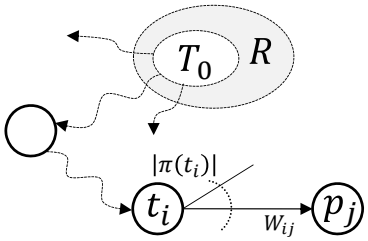


Figure 2.6: Interpreting probabilistic precision and recall as random walks.

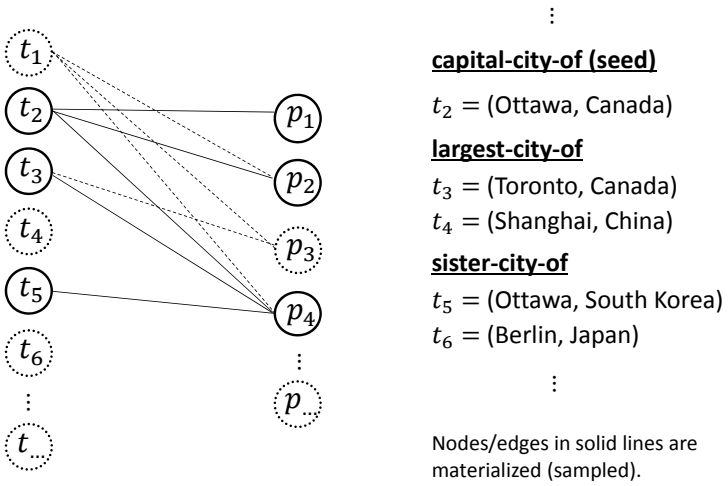


Figure 2.7: Partially materialized tuple-pattern graph.

2.5 Concrete Framework

The major challenge in realizing our inference framework PRDualRank lies in the materialization of the tuple-pattern graph. Unlike the QueST framework [2], we do not know the entire spaces of tuples and patterns as well as their associations, due to the immense scale of the corpus (such as the Web).

Thus, we resort to partial materialization, starting only with a few seed tuples. To illustrate, a partial graph is constructed in Figure 2.7, which is extended from the example graph in Figure 2.3 (tuples $t_1, \ldots, t_3$ and patterns $p_1, \ldots, p_4$ are from the earlier example). In this graph, only solid nodes and edges are materialized, and dashed ones are in the "unseen" part of the corpus.

Thus, the problem lies in devising a good sampling process to partially materialize the

graph. We first describe the guideline for the sampling process. Subsequently we present a concrete framework to the dual problems of pattern search and tuple extraction.

2.5.1 Sampling Guideline

In the sampling process, it is important to capture tuples of different relations. As illustrated in Figure 2.7, $t_2, \ldots, t_6$ satisfy three different relations on $\Omega\langle\mathsf{city}\rangle$ and $\Omega\langle\mathsf{country}\rangle$. Intuitively, if we sample tuples from different relations, it helps to better distinguish the "relevant" patterns from the irrelevant ones, similar to the building of a classifier which requires both positive and negative (or at least unlabeled) examples. The sampled tuples become the unlabeled examples. In our example, ideally at least one tuple from $\{t_3, t_4\}$ (for relation `largest-city-of`, as well as one from $\{t_5, t_6\}$ (for relation `sister-city-of`) should be sampled, given the seed t_2.

Equivalence class. We say that two tuples t, t' are *equivalent* if and only if they satisfy exactly the same set of relations, denoted $t \sim t'$. Note that a tuple may have multiple senses. For instance, (Paris, France) satisfies both `capital-city-of` and `largest-city-of` relations. In our example, t_3 is equivalent to t_4, and so is t_5 to t_6. Subsequently, the tuple space T can be partitioned into *equivalence classes* consisting of equivalent tuples. We denote the equivalence classes as sets $T_1, \ldots, T_m$, *i.e.*, $T = \cup_{k=1}^{m} T_k$.

Equivalence assumptions. We first assume that tuples in the same equivalence class can be extracted by the same set of patterns, and for the same number of times. In other words, the corpus is comprehensive and balanced, containing snippets that express each tuple in every possible way. The Web is a fair approximation of such a corpus. Using our notation, given two equivalent tuples $t_i \sim t_j$, we have

$$\forall p_k \in P, \quad t_i \in \tau(p_k) \Leftrightarrow t_j \in \tau(p_k) \;\text{ and }\; W_{ik} = W_{jk}. \tag{2.13}$$

We further assume that tuples in the same equivalent class have the same precision and

recall. That is, for $t \sim t'$,

$$\mathcal{P}(t) \equiv \mathcal{P}(t') \quad \text{and} \quad \mathcal{R}(t) \equiv \mathcal{R}(t'). \tag{2.14}$$

Equivalence sampling. In the following, we establish an "equivalence sampling" process, which, in principle, ranks patterns by probability precision and recall in the same order as full materialization does.

PROPOSITION 2.1 (EQUIVALENCE SAMPLING). Suppose the same proportion $\rho \in (0, 1)$ of tuples are sampled from each equivalence class, and all of the patterns and associations of a sampled tuple are also sampled. Let $\hat{T} = \cup_{k=1}^{m} \hat{T}_k \subseteq T$ denote the sampled subspace of tuples, where $\hat{T}_k \subseteq T_k$ and $|\hat{T}_k| = \rho |T_k|$. Furthermore, let $\hat{\tau}(p) \triangleq \{t | t \in \tau(p), t \in \hat{T}\}$. Assuming Eq. 2.13 and 2.14, we have the following results.

$$\mathcal{P}(p_j) = \sum_{t_i \in \tau(p_j)} \mathcal{P}(t_i) \cdot \frac{W_{ij}}{|\tau(p_j)|} \text{ (Eq. 2.9)} \equiv \sum_{t_i \in \tau(p_j) \cap \hat{T}} \mathcal{P}(t_i) \cdot \frac{W_{ij}}{|\hat{\tau}(p_j)|}, \tag{2.15}$$

$$\mathcal{R}(p_j) = \sum_{t_i \in \tau(p_j)} \mathcal{R}(t_i) \cdot \frac{W_{ij}}{|\pi(t_i)|} \text{ (Eq. 2.11)} \propto \sum_{t_i \in \tau(p_j) \cap \hat{T}} \mathcal{R}(t_i) \cdot \frac{W_{ij}}{|\pi(t_i)|}. \tag{2.16}$$

The above result implies that it is important to sample tuples of different relations at a similar rate, whereas the specific attributes of the sampled tuples does not matter. In other words, for two tuples $t \sim t'$, it makes no difference to sample either of them.

2.5.2 Pattern Search

As discussed, we need to partially construct the tuple-pattern graph, starting with only a few seed tuples. In addition to the seed tuples, our approach also requires a search engine S that interfaces with the corpus D. For the Web, general purpose keyword-based search engines are sufficient.

PR Duality implies that a tuple can be used to discover more patterns and vice versa. We illustrate key steps of this approach using snapshots in Figure 2.8. Starting with the seed

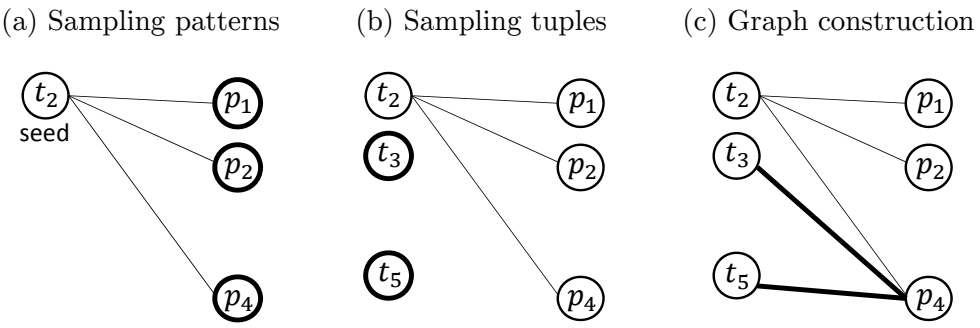


Figure 2.8: Snapshots of partial materialization.

t_2 = (Ottawa, Canada), we first query S with both attributes of t_2 (*i.e.*, Ottawa+Canada) to retrieve N_{doc} documents containing t_2 in D. From the retrieved documents, patterns associated with t_2 are found, namely p_1, p_2, p_4, as bolded in Figure 2.8(a).

Next, we query S with each attribute of the seeds. That is, for seed $t = (e_1, e_2)$, query with e_1 and e_2 separately. Then, we apply the newly found extraction patterns to sample unlabeled tuples, namely t_3 and t_5, as bolded in Figure 2.8(b). Querying with attributes of the seeds allows efficient discovery of unlabeled tuples. Note that search patterns are not used for this purpose, as they are not yet ranked at this stage, and thus it is inefficient to use all of them than to use only the attributes of the seeds. Although t_3 and t_5 both share an attribute with the seed t_2, we stress that the sampling is fair in the sense that different relations are still captured. In particular, t_3 satisfies the relation `largest-city-of`, so does t_4. Thus it makes no difference in sampling either of them. Similarly, t_5, t_6 satisfy the same relation `sister-city-of`. This is consistent with Proposition 2.1, which suggests that it only matters to sample different relations, whereas the specific attributes of the sampled tuples are unimportant.

Finally, after sampling the patterns and tuples, a partial tuple-pattern graph can be constructed by connecting tuples and patterns via their associations, as shown in Figure 2.8(c). The constructed partial graphs, one for the search patterns and one for the extraction patterns, subsequently enable PRDualRank to infer the probabilistic precision and recall for both classes of patterns.

Input: seed tuples $T_0 \subset T$; search engine S for corpus D; extractor E
Output: ranked list of search patterns P_s, and of extraction patterns P_e

1 $P_s \leftarrow \varnothing$; $P_e \leftarrow \varnothing$; $U \leftarrow \varnothing$;
2 **foreach** $t = (e_1, e_2) \in T_0$ **do**
3 $D' \leftarrow S$.Query$(e_1 + e_2, N_{\text{doc}})$;
4 $P_s \leftarrow P_s \cup \{$ search patterns found in $D' \}$;
5 $P_e \leftarrow P_e \cup \{$ extraction patterns found in $D' \}$;
6 **end**
7 **foreach** $t = (e_1, e_2) \in T_0$ **do**
8 **foreach** e_i **do**
9 $D' \leftarrow S$.Query(e_i, N_{doc});
10 $U \leftarrow U \cup \{$ tuples with attribute $e_i \in E$.Extract$(D', P_e) \}$;
11 **end**
12 **end**
13 $T \leftarrow \{$ top frequent N_{tuple} tuples $\in U \} \cup T_0$;
14 $D' \leftarrow \varnothing$;
15 **foreach** $t = (e_1, e_2) \in T$ **do**
16 $D' \leftarrow D' \cup S$.Query$(e_1 + e_2, N_{\text{doc}})$;
17 **end**
18 build graphs $G_s = (T, P_s)$ and $G_e = (T, P_e)$ based on D';
19 $P_s \leftarrow \text{PRDualRank}(G_s, T_0)$; $P_e \leftarrow \text{PRDualRank}(G_e, T_0)$;
20 **return** ranked P_s and P_e.

Figure 2.9: PatternSearch algorithm: Finding good patterns.

The above algorithm is outlined in Figure 2.9, where line 2–6 correspond to snapshot (a) in Figure 2.8, line 7–13 correspond to (b), and line 14–18 correspond to (c).

The outputs are a list of search patterns and a second list of extraction patterns. For now, each list is ranked by F-scores, the harmonic mean of precision and recall to obtain "balanced" patterns. Note that we normalize precision and recall to $[0, 1]$ to give them equal weights in computing the F-score. We will study the effect of using precision or recall-based patterns by experiments in Section 2.6.3.

2.5.3 Tuple Extraction

The ultimate goal of the patterns is to find more tuples of R from D. For instance, $t_1 =$ (Paris, France) in Figure 2.7 is not materialized, and we aim to find it by utilizing the learnt patterns (p_1, p_2, p_4).

Input: ranked search and extraction patterns P_s, P_e;
search engine S for corpus D; extractor E
Output: ranked list of tuples

1 $P_s \leftarrow \{\text{top } K_{\text{width}} \text{ patterns from } P_s\}$;
2 $D_s \leftarrow \varnothing$;
3 **foreach** $p \in P_s$ **do**
4 $\quad D_s \leftarrow D_s \cup S.\text{Query}(p, K_{\text{depth}})$;
5 **end**
6 $P_e \leftarrow \{\text{top } K_{\text{ext}} \text{ patterns from } P_e\}$;
7 $T_{\text{cand}} \leftarrow E.\text{Extract}(D_s, P_e)$;
8 **foreach** $t = (e_1, e_2) \in T_{\text{cand}}$ **do**
9 $\quad D' \leftarrow S.\text{Query}(e_1{+}e_2, K_{\text{canddoc}})$;
10 $\quad$ compute $\mathcal{P}(t)$ by rule P2 using patterns in D';
11 $\quad$ compute $\mathcal{R}(t)$ by rule R2 using patterns in D';
12 **end**
13 **return** ranked T_{cand};

Figure 2.10: TupleExtration algorithm: Extracting relevant tuples.

The learnt patterns consist of two classes, the search patterns and the extraction patterns, for different purposes. Each of the top K_{width} search patterns (search width) is used to retrieve K_{depth} documents (search depth) that may contain the target tuples. Based on our experiments, any single word is unlikely to be focused enough on the Web, thus we additionally require the selected search patterns to contain at least two words. Next, top K_{ext} extraction patterns are applied on the retrieved documents to extract candidate tuples. We also ignore extraction patterns of precision lower than 0.5, which tend to be too noisy.

Finally, we rank the candidate tuples using patterns from top K_{canddoc} documents containing each candidate tuple. We consider the F-score in ranking candidates, which can be computed from search patterns, extraction patterns, or both in conjunction. We will discuss the different ranking schemes in the experiments.

The above procedure is sketched in Figure 2.10. It has three major components: document retrieval by search patterns (line 1–5), tuple extraction by extraction patterns (line 6–7), and candidate ranking (line 8–12).

In fact, after we obtain a ranked list of candidate tuples, we can treat the highly ranked

Relation	Description	Type	Example
`birth`	Birth year of physics Nobel prize laureates	one-many	(1867, Marie Curie)
`capital`	Capital city of countries	one-one	(Paris, France)
`area-code`	Area code(s) of largest 100 U.S. cities	many-many	(617, Boston)

Figure 2.11: Target relations.

candidates as the seed tuples, and use them to relearn the patterns as Section 2.5.2 proposed. The new patterns learnt can be used to extract tuples again, resulting in an iterative process alternating between pattern search and tuple extraction. However, in practice, we observe that after one round of pattern search and tuple extraction, we can often obtain stable results as long as the parameters such as K_{width}, K_{depth} and K_{ext} are sufficiently large. We will investigate the effects of these parameters in our experiments.

2.6 Experiments

We showcase our experimental evaluation of PRDualRank on the Web across different relations. Overall, the experiments demonstrate that PRDualRank significantly outperforms the baseline in both effectiveness and efficiency.

2.6.1 Experimental Settings

Corpus. We use the World Wide Web as the corpus D, which is accessed through Yahoo API [105] as the search engine S.

Target Relations. Figure 2.11 shows the target relations we used, which are chosen for their different relationship types (*i.e.*, one-one, one-many and many-many). We use three seeds as user input for each relation (*i.e.*, $|T_0| = 3$). We report the overall performance for all the relations, whereas we report the results of finer grained experiments only for the `capital` relation, since we observe similar trends on the others.

Entity Tagging. We used a dictionary-based approach to recognize entities involved in

the target relations. Using the relation `capital` as an example, we manually prepared a dictionary of all countries in the world, as well as a dictionary of cities using sources like Wikipedia. The same tagging approach is used in PRDualRank and the baseline.

Baseline (QXtract&Snowball, or Q&S). This is the state-of-the-art scalable tuple extraction system for our problem scenario, which lack a principled framework of quality metrics. It is a two-phased system. In the first phase, QXtract [5] finds search queries based on seeds and an extraction system like Snowball [4]. The search queries are then used to retrieve documents as a preprocessing step to allow scalable tuple extraction. In the second phase, Snowball extracts tuples from the documents retrieved. For fairness, we used the same number of documents in learning the queries by QXtract and in ranking the patterns by PRDualRank as well as the same number of queries in QXtract as search patterns in PRDualRank to retrieve the same number of documents from the Web. For other parameters in Q&S, we follow the settings outlined in [5, 4].

Schemes. We include the performance of three different schemes of PRDualRank. These schemes only differ in ranking candidate tuples. They are: (a) Dual-Sch, ranking candidates by using search patterns only; (b) Dual-Ext, ranking by extraction patterns only; (c) Dual-Avg, which simply takes the average score from both classes of patterns.

Evaluation methodology. Since directly judging the quality of patterns is subjective, we only evaluate the extracted tuples, which indirectly reflect the quality of the patterns. Each method produces a ranked list of n tuples. We compare top N tuples from the output with the ground truth tuples of the target relation, which are manually prepared. For each $N = 1, \ldots, n$, we evaluate the precision and recall of the top N tuples with respect to the ground truth, and present the results in a precision-recall plot.

Additionally, there are web pages with a list of ground truth tuples on the Web. Taking `capital` as an example, there is a list of capitals for every country on `about.com` [85]. In general, we cannot assume such a list exists for any arbitrary relation, thus we should not

take advantage of them. For fairness, we treat a web page as a list if it contains more than 10% of all the ground-truth tuples, which is simply disregarded during the tuple extraction phase of every method.

Parameter settings. We discuss the parameters used in the partial materialization.

For pattern search, it is better to have larger N_{doc} (number of documents retrieved for a tuple) and N_{tuple} (number of unlabeled tuples), as the resulting graph would cover more of the corpus. In practice, as long as they are sufficiently large, the results will be fairly stable. Thus, we set $N_{\text{doc}} = 500$, $N_{\text{tuple}} = 10 \times |T_0|$.

Next, for tuple extraction, we vary K_{width} (search width), K_{depth} (search depth) and K_{ext} (number of extraction patterns) to study their effects. Otherwise, they are set to default values $K_{\text{width}} = K_{\text{ext}} = 120$, and $K_{\text{depth}} = 700$. We also set $K_{\text{canddoc}} = 50$ (number of documents retrieved for each candidate tuple), which is large enough to achieve a stable ranking for the tuples.

2.6.2 Overall Performance

Effectiveness. Following the evaluation methodology outlined in Section 2.6.1, the results of PRDualRank and the baseline on the three target relations are presented as precision-recall plots in Figure 2.12.

Observe that all schemes of PRDualRank substantially outperform Q&S (brown color or "4") on all relations. In particular, Dual-Sch (blue color or "3") which only uses search patterns to rank the tuple, is clearly better than Q&S. This implies that our search patterns are good as it not only retrieves the relevant documents, but also ranks tuples, outperforming the combined effort of Q&S. On the other hand, Dual-Ext (green color or "2") is even better, simply because extraction patterns are more tightly coupled with tuples than search patterns. This indicates that the extraction patterns have also performed well in extracting the tuples. Lastly, Dual-Avg (red color or "1") makes use of scores from both classes of patterns. Not

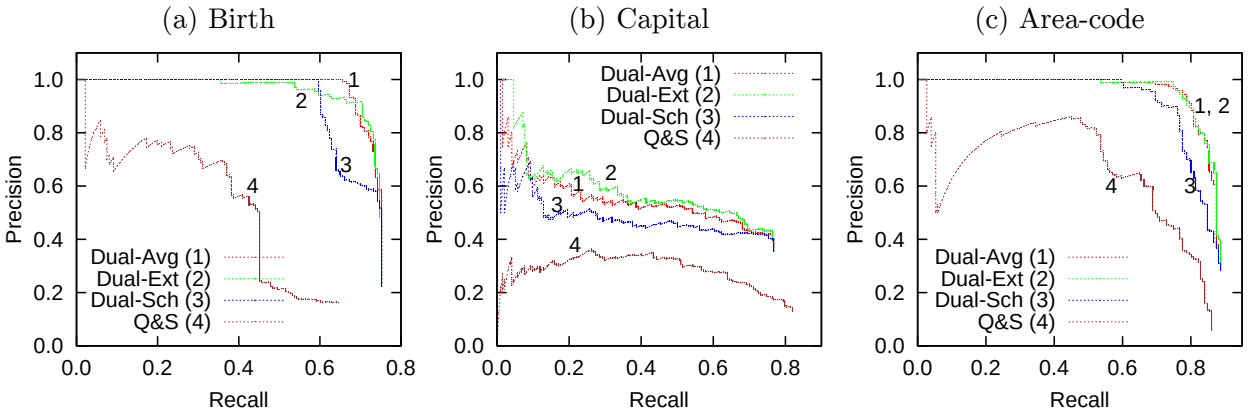


Figure 2.12: Comparison of precision and recall.

	Dual-Avg	Dual-Sch	Dual-Ext	Q&S
birth	**0.799**	0.748	0.796	0.485
capital	0.552	0.537	**0.571**	0.398
area-code	**0.859**	0.824	0.853	0.651

Figure 2.13: Comparison of optimal F-scores.

surprisingly, this combined approach achieves comparable or better results as Dual-Ext. This implies that we can potentially exploit different classes of patterns together to further improve ranking.

We also report the optimal F-scores achieved by each method in Figure 2.13. In particular, we improve the optimal F-score by a factor up to 1.6.

It is worth noting that the performance of all methods on `capital` is worse than the other two relations. The reason is that there can be many different relations on $\Omega\langle\mathsf{city}\rangle$ and $\Omega\langle\mathsf{country}\rangle$. Even the seeds themselves are inherently ambiguous, which satisfy the `city-of` relation as well.

Efficiency. To study the scalability of our approach, we present the execution time required for PRDualRank and Q&S. For different ranking schemes of PRDualRank, Dual-Avg is the slowest, whereas the other two schemes are slightly faster. Thus, for comparing with the baseline, we only show the time cost of Dual-Avg. Both methods consist of two phases. For Dual-Avg, Phase I is pattern search, and Phase II is tuple extraction. For Q&S, Phase I

	Dual-Avg			Q&S		
	Phase I	Phase II	Total	Phase-I	Phase-II	Total
birth	247	2474	2721	6124	25208	31332
capital	2580	2639	5219	13664	10020	23684
area-code	384	3243	3627	11093	8708	19801

Figure 2.14: Comparison of execution time in seconds.

is the preprocessing step of retrieving relevant documents, and Phase II is tuple extraction. The time costs for each method are presented in Figure 2.14. They do not include the time needed to download the remote web pages from the Web, but include the I/O time in accessing the downloaded local copies.

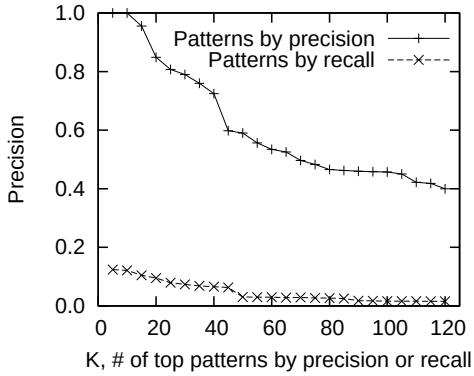
The results show that PRDualRank is generally one order of magnitude faster than Q&S in both phases, without compromising its effectiveness as we have seen earlier. In particular, Phase I of PRDualRank (*i.e.*, pattern search) is fast enough to make offline suggestions to a content query system such as DoCQS [109].

2.6.3 Precision and Recall

In this experiment, we demonstrate the validity of our metrics—probabilistic precision and recall. Previously, we used a number of top patterns (of both classes) by F-score to achieve a balance between the two metrics. We now compare the extractions for `capital` relation, either using top K patterns by probabilistic precision, or top K patterns by probabilistic recall. With respect to the ground truth tuples, we compute the precision and recall of the extractions. The results are presented in Figure 2.15 for different values of K.

It is not surprising that using top patterns by probabilistic precision results in better precision of the extracted tuples, as shown in Figure 2.15(a). Similarly, using top patterns by probabilistic recall captures more ground truth tuples and thus achieves better recall, as illustrated in Figure 2.15(b). This experiment validates that our proposed metrics of probabilistic precision and recall "work as intended."

(a) Precision evaluation of the extractions



(b) Recall evaluation of the extractions

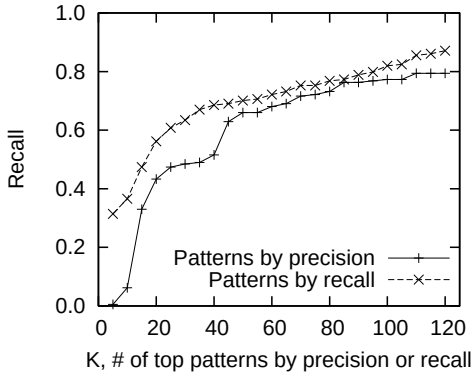


Figure 2.15: Using top K patterns by precision versus by recall.

(a) Search width (b) Search depth (c) Number of extraction patterns

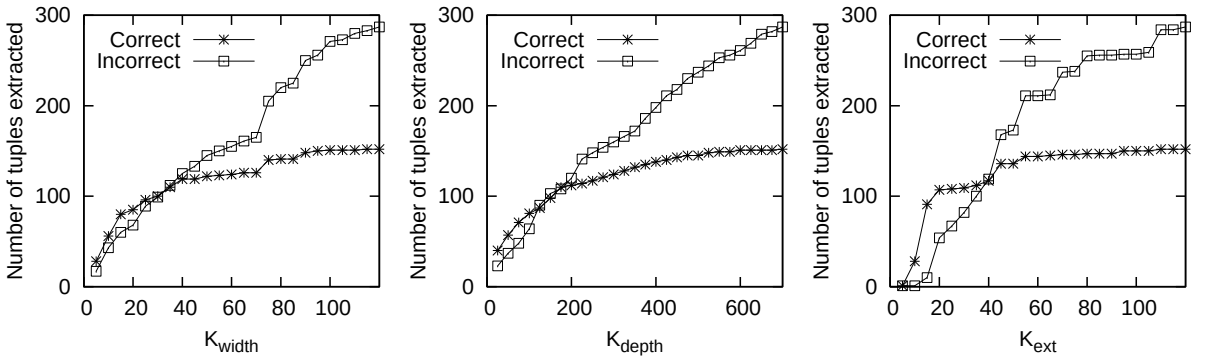


Figure 2.16: Effect of parameters on PRDualRank.

2.6.4 Effects of Parameters

Lastly, we experiment with the parameters K_{width}, K_{depth} and K_{ext} to study their effects on tuple extraction. We vary each parameter while assigning the other two their respective default values (as in Section 2.6.1). When tuning each parameter in a run, the number of correct and incorrect tuples are counted. The results for `capital` relation are shown in Figure 2.16, which clearly demonstrate that the number of correct tuples extracted would converge as the parameters increase. Hence, our approach is not sensitive to these parameters as long as they are reasonably large.

2.7 Conclusion

In this chapter, we supported extraction with the objective metrics of precision and recall. In particular, we formulate the dual problems of pattern search and tuple extraction, which can be co-optimized for both precision and recall. To solve the problems in a principled way, we formally "rediscovered" PR Duality through the conceptual model PRDualRank, which infers probabilistic precision and recall for both tuples and patterns probabilistically on a graph. Interestingly, probabilistic precision and recall can be mapped to two symmetric and "opposite" forms of random walk, namely the backward and forward walks, respectively. We further developed a concrete framework to partially materializes the tuple-pattern graph. Experiments show that our graph-based approach greatly outperforms the previous state-of-the-art system.

Chapter 3

Ranking with Importance and Specificity

In this chapter, we propose to enrich the task of ranking with the senses of *importance* and *specificity*. We focus on the ranking of objects on a graph with respect to a given query node. We perceive each directed edge on the graph as a "citation" or "endorsement," unifying the senses of importance and specificity as two symmetric forms of random walk—the forward and backward walks. As most ranking scenarios require a balance between both senses, we propose RoundTripRank to integrate forward and backward random walks into a coherent round trip. We further employ a scheme of hybrid random surfers and propose RoundTripRank+, enabling a flexible trade-off between importance and specificity. Finally, towards real-time ranking, we develop an efficient and scalable top-K algorithm to evaluate RoundTripRank and RoundTripRank+.

3.1 Introduction

We study the problem of ranking nodes on a graph by their "proximity" to some given query. Consider a graph $G = (V, E)$ with nodes V and edges E. Edges are directed and possibly weighted, where an undirected edge is treated as bidirectional. As each edge embeds certain semantic relationship, through these edges the proximity of two nodes on the graph can be quantified. Thus, given a *query node* $q \in V$, how to measure the *proximity* of every node $v \in V$ to q? More generally, a query can comprise multiple nodes $Q \subset V$.

Let us motivate graph-based proximity with some example tasks on two graphs, including a bibliographic network (which connects authors, papers, terms and venues) and a search

(a) Importance-based	(b) Specificity-based	(c) Balanced
SIGMOD	Spatio-Temporal Databases	VLDB
VLDB	Spatio-Temporal Data Mining	Spatio-Temporal Databases
ICDE	Temporal Aspects in Info. Sys.	ACM GIS

Figure 3.1: Top venues for "spatio temporal data."

log graph (which links search phrases and their clicked URLs). These examples illustrate quite varying ranking scenarios—of different *implicit* preferences, as we will see—even on the same graph.

- *Task A (Expert reviewer):* on a bibliographic network, given a paper, who are the experts best suited to review it?
- *Task B (Matching venue):* on a bibliographic network, given some terms as a topic, what are the matching venues?
- *Task C (Relevant URL):* on a search log graph, given a search phrase, what are the relevant URLs?
- *Task D (Equivalent search):* on a search log graph, given a search phrase, what are the equivalent phrases for the same concept (*e.g.*, "google mail" and "gmail")?

Taking Task B as an example, given a query, say $q =$ "spatio temporal data" (comprising three term nodes), what are the matching venues? One effective family of techniques build upon random walks on the graph, such as Personalized PageRank (PPR) [81, 59] and its variants [10, 80], which find venues that can be easily reached from q through the edges. Figure 3.1(a) shows venues found by PPR in our actual experiments—they are well-known venues like VLDB where the topic in q appears. As pioneered by PPR, ranking nodes by their *reachability from* the query intuitively captures the sense of *importance*.

However, such importance-based ranking is not ideal in many cases, as a node can be easily reached from q simply due to its popularity for being linked from a broad range of nodes, without being particularly tailored to q. As we re-examine Figure 3.1(a), the venues

are only categorically related to our query as general data-centric topics. In fact, the same venues could be ranked as important for almost any data-centric topic, such as "streaming data" or "information integration." How about those venues especially tailored to the query, as Figure 3.1(b) shows?

Thus, we argue that the sense of *specificity* is missing, as the results in Figure 3.1(a) are important, but not specific, to the query. As a key insight, proximity shall naturally encompass the dual senses of importance and specificity. A balanced ranking in the two senses is often more useful, such as Figure 3.1(c). Among the venues, VLDB is important, Spatio-Temporal Databases is specific, and the previously undiscovered ACM GIS is balanced between the two senses.

Although intuitive and appealing, dual-sensed graph proximity with both importance and specificity is not explored until recently by Hristidis *et al.* [56]. Their work hypothesizes various forms of specificity, including the inverse of node degree, the inverse of global ObjectRank [10], and Inverse ObjectRank [56] (which is ObjectRank on the graph with reversed edges). Each form of specificity is then combined with the reachability-based importance in different ways. Despite the novel insight, their particular approach is hindered by two fundamental drawbacks.

First, they treat the two senses independently and combine them heuristically, *lacking a unifying model* to coherently capture both. As they model specificity upon quite disparate concepts, there is no definitive view on the best form. Moreover, it is unclear which way of combination with importance is superior. In particular, some forms of specificity (*e.g.*, inverse of node degree [56]) may inherently differ from the reachability-based importance, and thus their combination appears ad hoc.

Second, they pursue a *fixed trade-off* between importance and specificity, but many applications require flexible or customizable trade-offs, since different users or ranking tasks often implicitly assume different preferences towards one of the senses. Let us revisit the example tasks given earlier.

- *Task A (Expert reviewer)*: Reviewers balanced between importance and specificity are preferred. A very important expert may only commit limited time, whereas a very specific researcher like a graduate student may lack authoritativeness.

- *Task B (Matching venue)*: Different scenarios require varying senses. For instance, to build some background on a topic, a specific book chapter may be preferred. In contrast, to submit one's best work, important venues are often sought after.

- *Task C (Relevant URL)*: Users often prefer important URLs for monetary transactions, such as booking a hotel online.

- *Task D (Equivalent search)*: Equivalent phrases are inherently specific to each other, as they ideally represent the same concept.

We stress that it is often rare to involve only a single sense in a task. The above tasks actually require some trade-off between the two senses. For instance, Task C values importance, but it still needs some specificity—for booking a hotel, an important "travel" site is expected. However, what sense weighs more varies across tasks as we have analyzed. We note that the benefit of a flexible trade-off has not been recognized in previous studies, and thus they miss the potential to improve ranking by catering to different ranking needs.

Thus, as our main hypothesis, we believe that an effective dual-sensed graph proximity measure must entail two aspects in the following.

- **Unified modeling.** Instead of combining the independent forms of importance and specificity, the dual senses must be seamlessly integrated under a single unifying process—coherently based on the same random walk principle as the well-studied importance-based ranking.

- **Customizable trade-off.** Instead of a "one-size-fits-all" solution, the trade-off between importance and specificity must be flexible—the unified modeling shall also allow for customizable trade-offs between the dual senses.

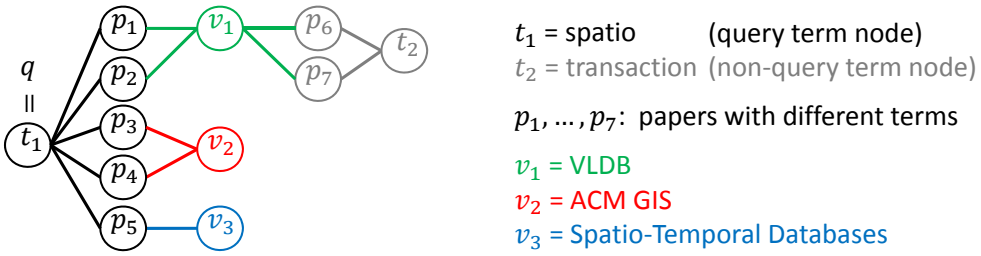


Figure 3.2: Toy graph for a bibliographic network.

To begin with, towards a unified modeling, we propose RoundTripRank to integrate the dual senses of importance and specificity in a round trip. Given a node q as the query, a *round trip*, with respect to a target node v, starts from q, goes "forward" to reach v, and then from v goes "backward" to finally return to q. As we believe importance and specificity naturally exist in symmetric forms, this round trip generalizes the reachability-based importance, capturing not only the reachability from the query (to the target) as importance, but also the reachability to the query (from the target) as specificity.

As an example, let us rank venues v_1, v_2, v_3 on Figure 3.2 for query $q = t_1$. v_1 is intuitively more important than v_3, as v_1 has two papers p_1, p_2 on t_1, but v_3 has only one p_5. Observe that from t_1 it is easier to reach v_1 than v_3. However, v_1 is less specific than v_3, as v_1 accepts two off-topic papers p_6, p_7, but v_3 does not. Observe that it is easier to return to t_1 from v_3 than from v_1. Thus, a round trip—from t_1, through a target venue v_i, and back to t_1—integrates both senses, favoring v_2 for being not only important (with two papers p_3, p_4 about t_1), but also specific (without off-topic papers).

Next, towards a customizable trade-off, we make a further generalization and develop RoundTripRank+ based on a novel scheme of *hybrid random surfers*. We consider random surfers of different minds who may shortcut different parts of the round trip to reflect their preference—some prefer importance, some prefer specificity, and others prefer a balance. In particular, we use different compositions of hybrid random surfers to mimic the varying objectives in importance and specificity.

A potential drawback of RoundTripRank is its computational overhead. Each "forward

trip" from a query node to every target node can be conceptually paired with many different "backward trips" to complete a round trip, causing an exponential blow-up to the possible round trips. Fortunately, we are able to decompose RoundTripRank into smaller units, where each unit can be computed without actually considering the large number of round trips. Moreover, since users are often interested in getting a small number of top results quickly, we introduce a top-K algorithm called 2SBound, building upon our decomposition of RoundTripRank as well as a novel two-stage bounds updating framework. To handle very large graphs, we further adapt 2SBound to a distributed environment.

Finally, we conduct extensive experiments on two real-world graphs. For effectiveness, we consistently and significantly outperform existing baselines over varying ranking tasks that require different trade-offs between importance and specificity. On average, RoundTripRank improves over existing mono-sensed measures by 10%, and RoundTripRank+ improves over existing dual-sensed measures by 7%. For efficiency, 2SBound is 2–10 times faster than various baseline solutions.

In summary, this chapter makes the following contributions.

- **Unification**: We unify both importance and specificity with reachability-based formulations, and propose RoundTripRank to integrate both senses into a single *coherent round trip process*.
- **Flexibility**: We develop RoundTripRank+ based on a scheme of hybrid random surfers, enabling a *customizable trade-off* between importance and specificity.
- **Computation**: We design an *efficient and scalable* top-K algorithm 2SBound to evaluate RoundTripRank and RoundTripRank+ in real time.
- **Empirical results**: We conduct *extensive experiments* to demonstrate the accuracy and efficiency of our approaches.

3.2 Related Work

Importance and specificity. While the sense of importance to a query has been studied since PPR [81, 59], the sense of specificity is only recently explored [56] with a few heuristic forms such as the inverse of node degree, the inverse of global ObjectRank [10], and Inverse ObjectRank, which are generally formulated independently of importance. We pursue a different approach by generalizing the importance-based random walk to a round trip, which coherently captures both importance and specificity in a single measure.

Mono-sensed proximity. Most existing graph-based proximity measures are *mono-sensed*, matching the query in only one sense—importance, specificity, or simply a vague sense of "proximity" without a finer interpretation. Examples include (a) importance: PPR [81, 59] and its variants [10, 80, 2, 29] (also called forward random walk); (b) specificity: backward random walk [2, 29] and Inverse ObjectRank [56]; (c) no finer interpretation: AdamicAdar [1], SimRank [58] and escape probability [68, 102]. As Sect. 3.1 motivated, they are inadequate since most tasks require some trade-off between importance and specificity.

Dual-sensed proximity. Some recent efforts, such as truncated commute time [88], the harmonic mean of precision and recall [2] (also investigated in Chapter 2), and ObjSqrtInv [56], can be deemed *dual-sensed* measures. We first note that none of them, except [56], connects explicitly to importance and specificity. While [88] does not argue any finer interpretation, [2] and [37] measure probabilistic precision and recall. Unlike importance and specificity which are directly defined on a query, precision and recall are indirectly measured against an underlying "relevant set" of nodes for a query. Despite the contrast, precision and recall seem to intuitively parallel specificity and importance, which may warrant further investigation. Second, most studies [2, 37, 56] heuristically combine the two senses as two independent measures, lacking an underpinning model to unify them coherently. We instead develop a round trip to directly entail both senses in a single measure. Third, all of them

ignore different trade-offs between importance and specificity across tasks. In contrast, our round trip model can be generalized to mimic the varying trade-offs using a scheme of hybrid random surfers, allowing for a customizable trade-off.

Efficient query processing. We also study online top-K processing for RoundTripRank. While we adopt a branch-and-bound graph expansion algorithm [90] as the backbone, significant innovations are still required to realize it, which include our original bounds decomposition, two-stage framework for bounds updating, and a cluster-based distributed architecture for scaling up. Our realization not only enables online search, but also requires no precomputation mandated by many other graph proximity algorithms [59, 10, 23, 51, 110].

3.3 RoundTripRank: Walking in Round Trips

In this section, we develop RoundTripRank to integrate importance and specificity in a coherent round trip, followed by a basic computational model.

3.3.1 Balancing Importance and Specificity

Towards a dual-sensed measure, we generalize PPR [81, 59] for its effectiveness in measuring importance. The generalization would capture both importance and specificity in a coherent random walk, instead of treating them independently and combining them heuristically.

Background: Personalized PageRank (PPR). We first review PPR [81, 59] as an effective measure of importance. On a graph $G = (V, E)$, a random surfer is initially at a given query node $q \in V$. (We ignore multi-node queries for now.) At each step, she has a probability $1 - \alpha$ to move to a neighbor randomly, and a probability α to teleport to q. Her stationary probability at node v is v's PPR for q, indicating v's importance to q.

To generalize PPR for capturing specificity as well, we need an alternative view with the notions of a "trip" and a "target." Let L be a random variable (of some distribution).

Starting from the query node q, the random surfer takes L random steps on the graph. This L-step walk is a *trip* from q, reaching v as the *target*. We call the probability that v is the target of a trip from q the *Forward Rank* or F-Rank of v for q, denoted $f(q, v)$. (Imagine that the random surfer is walking "forward" from q to reach v, in contrast to the "backward" walk as we will see soon.) Representing a trip as a sequence of visited nodes $W_0, \ldots, W_L$ on the graph, we formalize F-Rank below:

$$f(q, v) \triangleq p\left(W_L = v | W_0 = q\right). \tag{3.1}$$

The walk length L captures the range of influence from the starting node—how far the surfer can walk until the influence becomes too weak and a restart is necessary. In particular, if L is geometric with parameter $\alpha \in (0, 1)$, *i.e.*, $p(L = \ell) = (1 - \alpha)^\ell \alpha$, F-Rank is equivalent to PPR as shown below, which has been established elsewhere [43].

PROPOSITION 3.1 (F-RANK AND PPR EQUALITY [43]). Given the same query node q, any node v's PPR with teleporting probability α equals its F-Rank with a geometrically distributed walk length $L \sim \text{Geo}(\alpha)$.

In general, a geometric L is effective as it gives longer walk lengths smaller probabilities, agreeing with the intuition that the influence from a node weakens along a path. Unless otherwise stated, we will assume a geometric $L \sim \text{Geo}(\alpha)$ for F-Rank, which captures importance just as PPR does.

Generalization: from importance to specificity. F-Rank treats importance as reachability from q in the L steps before teleportation. We can interpret this notion of importance by viewing a directed edge $a \to b$ as "a cites b." Note that, more generally, depending on the context "a cites b" may carry different semantics, such as a (paper) mentions b (term), or a (venue) accepts b (paper), or a (paper) contributes to b (venue). Naturally, if a node v is more important to q than another node v' is, q is more likely to cite v than v', directly or indirectly. In other words, the more likely v can be reached from q via a directed path, the

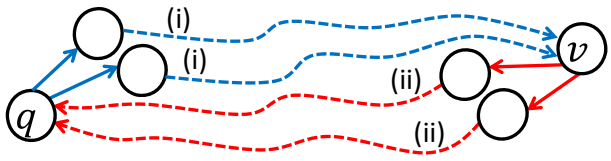
more important v is to q. As an example, in Figure 3.2 where $q = t_1$, observe that v_1 or v_2 (each with two papers about t_1) is intuitively more important to t_1 than v_3 (with only one such paper). Indeed, from q it is easier to reach v_1 or v_2 than v_3.

Interestingly, the citation analogy enables us to treat specificity in a symmetric form to importance. Imagine two nodes v and v', where v is more specific to q than v' is. Naturally, v tends to cite a focused set of nodes which are particularly tailored to q, whereas v' tends to cite a diverse set of nodes which may not match q at all. Hence, q is more likely to be cited by v than by v', directly or indirectly. In other words, the more likely q can be reached from v via a directed path, the more specific v is to q. In Figure 3.2 where $q = t_1$, v_2 or v_3 (accepting no off-topic papers) is intuitively more specific than v_1 (accepting two such papers). Observe that it is more likely to reach t_1 from v_2 or v_3 than from v_1.

Integration: RoundTripRank. To integrate the sense of specificity, upon reaching the target node v from q in L steps, the random surfer takes an additional L' steps, trying to return to q. The more likely to return to q from v, the more specific v is to q. Hence, the sense of specificity is embedded into these additional L' steps, symmetric to importance embedded in the first L steps.

Thus, we consider a round trip starting and ending at q, through v, as depicted in Figure 3.3. Intuitively, if v is important to q, the surfer will easily reach v from q in L steps (we say the forward trip); once at v, if v is specific to q, the surfer will easily return to q in L' steps (we say the backward trip). By finding out how likely a round trip goes through v, we capture both senses in one coherent random walk. In the following, we formally introduce the round trip concept and define RoundTripRank.

DEFINITION 3.1 (ROUND TRIP). A *round trip* is a random walk of $L+L'$ steps starting and ending at the same node, *i.e.*, $W_0 = W_{L+L'}$ (L, L' are i.i.d. geometric random variables). The node after the first L steps, W_L, is the *target* of the round trip.



(i) How important?
Reaching v from q (forward).

(ii) How specific?
Returning to q from v (backward).

Figure 3.3: Illustration of round trips. A dotted line indicates a random walk path that may pass through other nodes not shown in the figure.

DEFINITION 3.2 (ROUNDTRIPRANK). *RoundTripRank* of a node v for a query node $q \in V$, denoted $r(q, v)$, is defined as: given that a random surfer starting at q completes a round trip, the probability that this round trip has the target v.

$$r(q, v) \triangleq p\left(W_L = v | W_0 = W_{L+L'}, W_0 = q\right). \tag{3.2}$$

More generally, a query can consists of a set of nodes $Q \subset V$, and the round trip can start from any $q \in Q$. Then, RoundTripRank for Q, denoted $r(Q, v)$, is defined as

$$r(Q, v) \triangleq p\left(W_L = v | W_0 = W_{L+L'}, W_0 \in Q\right). \tag{3.3}$$

Similar to the *Linearity Theorem* [59] for PPR, RoundTripRank for a multi-node query $Q \subset V$ can be expressed as a linear function of RoundTripRank for each $q \in Q$, as established in the following proposition. Note that $\sum_{q \in Q} \lambda(q) = 1$, and each $\lambda(q) \in (0, 1)$. Interestingly, $\lambda(q)$ can be used to encode the user's desired weighting for q. Given this linearity property, the rest of our discussion in this chapter only considers a single-node query $q \in V$, without loss of generality.

PROPOSITION 3.2 (LINEARITY OF ROUNDTRIPRANK). Given a set of nodes $Q \subset V$ as the query, let $\lambda(q) \triangleq p\left(W_0 = q | W_0 = W_{L+L'}, W_0 \in Q\right), \forall q \in Q$. Then,

$$r(Q, v) = \sum_{q \in Q} \lambda(q) r(q, v). \tag{3.4}$$

To illustrate RoundTripRank, we revisit the toy graph in Figure 3.2. For simplicity, we assume a constant walk length $L = L' = 2$, and equal weights for all edges. Applying Bayes' law to Eq. 3.2, RoundTripRank of each target v for the query node t_1 can be computed as

Target v	**Round trip from** t_1	**Probability**	**RoundTripRank** $r(t_1, v)$
v_1	$t_1 \to p_1 \to v_1 \to p_1 \to t_1$	0.0125	$\propto 0.0125 \times 4 = 0.05$
	$t_1 \to p_1 \to v_1 \to p_2 \to t_1$	0.0125	
	$t_1 \to p_2 \to v_1 \to p_1 \to t_1$	0.0125	
	$t_1 \to p_2 \to v_1 \to p_2 \to t_1$	0.0125	
v_2	$t_1 \to p_3 \to v_2 \to p_3 \to t_1$	0.025	$\propto 0.025 \times 4 = 0.1$
	$t_1 \to p_3 \to v_2 \to p_4 \to t_1$	0.025	
	$t_1 \to p_4 \to v_2 \to p_3 \to t_1$	0.025	
	$t_1 \to p_4 \to v_2 \to p_4 \to t_1$	0.025	
v_3	$t_1 \to p_5 \to v_3 \to p_5 \to t_1$	0.05	$\propto 0.05$
t_1	$t_1 \to p_1 \to t_1 \to p_1 \to t_1$	0.01	$\propto 0.01 \times 25 = 0.25$
	24 more ...	0.01 each	
others	none	0	0

Figure 3.4: RoundTripRank for the bibliographic toy graph.

follows, where the denominator is independent of v and thus does not affect ranking (let $\propto$ represent a monotonic transformation).

$$r(t_1, v) = \frac{p\left(W_0 = W_{L+L'}, W_0 = t_1, W_L = v\right)}{p\left(W_0 = W_{L+L'}, W_0 = t_1\right)}$$
$$\propto p\left(W_0 = W_{L+L'}, W_0 = t_1, W_L = v\right), \tag{3.5}$$

In other words, it is proportional to the sum of probabilities of round trips starting from t_1 with target v. The probability of a single round trip can be found using node degrees, *e.g.*, $p\left(t_1 \to p_1 \to v_1 \to p_1 \to t_1\right) = \frac{1}{5} \cdot \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} = 0.0125$. The detailed computation for each target node is shown in Figure 3.4, where all round trips starting from t_1 are listed and grouped by their target nodes.

In particular, let us compare v_1, v_2, v_3. As discussed earlier, v_2 is as important as but more specific than v_1, and v_2 is as specific as but more important than v_3. Hence, v_2 has larger RoundTripRank than v_1 and v_3 for being both important and specific. Also note t_1 itself has the largest RoundTripRank, which is intuitive that self-proximity is high.

3.3.2 Basic Computational Model

In Figure 3.4, we compute RoundTripRank by summing up the round trips for each target. However, the number of such round trips increases exponentially on larger graphs, calling for a more practical computational model.

To begin with, we examine how PPR/F-Rank can be computed. One simple method applies iterative computation [81], which is linear in the number of nodes and edges:

$$f^{(i+1)}(q,v) = \alpha I(q,v) + (1-\alpha) \sum_{v' \in \text{In}(v)} M_{v'v} f^{(i)}(q,v'), \tag{3.6}$$

where $I(q,v) = 1$ if $q = v$, or 0 if $q \neq v$; $\text{In}(v)$ is the set of v's in-neighbors; $M_{v'v}$ is the one-step transition probability from v' to v; α is the teleporting probability. Convergence is guaranteed on an irreducible and aperiodic graph [52].

Thus, we wonder if RoundTripRank can also be computed iteratively. Simply put, iterative computation is possible for F-Rank as it is recursive in nature—to reach a node v from q, the surfer must first reach one of v's in-neighbors v'. In other words, as the target v is at an "end" of a walk, we can reduce the walk by one step to utilize the F-Rank of v'. However, in RoundTripRank, a round trip merely passes through v which is not at an "end" of the walk, and thus no apparent iterative computation exists.

Consequently, we resort to divide and conquer. We decouple a round trip into smaller units with the target at the "end" of each unit, such that we can iteratively compute each. It turns out that we can rewrite RoundTripRank defined in Eq. 3.2 into an equivalent form with two decoupled units.

Proposition 3.3 (Decomposition of RoundTripRank). The RoundTripRank of v for a query node q can be expressed as follows:

$$r(q,v) \propto p\left(W_L = v | W_0 = q\right) p\left(W_{L'} = q | W_0 = v\right) \tag{3.7}$$

On the one hand, $p\left(W_L = v | W_0 = q\right)$ is the F-Rank (Eq. 3.1), *i.e.*, the reachability from

q to v during the forward trip. On the other hand, $p\,(W_{L'} = q|W_0 = v)$ is the reachability from v to q during the backward trip, which we call the *Backward Rank* or *B-Rank* of v for q, denoted $b(q, v)$. Hence,

$$r(q, v) \propto f(q, v)b(q, v) \tag{3.8}$$

While F-Rank can be computed iteratively (see Eq. 3.6), B-Rank can be computed in a symmetric way, where $\mathrm{Out}(v)$ is the set of v's out-neighbors:

$$b^{(i+1)}(q, v) = \alpha I(q, v) + (1 - \alpha) \sum_{v' \in \mathrm{Out}(v)} M_{vv'} b^{(i)}(q, v'). \tag{3.9}$$

As a minor caveat, if there is at least one directed path from q to v but not from v to q, we obtain $f(q, v) > 0$ and $b(q, v) = 0$. Hence, it may be counter-intuitive that $r(q, v) = 0$ no matter how large $f(q, v)$ is. Fortunately, this does not happen on an irreducible (*i.e.*, strongly connected) graph, which is a prerequisite for convergence. In practice, we can always make a graph irreducible by adding some dummy edges [52].

3.4 RoundTripRank+: Customizing the Trade-off

Although RoundTripRank balances importance and specificity, different tasks often require varying trade-offs, as Section 3.1 motivated. To this end, we propose RoundTripRank+.

3.4.1 Hybrid Random Surfers

To reflect different user objectives over importance and specificity, consider some *hybrid random surfers* Ω consisting of different-minded surfers in three disjoint groups: (a) Ω_{11}, seeking targets balanced between importance and specificity, *e.g.*, ACM GIS for $q =$ "spatio temporal data"; (b) Ω_{10}, seeking important targets, *e.g.*, VLDB for q; (c) Ω_{01}, seeking specific targets, *e.g.*, Spatio-Temporal Databases for q.

These surfers potentially take variants of round trips by "shortcutting", as follows. Given

a query node q and a target v, the surfers in Ω_{11} take regular round trips. However, the surfers in Ω_{10}, after reaching v in L steps, shortcut the next L' steps by teleporting to q, *i.e.*, $\forall \omega \in \Omega_{10}, p\left(W^{\omega}_{L+L'} = q | W^{\omega}_{L} = v\right) = 1$ where $\{W^{\omega}_{\ell} : \ell \geq 0\}$ represents the sequence of nodes visited by ω. Likewise, the surfers in Ω_{01} shortcut the first L steps by teleporting to v, *i.e.*, $\forall \omega \in \Omega_{01}, p\left(W^{\omega}_{L} = v | W^{\omega}_{0} = q\right) = 1$.

Using different compositions of Ω, *i.e.*, different distributions of the three groups, we can customize the trade-off between importance and specificity with RoundTripRank+.

DEFINITION 3.3 (ROUNDTRIPRANK+). With respect to some hybrid random surfers Ω, *RoundTripRank+* of v for a query node q, denoted $r_{\Omega}(q, v)$, is defined as follows. Given that each random surfer in Ω *independently* completes a round trip from q with a common target, $r_{\Omega}(q, v)$ is the probability that the common target is v:

$$r_{\Omega}(q, v) \triangleq p\left(x = v | \forall_{\omega \in \Omega} : W^{\omega}_{0} = W^{\omega}_{L+L'} = q, W^{\omega}_{L} = x\right) \tag{3.10}$$

We illustrate the customizability of RoundTripRank+ with some special cases. (a) $\Omega = \Omega_{11}$, *i.e.*, $\Omega_{10} = \Omega_{01} = \emptyset$. It reduces to (the non-customizable) RoundTripRank for a balance between the two senses, since all the surfers only walk in regular round trips. (b) $\Omega = \Omega_{10}$ reduces to F-Rank and captures only importance, since all the surfers teleport back to q from v. (c) Likewise, $\Omega = \Omega_{01}$ reduces to B-Rank and captures only specificity.

However, to customize the trade-off we need to adjust the compositions of Ω, which involves three parameters $|\Omega_{11}|, |\Omega_{10}|, |\Omega_{01}|$. To simplify it, we will introduce a computational model with only one parameter.

3.4.2 Basic Computational Model

We can decompose RoundTripRank+ into an equivalent form, similar to Proposition 3.3.

PROPOSITION 3.4 (DECOMPOSITION OF ROUNDTRIPRANK+). With respect to some hybrid random surfers Ω consisting of three groups $(\Omega_{11}, \Omega_{10}, \Omega_{01})$, let $\beta \triangleq \frac{|\Omega_{11}|+|\Omega_{01}|}{|\Omega|+|\Omega_{11}|}$. Then, the RoundTripRank+ of v for a query node q can be expressed as follows without altering ranking:

$$r_\Omega(q,v) \propto r_\beta(q,v) \triangleq f(q,v)^{1-\beta} \cdot b(q,v)^\beta. \tag{3.11}$$

Intuitively, $\beta \in [0,1]$ is the fraction of objectives by all surfers that are specificity (note that each surfer in Ω_{11} has two objectives). We call β the *specificity bias*—a large β favors specificity over importance. A unique ranking can be determined by a given β, which is itself uniquely determinable from any given hybrid surfers Ω. Hence, we can adjust β directly to customize the trade-off between importance and specificity. That is, we compute RoundTripRank+ as $r_\beta(q,v)$ in Proposition 3.4, for a chosen β. Choosing β for each ranking task enables a customized trade-off, which corresponds to, as its physical meaning, adjusting the composition of hybrid surfers. As special cases, $\beta = 0$ (or 1) reduces to F-Rank (or B-Rank) for only importance (or specificity); $\beta = 0.5$ reduces to RoundTripRank.

To obtain the optimal β for a ranking task, we may use some development queries, or consult domain experts. Users may also specify β directly. As a last resort, we can always fall back to the default $\beta = 0.5$, which generally outperforms the extreme cases of $\beta = 0$ or 1 as demonstrated in our experiments, since most real-world ranking scenarios require both importance and specificity.

3.5 Online Top-K Processing

The basic computational models discussed earlier require computing F-Rank and B-Rank. However, their iterative computation in Eq. 3.6 and 3.9 is not scalable, as it involves multiple passes of the entire graph for each query. Fortunately, in many applications, users only need to quickly get a small number of top results, which could be a close approximation.

Input: graph G; query node q; number of desired results K
Output: top-K ranking T_K of nodes v on G by $r(q, v)$

1 Neighborhood $S \leftarrow \emptyset$;
2 **repeat**
3 | Two-stage bounds updating framework:
4 | | *Stage I.* Expand S and initialize bounds Δ;
5 | | *Stage II.* Iteratively refine Δ over S;
6 | $T_K \leftarrow$ current top-K by lower bounds;
7 **until** T_K satisfies top-K conditions;
8 **return** T_K.

Figure 3.5: 2SBound algorithm: Two-Stage Bounding for RoundTripRank.

Hence, we propose an online approximate top-K method 2SBound, followed by a distributed solution to handle larger graphs. Our discussion only covers RoundTripRank, but extending to RoundTripRank+ is straightforward.

3.5.1 Approximate Top-K Processing: 2SBound

We propose *Two-Stage Bounding*, or 2SBound, for online top-K processing. As its backbone, we adopt the standard branch-and-bound expansion on graphs [90]. However, its realization for RoundTripRank still requires significant innovations, hinging on our original basic computational model and two-stage bounds updating framework.

We outline 2SBound in Figure 3.5. Given a query node q, we maintain a *neighborhood* S, which is a subset of the nodes: $S \subseteq V$. We refer to those nodes currently in S as *seen* nodes, and those outside S as *unseen* nodes. We also maintain a set of bounds Δ for RoundTripRank: (a) each seen node is sandwiched by an upper bound $\hat{r}(q, v)$ and a lower bound $\check{r}(q, v)$, *i.e.*, $\check{r}(q, v) \leq r(q, v) \leq \hat{r}(q, v)$, $\forall v \in S$; (b) all unseen nodes have a common *unseen* upper bound $\hat{r}(q)$, *i.e.*, $r(q, v) \leq \hat{r}(q)$, $\forall v \notin S$. Starting with an empty neighborhood S (line 1), we repeatedly expand S and update the bounds Δ (line 2–5). After each expansion and updating, we obtain a candidate top K ranking T_K according to their lower bounds (line 6). The process stops if T_K satisfy the top-K conditions (line 7).

To materialize the above algorithm, we first need to discuss the top-K conditions (Section 3.5.1.1). Furthermore, we must solve the core problem of bounds updating, which relies on our bounds decomposition technique (Section 3.5.1.2) and two-stage bounds updating framework (Section 3.5.1.3).

3.5.1.1 Top-K Conditions

After each neighborhood expansion and updating, we try to decide the top-K nodes if $|S| \geq K$. The nodes in S are sorted as $v_1, \ldots, v_{|S|}$ such that $\check{r}(q, v_1) \geq \ldots \geq \check{r}(q, v_{|S|})$. $T_K = \langle v_1, \ldots, v_K \rangle$ is a candidate top-K ranking, which will be further judged by two conditions below, assuming $\epsilon = 0$ for the moment. First, Eq. 3.12 ensures that T_K contains correct top K nodes (in no particular order), since their lower bounds are no less than the largest upper bound of all other nodes. Second, Eq. 3.13 ensures that T_K is ordered correctly.

$$\check{r}(q, v_K) > \max\left\{\hat{r}(q, v_{K+1}), \ldots, \hat{r}(q, v_{|S|}), \hat{r}(q)\right\} - \epsilon \tag{3.12}$$

$$\check{r}(q, v_i) > \hat{r}(q, v_{i+1}) - \epsilon, \quad \forall i \in \{1, \ldots, K-1\} \tag{3.13}$$

In many applications, it is desirable to speed up the computation by slightly sacrificing ranking quality. Hence, we relax the conditions in Eq. 3.12–3.13 with a positive *slack* parameter ϵ. Accordingly, we obtain an ϵ-approximate top-K ranking, which (a) does not miss any node whose score exceeds v_K's by at least ϵ; (b) does not swap the order of two nodes if their scores differ by at least ϵ.

3.5.1.2 Bounds Decomposition

Our computational model (Eq. 3.8) implies that the bounds for RoundTripRank can be computed based on the bounds for F-Rank and B-Rank. Thus, we maintain two neighborhoods: the f-neighborhood S_f for F-Rank, and the b-neighborhood S_b for B-Rank. In general $S_f \neq S_b$, because their expansions differ as we shall see. We ultimately define the r-neighborhood S for RoundTripRank using S_f and S_b.

Specifically, $\forall v \in S_f$, let $\hat{f}(q,v)$ and $\check{f}(q,v)$ denote the upper and lower bounds for F-Rank. Similarly, $\forall v \in S_b$, let $\hat{b}(q,v)$ and $\check{b}(q,v)$ denote the upper and lower bounds for B-Rank. By defining the r-neighborhood as the set of nodes common in both the f- and b-neighborhoods, *i.e.*, $S = S_f \cap S_b$, we can compute the bounds for RoundTripRank, $\forall v \in S$:

$$\check{r}(q,v) = \check{f}(q,v)\check{b}(q,v), \tag{3.14}$$

$$\hat{r}(q,v) = \hat{f}(q,v)\hat{b}(q,v). \tag{3.15}$$

Additionally, to compute the unseen upper bound $\hat{r}(q)$, we also maintain its counterparts $\hat{f}(q)$ and $\hat{b}(q)$ for F-Rank and B-Rank, respectively. However, $\hat{r}(q)$ does not simply decompose as $\hat{f}(q)\hat{b}(q)$. Since $S = S_f \cap S_b$, some unseen nodes by S may be "seen" by either S_f or S_b, but not both. These nodes $v \in S_f \backslash S$ or $v \in S_b \backslash S$ have individual upper bound $\hat{f}(q,v)$ or $\hat{b}(q,v)$. Hence, we can compute $\hat{r}(q)$ as follows:

$$\hat{r}(q) = \max\left\{\hat{f}(q)\hat{b}(q), \max_{v \in S_f \backslash S} \hat{f}(q,v)\hat{b}(q), \max_{v \in S_b \backslash S} \hat{f}(q)\hat{b}(q,v)\right\} \tag{3.16}$$

3.5.1.3 Two-Stage Bounds Updating Framework

As just explained, to obtain the bounds for RoundTripRank, we only need those for F-Rank and B-Rank. Given the graph and the current neighborhood (S_f or S_b) as input, we propose a novel two-stage framework common to both S_f and S_b, although their realizations are different. In particular, we update bounds for each seen node by considering not only the node itself for *initialization* (Stage I), but also its relationship with its neighbors for *iterative refinement* (Stage II).

To simplify notations in the common framework, let S_x denote either the f- or b-neighborhood, *i.e.*, x refers to either f or b. Likewise, $\check{x}(\cdot)$ and $\hat{x}(\cdot)$ denote the lower and upper bounds for either F-Rank or B-Rank.

Stage I: Expansion and initialization. Expand the given neighborhood S_x. Assign lower and upper bounds $\check{x}^{(0)}(q,v)$ and $\hat{x}^{(0)}(q,v)$ for each $v \in S_x$, and the unseen upper

bound $\hat{x}^{(0)}(q)$ for all other nodes $v \notin S_x$. We are essentially initializing the bounds—as the superscript (0) indicates—for each node individually, before further refining them iteratively using their neighbors in Stage II. To compute these initial values, we can leverage some of the previous studies [15, 51, 89].

Stage II: Iterative refinement. We further improve the initial bounds from Stage I by exploiting the relationships between nodes. As implied by the naïve computation in Eq. 3.6 (or Eq. 3.9), the F-Rank (or B-Rank) of a node v can be expressed in terms of its in- (or out-) neighbors' values. Since Eq. 3.6 and 3.9 are monotonic additions, by using the lower or upper bound of each summand that involves a neighbor v' of v, we will get a lower or upper bound on the overall sum for v. Hence, we can iteratively update the bounds of each seen node using its neighbors. To tighten the bounds, we only decrease an upper bound or increase a lower bound in any update. Subsequently, the initial bounds from Stage I can be iteratively refined over the neighborhood S_x, such that in iteration $i+1$ (where $i = 0$ corresponds to the initialization in Stage I), $\forall v \in S_x$,

$$\check{x}^{(i+1)}(q,v) = \max\left\{\check{x}^{(i)}(q,v), \alpha I(q,v) + (1-\alpha)\sum_{v' \in N_x(v)} X_{vv'}\check{x}^{(i)}(q,v')\right\}, \tag{3.17}$$

$$\hat{x}^{(i+1)}(q,v) = \min\left\{\hat{x}^{(i)}(q,v), \alpha I(q,v) + (1-\alpha)\sum_{v' \in N_x(v)} X_{vv'}\hat{x}^{(i)}(q,v')\right\}, \tag{3.18}$$

where $N_x(v) = \text{In}(v)$, $X_{vv'} = M_{v'v}$ for F-Rank, and $N_x(v) = \text{Out}(v)$, $X_{vv'} = M_{vv'}$ for B-Rank. If a neighbor v' of v is unseen, we simply use a lower bound zero and the unseen upper bound for it. We terminate the iterative refinement when the bounds converge, which is guaranteed because $\{\check{x}^{(i)}(q,v)\}_{i=0}^{\infty}$ and $\{\hat{x}^{(i)}(q,v)\}_{i=0}^{\infty}$ are bounded monotone sequences due to the $\max\{\cdot\}$ and $\min\{\cdot\}$ functions, respectively.

In general, improving the bounds even for a single node may further improve the bounds for its neighbors, and this effect propagates across the neighborhood. As the neighborhood is often far smaller than the entire graph, such iterative refinement is feasible.

Next, we discuss the realizations of F-Rank and B-Rank under the two-stage framework.

Realization of F-Rank. As F-Rank is equivalent to PPR (see Proposition 3.1), we can leverage an existing work on PPR for Stage I, namely the Bookmark-Coloring Algorithm (BCA) [15]. BCA iteratively spreads the "residual" starting from the query node over the graph to form the PPR at each node. It maintains two values for each node $v \in V$ with respect to a query node q: the current estimated PPR score $\rho(q, v)$ and the residual $\mu(q, v)$. Initially, $\forall v, \rho(q, v) = 0$. Moreover, there is initially a total of one unit residual, all of which is concentrated at q, *i.e.*, $\mu(q, q) = 1$ and $\forall v \neq q, \mu(q, v) = 0$. Subsequently, BCA picks the node $v_{\max}$ with the largest residual currently, and performs *BCA processing* on $v_{\max}$:

- α portion of its residual adds to its current estimated PPR, *i.e.*,

$$\rho(q, v_{\max}) \leftarrow \rho(q, v_{\max}) + \alpha \cdot \mu(q, v_{\max}); \tag{3.19}$$

- the remaining $1 - \alpha$ portion spreads to its out-neighbors, *i.e.*,

$$\forall v' \in \mathrm{Out}(v_{\max}), \quad \mu(q, v') \leftarrow \mu(q, v') + (1 - \alpha)\mu(q, v_{\max})M_{v_{\max} v'}; \tag{3.20}$$

- finally, reset its residual to zero, *i.e.*,

$$\mu(q, v_{\max}) \leftarrow 0. \tag{3.21}$$

This procedure—picking $v_{\max}$ and apply BCA processing—is repeated until the total amount of remaining residual becomes zero asymptotically. We refer readers to the original work [15] for full details.

Stage I. We concretize the f-neighborhood as the set of nodes currently with non-zero estimated PPR in BCA: $S_f \triangleq \{v \in V : \rho(q, v) > 0\}$. The precondition of BCA dictates $\rho(q, v) = 0, \forall v \in V$, which results in an initially empty S_f.

As the first step in Stage I, to expand S_f, instead of picking one node $v_{\max}$ with maximal residual as in the original BCA, we generally pick m nodes by some strategy (which we will

discuss next), and apply BCA processing to each of them. After the processing, their $\rho(q, *)$ become non-zero, and thus they are included into S_f.

The question now is how to select the m nodes. First, to tighten the bounds quickly, we need to reduce the total residual (as we will see soon in Proposition 3.5). Thus, it is preferred to select a node v with a large residual. Second, it is better to select nodes with few out-neighbors, since the BCA processing time of a node is linear in its number of out-neighbors. Factoring in both criteria, we define the *benefit* of a node v as $\mu(q, v)/|\text{Out}(v)|$. Thus, in each expansion we pick at most m nodes with the largest non-zero benefits. Note that the first expansion will only bring in the query node q, since it is the only node with non-zero residual initially.

In the above expansion model, m controls the granularity of the expansion. That is, a small m may result in too frequent bounds updating, while a large m may miss the opportunity to stop early. We use $m = 100$ based on some trial queries. The performance is not sensitive to small changes in m.

As the second step in Stage I, we initialize the bounds $\check{f}^{(0)}(q, v)$ and $\hat{f}^{(0)}(q, v)$ for each seen node $v \in S_f$, as well as the unseen upper bound $\hat{f}^{(0)}(q)$. The initializations are based on the current $\rho(q, *)$ and $\mu(q, *)$ in BCA, as below.

PROPOSITION 3.5 (BOUNDS OF F-RANK). Given the current $\mu(q, *)$ and $\rho(q, *)$ values as maintained by BCA, the following bounds hold:

$$\hat{f}^{(0)}(q) = \frac{\alpha}{2-\alpha} \max_{u \in V} \mu(q, u) + \frac{1-\alpha}{2-\alpha} \sum_{u \in V} \mu(q, u) \tag{3.22}$$

$$\check{f}^{(0)}(q, v) = \rho(q, v), \forall v \in S_f \tag{3.23}$$

$$\hat{f}^{(0)}(q, v) = \rho(q, v) + \hat{f}^{(0)}(q), \forall v \in S_f \tag{3.24}$$

Note that we consider the residual that may repeatedly spread to and from a node, and thus obtain better upper bounds than the work by Gupta *et al.* [51]. The latter only accounts for residual that may spread to a node for the first time.

Stage II. For each seen node, the iterative refinement of its bounds (Eq. 3.17 and 3.18) is applied as is.

Realization of B-Rank. Our realization of Stage I and II hinges on the concept of *border nodes* [90, 89]. A border node of the b-neighborhood S_b has at least one of its in-neighbors outside S_b. Thus, to reach q from any unseen node $v \notin S_b$, we must first pass through a border node. Let u be a border node with the largest upper bound. Then, the unseen upper bound, which is the largest probability of reaching q from $v \notin S_b$, can be achieved by reaching u with probability 1 in one step, and then continuing from u to q:

$$\hat{b}(q) = (1-\alpha) \max_{u \in \partial(S_b)} \hat{b}(q,u), \tag{3.25}$$

where $\partial(S_b)$ is the set of border nodes of S_b. Note that reaching u in one step dampens the probability by a factor $1-\alpha$ due to the geometrically distributed walk length.

Stage I. In the first expansion, let $S_b = \{q\}$ with $\check{b}^{(0)}(q,q) = \alpha$ due to Eq. 3.9, and $\hat{b}^{(0)}(q,q) = 1$. The unseen upper bound is $\hat{b}^{(0)}(q) = 1-\alpha$ due to Eq. 3.25.

In subsequent expansions, we aim to reduce the unseen upper bound, which will further improve the bounds of the seen nodes owing to the propagating effect of the iterative refinement in Stage II. We pick up to m border nodes with the largest upper bounds, and bring all of their in-neighbors into S_b. This makes these m nodes no longer border nodes, and thus reduces the unseen upper bound (which is based on the border node with the largest upper bound, see Eq. 3.25). As we discussed for F-Rank, m controls the granularity of expansion, and it can be set empirically ($m = 5$ in our experiments).

To initialize the bounds for nodes already in S_b before this expansion, we use their bounds from the last expansion. For each newly included node, we use a lower bound of zero, and the unseen upper bound from the last expansion. Finally, we initialize the current unseen upper bound using Eq. 3.25.

Stage II. We iteratively refine the bounds for each seen node as in Eq. 3.17–3.18. In

addition, based on Eq. 3.25 we can also refine the unseen upper bound in each iteration:

$$\hat{b}^{(i)}(q) = (1-\alpha) \max_{u \in \partial(S_b)} \hat{b}^{(i)}(q,u). \tag{3.26}$$

3.5.2 Distributed Solution for 2SBound

We have so far assumed that the entire graph resides in the main memory of a single machine. To scale 2SBound to very large graphs, we design a solution based on the observation of the *active set* and the technique of *data striping*.

3.5.2.1 Active Set

For any query, 2SBound only needs to maintain a subset of the nodes and edges in G rather than the entire G. As our realization is decoupled into F-Rank and B-Rank, we maintain the nodes in their respective neighborhood, and the set of edges for these nodes. We call the nodes and edges to be maintained the *active nodes* and *active edges* respectively, which collectively form the *active set*.

Memory requirement. The active set is the minimum working set that must reside in the main memory. Otherwise, frequent page swappings occur during iterative refinement. Fortunately, in terms of the absolute space cost, the active set is usually a minute subset of the entire graph. In terms of the scalability, as the graph grows, the active set scales at a slower rate, as our analysis below and the experiments show. Thus, it is practical to fit the active set in the main memory.

Orders of growth. Assume a graph G with $|V|$ nodes and average degree $\bar{D}$. As observed by Leskovec *et al.* [72], the average degree can be modeled by power laws: $\bar{D} \approx c|V|^{a-1}$, where c and a are graph-dependent constants and $1 < a < 2$ on most real-world graphs. Hence, G incurs $O(|V| + |V|\bar{D}) = O(c|V|^a)$ space, where $|V|\bar{D}$ is the number of edges.

Next, we examine the active set. In each neighborhood expansion, we pick m nodes and

bring their neighbors into the active set. After n expansions we have $O(nm\bar{D})$ active nodes. Since m is a constant and we assume n is also a constant for a given slack ϵ and graph G, the number of active nodes is $O(\bar{D})$. Hence, the active set incurs $O(\bar{D} + \bar{D}^2) = O(c^2|V|^{2(a-1)})$ space. To compare the orders of growth of the active set and the graph, we obtain the following result, which implies that the active set grows slower than the graph.

$$\forall a \in (1,2), \quad \lim_{|V|\to\infty} \frac{c^2|V|^{2(a-1)}}{c|V|^a} = \lim_{|V|\to\infty} c|V|^{a-2} = 0. \tag{3.27}$$

3.5.2.2 Distributed Architecture

Our architecture includes one *active processor*, which is connected to multiple *graph processors* over a network.

Active processor (AP). Starting with the query node, AP expands the neighborhoods by picking some nodes as in Section 3.5.1, whose neighbors are subsequently brought into the active set. But instead of pulling them directly from the graph (which is not in its main memory), AP queries the graph processors over a network, which are responsible for identifying and sending back the new active nodes and edges. Subsequently, AP incrementally assembles the active set from the responses, updates the bounds, and proceeds to the next expansion until the top K nodes can be determined.

Graph processors (GP). When the graph does not fit into the main memory of a single machine, we apply *data striping* [87] to segment data over multiple storage units. In our case, the graph is segmented across multiple GPs, where each GP stores a subset of the nodes and edges in its main memory. In particular, we assign nodes (and their edges) in the graph to GPs in a round-robin fashion. During query processing, upon an expansion request from AP, each GP identifies the requested active nodes and edges stored in it, and sends them to AP. AP then incrementally assemble the active set, as described earlier.

Such striping presents several benefits. First, it aggregates the fast storage (main memory) of GPs to handle large graphs. Second, it enables parallel access to different parts of

the graph. Third, apart from segmenting the graph which involves minimal work, there is no offline precomputation like the original 2SBound. In general, using a cluster of commodity computers is more flexible, cost effective, and reliable (if with redundancy) than using a single powerful machine.

3.6 Experiments

We aim to evaluate the effectiveness of RoundTripRank and RoundTripRank+, as well as the efficiency of the top-K algorithm 2SBound.

3.6.1 Experimental Datasets

We use two real-world graphs for our experiments, as follows. In particular, the first one is a directed graph, whereas the second is undirected.

Bibliographic network (BibNet). There are 2 million nodes (papers, authors, terms, venues) and 25 million edges (paper-paper, paper-term, paper-venue, paper-author), extracted from DBLP [31] and CiteSeer [49]. The paper-paper citation edges are directed, while the others are undirected. We set the edge weights largely following a previous work [90]. For each paper, we assign a total weight of 1, 10, 10, 1 to its edges connecting its terms, cited papers, authors, and venue, respectively, split equally among each type of edge.

Search log graph (SLog). We use a search engine log [2]. After removing search phrases and clicked URLs that only appear once, we construct a graph with 2 million nodes and 4 million edges, where the search phrases and clicked URLs are the nodes, and an undirected edge is drawn between them if the search phrase has a click landing on the URL. The count of such clicks is used as the edge weight. Finally, we add self-loops of weight 0.1 to all nodes to make the graph non-bipartite, as some baselines like SimRank do not work for nodes on different sides of a bipartite graph.

3.6.2 Effectiveness of RoundTripRank and RoundTripRank+

We evaluate the ranking of the results to validate our hypotheses. First, most ranking scenarios require some trade-off between importance and specificity, as RoundTripRank pursues. Second, the optimal trade-off varies from task to task, depending on the user preference or task objective. Hence, an effective proximity measure should cater to different tasks in a flexible manner, as RoundTripRank+ pursues.

Hence, after describing the experimental settings, we first compare RoundTripRank with a few mono-sensed baselines to demonstrate the need for the dual senses. Next, we show that task-oriented customization is indeed beneficial, and compare RoundTripRank+ with various dual-sensed baselines.

Subgraphs. As we evaluate the effectiveness rather than efficiency here, we use the iterative method in Eq. 3.6 and 3.9 to obtain the exact ranking, in order to eliminate the effect of approximation. However, some baselines (*e.g.*, SimRank and TCommute) are very expensive to compute exactly on the full graphs. Thus, as some previous studies [100, 99], we use smaller subgraphs for the effectiveness evaluation. The full graphs will be used in Section 3.6.3 for the efficiency study.

For BibNet, we focus on 28 hand-picked major venues in four related areas (database, data mining, information retrieval and artificial intelligence). Moreover, we only consider the 500 most prolific authors in these venues, and terms appearing at least twice. Subsequently, we obtain a subgraph of $20,545$ nodes and $252,272$ edges.

For SLog, we start with 200 random nodes, and expand to their neighbors for three hops, To prevent the subgraph from growing too large quickly, we restrict the expansion at each node to at most 20 neighbors. We obtain a subgraph of $23,665$ nodes and $74,504$ edges.

Experiment methodology. In our experiments, we reserve some nodes with known association to the query, and then test whether a proximity measure can rank these nodes highly without the knowledge of the association. In other words, such reserved nodes form

the ground truth, which we aim to re-discover. This methodology makes the ground truth easily available for large-scale evaluation, which has been used as a benchmark test [90] for graph proximity. If a measure performs well on this ground truth, presumably it can also rank other matching results highly.

Thus, we use the ranking scenarios in Task 1–4 below, which are adapted from Task A–D in Section 3.1, such that the ground truth nodes for each query are automatically known. To test the ability to recover the ground truth, we remove all direct edges between the query and ground truth nodes. As these tasks mimic the different trade-offs between importance and specificity (which we will analyze in Section 3.6.2.2), the need for a customizable trade-off can be justified.

- *Task 1 (Author)*: On BibNet, given a paper as the query node, find all of its author(s).
- *Task 2 (Venue)*: On BibNet, given a paper as the query node, find its venue.
- *Task 3 (URL)*: On SLog, given a search phrase as the query node, find a randomly chosen clicked URL.
- *Task 4 (Search)*: On SLog, given a search phrase as the query node, find its equivalent phrases. We deem two phrases equivalent if they contain the exact same non-stop words (*e.g.*, "the apple ipod" and "ipod of apple").

Evaluation. For each task, we randomly sample 1000 nodes as the *test queries*. To assess the ranking for a query, we filter out the query node itself and nodes not of the target type. We then evaluate the filtered ranking against the ground truth using NDCG@K with ungraded judgments. Statistical significance is verified using two-tail paired t-tests.

3.6.2.1 RoundTripRank and Mono-Sensed Baselines

As the first dimension, we validate that dual-sensed RoundTripRank outperforms mono-sensed baselines.

	Task 1			Task 2			Task 3		
K	5	10	20	5	10	20	5	10	20
RoundTripRank	**0.380**	**0.419**	**0.453**	**0.757**	**0.785**	**0.797**	**0.375**	**0.411**	**0.436**
F-Rank/PPR	0.328	0.365	0.399	0.750	0.784	0.794	0.369	0.408	0.435
B-Rank	0.304	0.343	0.382	0.686	0.731	0.744	0.196	0.231	0.270
SimRank	0.198	0.224	0.251	0.575	0.620	0.631	0.099	0.121	0.145
AdamicAdar	0.169	0.177	0.183	0.213	0.243	0.303	0.000	0.000	0.001

(Continue'd)

	Task 4			Average		
K	5	10	20	5	10	20
RoundTripRank	**0.488**	**0.538**	**0.576**	**0.500**	**0.538**	**0.566**
F-Rank/PPR	0.378	0.431	0.476	0.456	0.497	0.526
B-Rank	0.452	0.509	0.553	0.410	0.453	0.487
SimRank	0.437	0.495	0.540	0.327	0.365	0.392
AdamicAdar	0.419	0.471	0.519	0.200	0.223	0.251

Figure 3.6: NDCG@K of RoundTripRank and mono-sensed baselines. The best in each column is bolded, and the runner-up is underlined.

Quantitative results. We set $\alpha = 0.25$ for RoundTripRank, *i.e.*, $L, L' \sim \text{Geo}(0.25)$. Its ranking is stable for a wide range of α between 0.1 and 0.5. As baselines, we use importance-based F-Rank/PPR and specificity-based B-Rank with the same α. We also use SimRank [58] (with $C = 0.85$) and AdamicAdar [1], both of which capture some form of "closeness" as Section 3.2 explained.

As reported in Figure 3.6, RoundTripRank consistently outperforms all the baselines across all four tasks. On average, it improves NDCG@5 over the runner-up (F-Rank/PPR) by 10%, with statistical significance ($p < 0.01$). Hence, some balance between importance and specificity is indeed necessary.

Illustrative results. We list the top 5 results of two example queries on the full BibNet graph. Given some term nodes as the query, we rank the matching venues. It is relatively objective to judge the importance and specificity of venues as compared to other nodes.

We show the results of the first query, "spatio temporal data" (three term nodes), in Figure 3.7. F-Rank finds important venues in the database area. However, most of them are too general, and will be ranked highly for most data-centric topics. In contrast, B-

(a) F-Rank/PPR	(b) B-Rank	(c) RoundTripRank
Comp. Res. Repository	Spatio-Temporal DBs	Spatio-Temporal DBs
SIGMOD	Temporal DBs, Dagstuhl	Temp. Repr. & Reasoning
VLDB	Spatio-Temporal Data Mining	GeoInformatica
ICDE	Temporal Aspects in Info. Sys.	ACM GIS
Temp. Repr. & Reasoning	Ana. & Retr. in Video Streams	VLDB

Figure 3.7: Ranking venues for "spatio temporal data."

(a) F-Rank/PPR	(b) B-Rank	(c) RoundTripRank
World Conf. on WWW	Wkshp. on Semantic Web	Intl. Semantic Web Conf.
IEEE Intelligent Sys.	Trends in Web Sci.	Intl. Conf. on Web Services
Intl. Conf. on Web Services	Semantic Grid	IEEE Intelligent Sys.
Commun. ACM	J. Web Engineering	J. Web Semantics
Comp. Res. Repository	Semantic Tech.	Euro. Semantic Web Conf.

Figure 3.8: Ranking venues for "semantic web."

Rank favors venues specifically tailored to the topic, but they are less well known. Lastly, RoundTripRank's ranking is comprehensive and balanced, with not only important (*e.g.*, VLDB) or specific venues (*e.g.*, Spatio-Temporal DBs), but also venues that are themselves a balance of the two senses (*e.g.*, ACM GIS). Similar observations can be made in Figure 3.8 for the second query "semantic web."

3.6.2.2 RoundTripRank+ and Dual-Sensed Baselines

As the second dimension, we investigate the benefit of a customizable trade-off between importance and specificity. In particular, we study the effect of varying β on RoundTripRank+, and compare it to various dual-sensed baselines.

Effect of specificity bias β. Recall that $\beta \in [0, 1]$ is used to customize the trade-off between importance and specificity in RoundTripRank+. Figure 3.9 shows the performance when varying β between 0 and 1.

We first observe that extreme β values (close to 0 or 1) result in poor performance. When $\beta = 0$ (or 1), RoundTripRank+ only captures importance (or specificity). This reconfirms our findings in Section 3.6.2.1 that we need both senses.

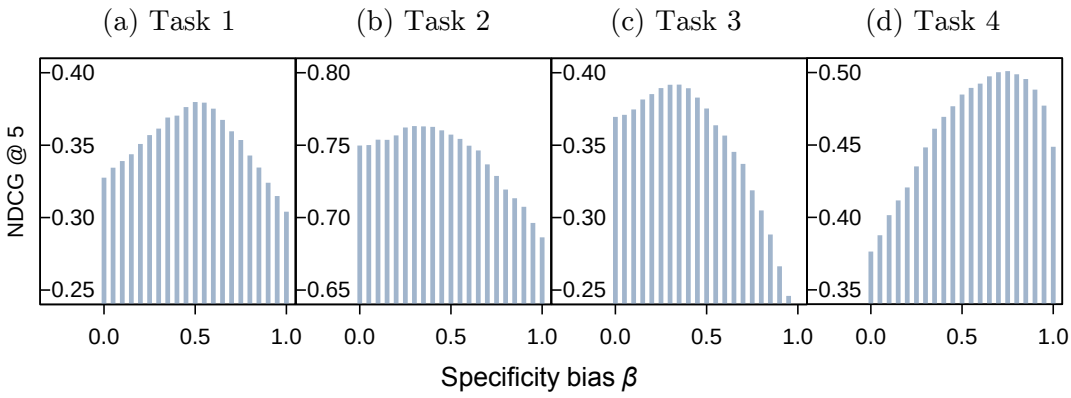


Figure 3.9: Effect of the specificity bias.

Second, different tasks have varying optimal values β^*, naturally reflecting the different trade-offs required in the tasks. In Task 1 (Author), $\beta^* \approx 0.5$, as paper authors can be either important (*e.g.*, the faculty) or specific (*e.g.*, the student). In Task 2 (Venue), $\beta^* < 0.5$, as paper submissions tend to favor important venues. In Task 3 (URL), $\beta^* < 0.5$, as users are often biased to click on important and well-known sites. In Task 4 (Search), $\beta^* > 0.5$, as equivalent search phrases ideally refer to the exact same concept, and thus are inherently specific to each other. The varying β^* values imply no "one-size-fits-all" solution, and thus a customizable trade-off is beneficial.

For a given task at hand, the optimal β can be tuned using some development queries, or predetermined by identifying the underlying objective as we analyzed above, or directly specified by users to reflect their needs. If such options are not available, we can use the default $\beta = 0.5$, which still fares better than $\beta \to 0$ or 1 as just observed.

Comparing to existing dual-sensed baselines. We set $\alpha = 0.25$ for RoundTripRank+ as before. For each task, to choose the optimal β, we use 1000 randomly sampled *development queries* that do not overlap with the test queries.

We use existing dual-sensed baselines, namely truncated commute time or TCommute [88, 90] (with $T = 10$), and ObjSqrtInv [56] (with $d = 0.25$). Additionally, as our computational model is actually a geometric mean of F-Rank and B-Rank, we also compare

	Task 1			Task 2			Task 3		
K	5	10	20	5	10	20	5	10	20
RoundTripRank+	**0.380**	**0.419**	**0.453**	**0.763**	**0.795**	**0.805**	**0.386**	**0.419**	**0.445**
TCommute	0.338	0.380	0.415	0.751	0.786	0.797	0.372	0.410	0.435
ObjSqrtInv	0.331	0.369	0.407	0.756	0.785	0.796	0.359	0.398	0.424
Harmonic	0.316	0.354	0.393	0.700	0.743	0.755	0.301	0.338	0.368
Arithmetic	0.336	0.376	0.411	0.750	0.784	0.795	0.369	0.410	0.432

(Continue'd)

	Task 4			Average		
K	5	10	20	5	10	20
RoundTripRank+	**0.504**	**0.556**	**0.594**	**0.508**	**0.547**	**0.574**
TCommute	0.432	0.487	0.530	0.473	0.516	0.544
ObjSqrtInv	0.405	0.459	0.502	0.462	0.503	0.532
Harmonic	0.493	0.544	0.583	0.452	0.495	0.525
Arithmetic	0.421	0.480	0.522	0.469	0.513	0.540

Figure 3.10: NDCG@K of RoundTripRank+ and existing dual-sensed baselines. The best in each column is bolded, and the runner-up is underlined.

with their harmonic [2, 37] and arithmetic means. These approaches [56, 88, 2, 37] neither recognize the benefit of, nor implement, customizable trade-offs for different tasks.

As reported in Figure 3.10, RoundTripRank+ consistently outperforms all the baselines in all four tasks. On average, it improves NDCG@5 by 7% over the runner-up (TCommute) with statistical significance ($p < 0.01$). The results clearly highlight the advantage of flexible trade-offs across tasks.

Comparison to customized dual-sensed baselines. Recall that existing dual-sensed baselines [56, 88, 2, 37] apply a fixed trade-off across tasks. To give them the benefit of a flexible trade-off, we customize each of them with a tunable $\beta \in [0, 1]$, putting weights $1 - \beta$ and β on their two sub-measures, respectively. To choose the optimal β for each task, we use the same development queries of RoundTripRank+. We stress that the customizations are implemented by us, and existing works are unaware of such a need.

Nonetheless, RoundTripRank+ still performs the best consistently, as summarized in Figure 3.11 where each customized baseline is marked with "+". For brevity only NDCG@5 is shown, but the conclusion is similar @10 or @20. On average, RoundTripRank+ improves

	Task 1	Task 2	Task 3	Task 4	Average
RoundTripRank+	**0.380**	**0.763**	**0.386**	**0.504**	**0.508**
TCommute+	<u>0.361</u>	0.751	<u>0.377</u>	0.460	<u>0.488</u>
ObjSqrtInv+	0.323	<u>0.756</u>	0.369	0.448	0.474
Harmonic+	0.326	0.745	0.360	<u>0.496</u>	0.482
Arithmetic+	0.356	0.751	0.373	0.458	0.484

Figure 3.11: NDCG@5 of RoundTripRank+ and customized dual-sensed baselines. The best in each column is bolded, and the runner-up is underlined.

over the runner-up (TCommute) by more than 4%, with statistical significance ($p < 0.01$). In addition, the baselines perform unevenly across tasks—the runner-up varies from task to task, providing no evidence to argue one or another even empirically.

We attribute the better performance of RoundTripRank+ to the coherent integration of importance and specificity in a round trip. In contrast, most baselines combine their sub-measures for the two senses in a somehow ad hoc manner. For example, the arithmetic mean is simply the expectation of two *independent* trials, one for each sense, lacking coherence in their integration. Others such as TCommute, as Section 3.2 discussed, lack an interpretation in terms of the two senses (which has not been thoroughly investigated).

3.6.3 Efficiency of 2SBound

We evaluate the top-K algorithm 2SBound for RoundTripRank on the two full graphs. In particular, we examine its query time and approximation quality on a single machine, and its scalability on our distributed architecture.

We use $K = 10$ throughout. Occasionally some queries are very slow for their bounds to become tight enough. Hence, we stop prematurely after 50 neighborhood expansions for 2SBound. This only affects a few queries, and the performance of the prematurely terminated queries are still accounted for in our evaluation.

3.6.3.1 Query Time and Approximation Quality

We load the entire graph into the main memory of a single computer, which has 8GB capacity. From each graph, we randomly sample 1000 nodes as queries. We only report the results on BibNet, as similar observations can be made on both graphs.

Query time. We compare the query time of 2SBound with that of a naïve baseline and three weaker schemes of 2SBound:

- Naive: The naïve iterative method (Eq. 3.6 and 3.9).
- G+S: A weaker scheme of 2SBound. We apply the work by Gupta *et al.* [51] to update the bounds for F-Rank, and the work by Sarkar *et al.* [89] for B-Rank, which are their respective state-of-the-art algorithms.
- Gupta: Same as G+S, but using our two-stage framework for B-Rank, while still using Gupta's method for F-Rank.
- Sarkar: Same as G+S, but using our two-stage framework for F-Rank, while still using Sarkar's method for B-Rank.

We show their average query time in Figure 3.12(a). As expected, a large slack ϵ greatly reduces the query time (except Naive which does not use ϵ). Notably, 2SBound is two orders of magnitude faster than Naive, and 2–10 times faster than the others. It is also stable across queries. For instance, its 99% confidence interval is 1255 ± 154ms at $\epsilon = 0.01$.

Approximation quality. To evaluate the approximation quality, we compare 2SBound's ranking with the exact ranking using NDCG, precision and Kendall's tau [23].

We present the approximation quality of 2SBound in Figure 3.12(b), along with its query time for reference. While query processing speeds up five-fold as the slack increases, the quality drops slightly. Nevertheless, all metrics are still above 0.9 when the time is only 300ms. Most drops are observed in precision and Kendall's tau, which penalize mistakes

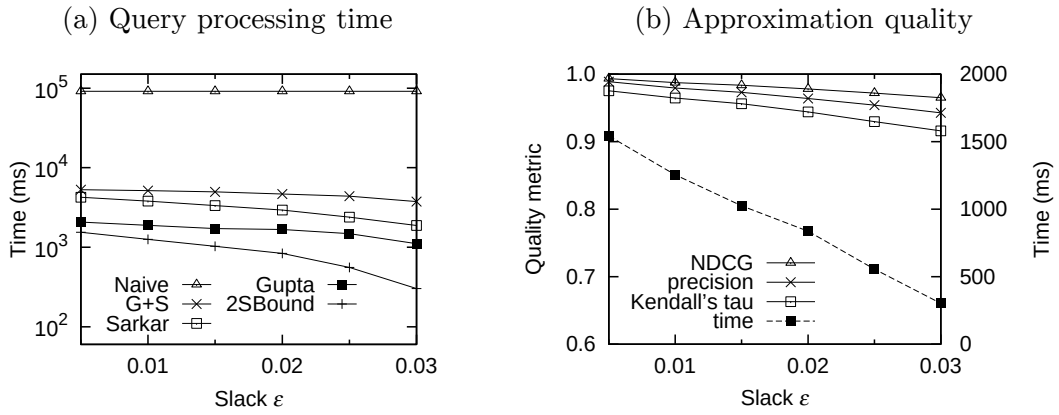


Figure 3.12: Time and quality of 2SBound on BibNet under varying slacks.

equally regardless of their ranked positions. In contrast, NDCG has a smaller drop, meaning that mistakes are rare at highly ranked positions.

3.6.3.2 Scalability

We study the scale-up of the distributed solution for 2SBound. The goal is scaling to larger graphs with minimal increase in the active set and query time.

As both BibNet and SLog grow over time, we model their growth by taking five snapshots at different timestamps—every four years on BibNet from 1994 to 2010, and about every six days on SLog during May 2006. To simulate our AP/GP architecture, for each graph, we assume its i-th snapshot requires i GPs, $\forall i \in \{1, \ldots, 5\}$. Note that all snapshots are cumulative, and thus they are larger at later timestamps. On each snapshot, we randomly sample 1000 nodes as queries, and set a slack $\epsilon = 0.01$.

Efficiency. We report the 99% confidence intervals of the active set size and the query time in Figure 3.13. Recall that the active set is the minimum working set on AP.

We observe that the active set remains small even on the largest snapshot. For instance, on BibNet it is merely 2.4MB or 0.3% of the 2010 snapshot. Further, SLog has larger snapshots but smaller active sets than BibNet. As Section 3.5.2.1 showed, the active set size, $O(\bar{D} + \bar{D}^2)$, is correlated with the average degree $\bar{D}$, which is smaller on SLog.

(a) BibNet snapshots

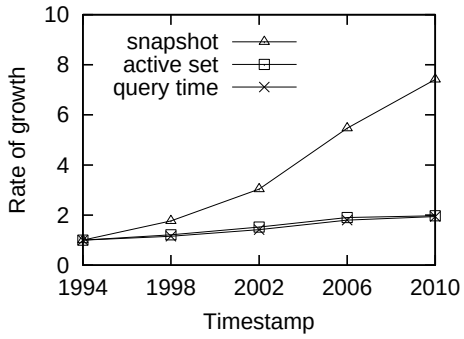
Time-stamp	Snapshot size/MB	Active set size/MB	Query time/ms
1994	93	1.2±.1	728± 48
1998	165	1.5±.1	842± 53
2002	284	1.8±.1	1031± 66
2006	510	2.3±.2	1311±101
2010	692	2.4±.2	1415±102

(b) SLog snapshots

Time-stamp	Snapshot size/MB	Active set size/MB	Query time/ms
5/6	264	.041±.001	51±1
5/12	495	.045±.002	55±2
5/18	713	.049±.002	57±2
5/24	905	.053±.002	61±3
5/31	1114	.054±.002	66±3

Figure 3.13: Active set size and query time on growing graphs.

(a) BibNet snapshots



(b) SLog snapshots

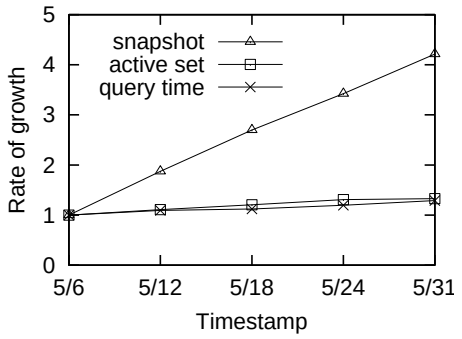


Figure 3.14: Rate of growth in snapshot, active set and query time.

Rate of growth. We compare the rate of growth of three quantities, namely, snapshot size, active set size and query time. For each graph, we normalize each of the three quantities on all snapshots by their respective values on the first snapshot. Thus, for each quantity, we obtain a rate of growth across different snapshots with respect to the first snapshot, as shown in Figure 3.14.

As Section 3.5.2.1 discussed, the active set grows much slower than the snapshot. In particular, while the snapshot grows by a factor of 7.4 on BibNet and 4.2 on SLog over the entire period, the active set only grows by a factor of 1.9 and 1.3, respectively. Furthermore, query time depends only on the active set size, and thus has a similar rate of growth. Based on these trends, we can scale query processing to even larger graphs with both space and computational efficiency.

3.7 Conclusion

In this chapter, we diversified ranking with the senses of importance and specificity, as well as their customizable trade-offs. In particular, we developed RoundTripRank for networked entities, which coherently integrates importance and specificity in a round trip random walk. We further generalized it to RoundTripRank+ using a scheme of hybrid random surfers for a flexible trade-off between the two senses. Empirically, we showed them to be effective on two real-world graphs across various ranking tasks. Finally, we proposed a top-K algorithm 2SBound, enabling online query processing with a close approximation.

Chapter 4

Classification with Heterogeneous Contexts

In this chapter, we propose to enrich the task of classification with *heterogeneous contexts*, in terms of two dimensions: different *relationship types* and varying *confidence levels*. In particular, we focus on graph-based regularization, where the classification of an object is influenced or regularized by its neighbors on the graph. However, the contexts of an object on a graph are often heterogeneous. First, an object can be related to others in many different ways, and our confidence of the classifications varies across different objects. These heterogeneous contexts should be exploited to guide the regularization on the graph. Thus, we propose a Dirichlet-based graph regularization framework, which can effectively incorporate both dimensions of heterogeneity. Interestingly, our framework can be interpreted as a special form of backward random walk.

4.1 Introduction

Many applications in information retrieval (IR) call for effective classification techniques. For instance, the problem of text categorization is fundamental to a myriad of real-world tasks, such as automated email organization, spam detection, and information filtering. As another example, fine-grained query intent classification enables a search engine to direct users to the intended search verticals, greatly enhancing user experience.

While a conventional machine learning algorithm can be used for classification in IR applications, straightforward adaptation is often unfruitful, since there exist challenges beyond traditional classification tasks. One prominent issue is the sparsity of feature vectors. For

instance, in our query intent classification dataset, about 95% of the queries contain no more than five words.

Recent developments in graph-based regularization [113, 107, 13, 14] somewhat tackles the problem of feature sparsity. Instead of object-level features that are independently extracted from each object, we now consider relationships that are extracted from each *pair* of objects. Hence, each object can potentially pair with any other object to form some type of relationship. In this chapter, we focus on relationships that capture different forms of similarity between two objects, such that a graph can be constructed by linking similar objects with edges. The edges can be used to propagate classification evidence from one object to another based on the key assumption of *consistency*: similar objects are likely to have similar class labels [113]. In other words, the classification of an object can be influenced or regularized by its neighbors on the graph.

Given the above benefits, graph-based regularization has also been adopted in IR applications, most notably through the use of a click graph [73, 29, 61] in query intent classification. A click graph usually consists of queries and websites as its nodes. A query q and a website w are connected by an edge if q results in a click landing on w. Intuitively, two queries connected to the same website are related, and thus should share a similar prediction.

While making the same assumption of consistency, we are motivated by two core observations in the following, which entail key intuitions towards a more effective graph-based regularization framework.

First, most earlier regularization models only deal with a single relationship type. However, heterogeneous types often exist. For instance, in query intent classification, queries can be related through not only clicks, but also other relationships, including lexical similarity ("acer laptop" and "laptop" are related due to word overlap), co-session (the second query could be a refinement of the first in the same session), and the similarity of search result pages (two queries that retrieve similar pages may be similar themselves). These different relationship types potentially complement each other, as some may only cover a

small fraction of all queries (*e.g.*, not all shopping queries have associated product clicks). In addition to sparsity, some relationship types may lack the robustness to capture similarities between two objects. To illustrate, while both queries "hp 3 in 1" and "canon printer" refer to printers, they contain no common words and thus their similarity cannot be represented by lexical relationships. To support heterogeneous relationship types, the challenge lies in how different similarity functions can be aggregated to optimize the regularization, which is missing from previous regularization frameworks.

Second, in existing graph regularization, the extent to which the classification of an object o influences o' only depends on their similarity. Greater similarity between o and o' means that they have stronger influences on each other. While this is reasonable, a second factor, largely overlooked by previous approaches, is the confidence of classification. If we are more confident about the prediction on o, we expect it to influence its neighbors on the graph more. On the other hand, if we are unsure about the prediction on o, we should minimize its influence. In other words, we use objects that are easier to classify (with higher classification confidence) to help predict harder ones (with lower classification confidence), but less so the other way round. Existing regularization studies do not provide for a mechanism to incorporate the heterogeneous confidence levels. In contrast, our modeling of objects with Dirichlet priors allows us to interpret the observation counts as confidence.

Finally, another challenge of classification lies in insufficient labeled data, especially in real-world IR applications with a large number of classes. Our query intent classification dataset involves 2043 classes, and thus require much more labeled training queries than, say, a binary classification task. Our framework deals with the shortage of labeled training data in two ways. To begin with, like other graph-based regularization frameworks [113, 107], classification evidence is propagated across the graph along the edges. By using more unlabeled data, we generally obtain a denser graph that promotes such propagation, and hence improve the performance. Our experiments show that adding more unlabeled data yields superior results, despite using the same set of labeled training data. Next, our framework

allows for fewer parameters, independent of the number of classes. Hence, it is effective even with limited training data and a large number of target classes.

In this chapter, we propose a regularization framework based on our insights above. To summarize, we make the following contributions.

- **Insight:** We recognize the need for different relationship types that complement each other, and confidence-aware regularization that distinguishes objects of varying classification confidence.
- **Framework**: We develop a Dirichlet-based graph regularization framework DirGraph. The framework supports arbitrary heterogeneous relationships, is confidence-aware, and works with a limited amount of training data.
- **Applications**: We showcase how DirGraph can be applied to a few applications in IR, primarily focusing on query intent classification in the shopping domain.
- **Experiments**: We conduct extensive experiments on a real-world dataset for query intent classification. The results show that our approach substantially outperforms traditional regularization methods that do not consider heterogeneous relationship types or confidence levels, validating our observations.

4.2 Related Work

Pairwise relationships. Many IR classification tasks are plagued by feature sparsity. Thus, a significant amount of research has studied augmenting the feature vector (*e.g.*, [20, 94, 93, 21] for query-intent classification). Unfortunately, these techniques are generally not universal to other classification tasks. Driven by the hypothesis that "similar objects share similar labels" [113], we consider pairwise relationships that allow for "feature extraction" from any pair of objects, expanding the feature space in some way.

Although it is difficult to incorporate relationships in conventional classification algorithms without blowing up the parameter space, they have been used in recent graph-based regularization frameworks [113, 107, 13, 14] and related random walk approaches [101, 29]. Examples of their applications in IR include the use of click graphs [73, 54, 29, 61] in query intent classification, and document affinity matrices [32, 33] in text retrieval. However, unlike this chapter, most of these methods do not consider heterogeneous types of relationship, which complement each other and thus play an important role in improving classification accuracy. While a recent study [61] explores a content-based similarity in addition to co-clicks, it is proposed for a different problem on the co-classification of webpages and queries. In addition, its treatment is limited as it lacks a mechanism to effectively aggregate arbitrary relationships. In particular, their parameters are manually selected, making it difficult to extend to more types of relationship. In comparison, we learn the parameters via an iterative optimization process.

Confidence levels. Existing regularization frameworks [113, 107, 13, 14] and their task specific realizations (*e.g.*, [73, 54, 29, 61] for query intent classification and [32, 33] for text retrieval) do not recognize the importance and hence take advantage of the classification confidence associated with objects. We observe the need of confidence-aware regularization, and exploit it to improve classification accuracy. More specifically, we are more confident about objects that are easier to classify, which should have a larger influence towards the prediction of their neighbors on the graph.

Limited training data. To deal with the shortage of labeled training data, many task specific techniques exist (*e.g.*, [12, 73, 74, 93] for query-intent classification). However, they are often highly tailored and cannot be easily extended to a generic framework. In our framework, we benefit from using more unlabeled data by exploiting the relationships between objects, leveraging the semi-supervised nature of graph-based regularization [113, 107, 13, 14]. In addition, our framework involves a small parameter space independent of

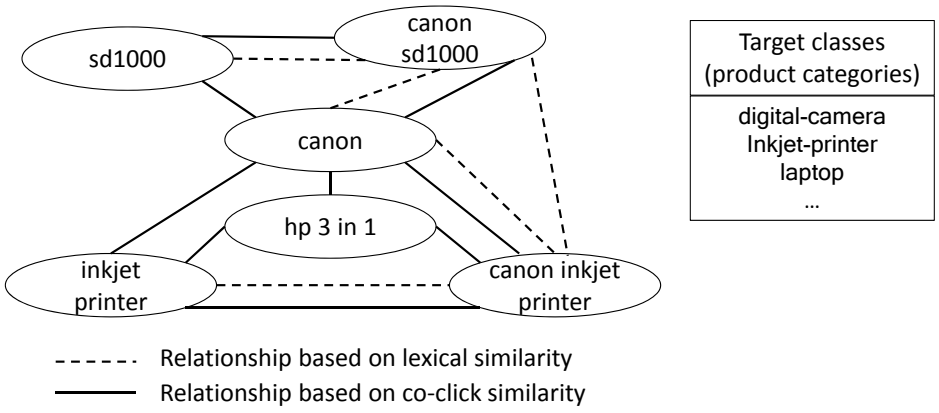


Figure 4.1: Toy graph for classification of shopping queries.

the number of target classes, and thus performs well even with limited training data and thousands of target classes.

4.3 Regularization Framework

In this section, we develop our graph regularization framework called DirGraph. Given a set of objects O and a set of classes C, each object $o \in O$ has some distribution over C. We regularize the class distribution of o by the class distributions of objects similar to o, *i.e.*, adjust the prediction on o according to objects similar to it. As the first step, we capture objects and their pairwise relationships using a graph. Next, based on the graph, we introduce our Dirichlet-based regularization model.

4.3.1 Object-Relationship Graph

We model objects and their relationships using a graph $G = (O, R)$. O is the set of objects that define the *nodes*, and R is the set of relationships between objects that describe the *edges* of the graph. We use the toy graph in Figure 4.1 for the task of query intent classification in the shopping domain as our running example.

Nodes. Each object is a node on the graph. For example, the query "canon" in Figure 4.1

is a node to be classified into one of the target classes, such as `digital camera` or `inkjet printer`, among others.

Edges. An edge on the graph is represented as a triplet $(o, o', \tau) \in R$, which models the relationship of type τ between two objects o and o'. Unlike traditional object-level features that are independently extracted from each object (*e.g.*, words in each document in text classification), τ is in fact a feature defined on a *pair* of objects that describes their mutual similarity. Conceptually, τ can be understood as a particular similarity function on two objects. For instance, as Figure 4.1 illustrates, it can be the cosine similarity between the word vectors of two queries (lexical similarity), or the number of co-clicks for two queries (co-click similarity). Thus, there can exist as many edges between two objects as the number of relationship types, such as between "canon" and "canon sd1000" in Figure 4.1.

The strength of a relationship is encoded as the positive weight of its corresponding edge (o, o', τ), and is thus a function of the two objects o, o' and the relationship type τ, denoted as $W(o, o', \tau)$. $W(o, o', \tau)$ can be understood as the similarity between o and o' with respect to τ. By convention, $W(o, o', \tau) = 0$ iff the edge $(o, o', \tau) \notin R$. In this chapter, we consider bi-directional, symmetric relationships, which have equal strengths in both directions, *i.e.*, $W(o, o', \tau) = W(o', o, \tau)$.

4.3.2 Regularization Model

Given a set of target classes $C = \{1, \ldots, |C|\}$, each object $o \in O$ has an underlying probability distribution over the $|C|$ classes that captures the possible user intents. For instance, in Figure 4.1 the distribution for the query "canon" could be (`digital-camera`:0.3, `inkjet-printer`:0.2, ...), indicating that the user may be looking for a digital camera with probability 0.3, and an inkjet printer with probability 0.2.

As such a distribution is inherently latent, we undertake a "distribution over distributions" instead. Specifically, we model the class distribution of each object o using a Dirichlet

prior $Dir(\boldsymbol{\alpha}_o)$, parameterized by a vector $\boldsymbol{\alpha}_o = (\boldsymbol{\alpha}_o[1], \ldots, \boldsymbol{\alpha}_o[|C|])$. Informally, it describes the distribution over all possible latent class distributions of o, given that each class $i \in \{1, \ldots, |C|\}$ has been observed $\boldsymbol{\alpha}_o[i] - 1$, potentially fractional, times. The Dirichlet prior also conveniently allows us to treat the total count of observations σ_o as the classification confidence—we are more confident in the prior if σ_o is larger. Thus, we call σ_o the prior *confidence* of o, as defined below.

$$\sigma_o \triangleq \sum_{i=1}^{|C|} (\boldsymbol{\alpha}_o[i] - 1) = \sum_{i=1}^{|C|} \boldsymbol{\alpha}_o[i] - |C|. \tag{4.1}$$

In particular, we only consider unimodal Dirichlet priors, *i.e.*, $\forall i \in \{1, \ldots, |C|\}, \boldsymbol{\alpha}_o[i] > 1$, such that $\sigma_o > 0$. As we shall describe later, such a prior may be derived from an initial object-level classifier or previous iterations of the regularization algorithm.

4.3.2.1 Regularization by Neighbors

Given a prior over possible class distributions for an object o, we can treat its neighbors as additional evidence to support its classification. In particular, we consider the neighbors as observation data, weighted by their similarity, from which we can obtain a posterior distribution over possible class distributions for o. In other words, this is a form of regularization in that the original belief on o is regularized by the belief on its neighbors.

Let $N(o)$ denote the set of neighbors of o on the graph. Specifically, for each neighbor $o' \in N(o)$, we can interpret its Dirichlet prior $Dir(\boldsymbol{\alpha}_{o'})$ as observing $\boldsymbol{\alpha}_{o'}[i] - 1$ instances of class i for each of the $|C|$ classes when drawing $\sigma_{o'}$ instances from the underlying multinomial distribution for o'. Under the assumption that similar objects share similar labels and hence observations, we deem that each neighbor o' contributes to the classification of o an observation of $S(o, o')(\boldsymbol{\alpha}_{o'}[i] - 1)$ counts for each of the $|C|$ classes, weighted by the similarity $S(o, o')$ between o and o'. As o can be connected to o' via multiple relationships, we define the similarity between the two objects as a linear combination of their similarity $W(o, o', \tau)$

across all types of relationships:

$$S(o, o') \triangleq \sum_{\tau} \lambda_\tau W(o, o', \tau), \tag{4.2}$$

where each λ_τ is a parameter specifying the overall influence of τ towards $S(o, o')$. We will learn these parameters in Section 4.3.3 to optimize the regularization.

Since the Dirichlet distribution is *conjugate* to the multinomial distribution, factoring in additional multinomial observations still results in a Dirichlet posterior. Specifically, given a Dirichlet prior $Dir(\boldsymbol{\alpha})$, the posterior distribution after observing $\boldsymbol{\beta}[i]$ counts in each class i is simply $Dir(\boldsymbol{\alpha} + \boldsymbol{\beta})$, where $\boldsymbol{\beta} = (\boldsymbol{\beta}[1], \ldots, \boldsymbol{\beta}[|C|])$. Hence, for an object o with prior $Dir(\boldsymbol{\alpha}_o)$, the additional weighted observation data from each neighbor o' would give us the Dirichlet posterior $Dir(\widehat{\boldsymbol{\alpha}}_o)$, where

$$\widehat{\boldsymbol{\alpha}}_o = \boldsymbol{\alpha}_o + \sum_{o' \in N(o)} S(o, o')(\boldsymbol{\alpha}_{o'} - \mathbf{1}). \tag{4.3}$$

Similar to the prior confidence (Eq. 4.1), we can compute the posterior confidence for the object o, as follows.

$$\widehat{\sigma}_o = \sum_{i=1}^{|C|} \widehat{\boldsymbol{\alpha}}_o[i] - |C| = \sigma_o + \sum_{o' \in N(o)} S(o, o')\sigma_{o'} \tag{4.4}$$

However, since all observations are potentially noisy in practice, having more neighbors and hence more observation data does not necessarily imply a higher posterior confidence. As a heuristic, we normalize the observation data by the number of contributing objects weighted by their similarity, $S_o = 1 + \sum_{o' \in N(o)} S(o, o')$. This results in a normalized Dirichlet posterior $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ and a normalized posterior confidence $\widetilde{\sigma}_o$ for each object o:

$$\widetilde{\boldsymbol{\alpha}}_o - \mathbf{1} = \frac{\widehat{\boldsymbol{\alpha}}_o - \mathbf{1}}{S_o} = \frac{1}{S_o}\left(\boldsymbol{\alpha}_o - \mathbf{1} + \sum_{o' \in N(o)} S(o, o')(\boldsymbol{\alpha}_{o'} - \mathbf{1})\right) \tag{4.5}$$

$$\widetilde{\sigma}_o = \frac{\widehat{\sigma}_o}{S_o} = \frac{1}{S_o}\left(\sigma_o + \sum_{o' \in N(o)} S(o, o')\sigma_{o'}\right) \tag{4.6}$$

Intuitively, this normalization step causes observations from neighbors of o to "adjust"

our confidence in o (Eq. 4.6), rather than to raise it monotonically in the unnormalized version (Eq. 4.4).

We can understand Eq. 4.5 as a form of regularization. Informally, our original belief on object o, as prescribed by its prior $Dir(\boldsymbol{\alpha}_o)$, is regularized by its neighbors, which change our belief and produce the regularized distribution $Dir(\widetilde{\boldsymbol{\alpha}}_o)$. Note that in the special case that o has no neighbors ($N(o) = \emptyset$), Eq. 4.5 would give us $\widetilde{\boldsymbol{\alpha}}_o = \boldsymbol{\alpha}_o$, which means no regularization would be done for o.

Confidence-aware prediction. Given the regularized Dirichlet distribution $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ for an object o, we can make a prediction on o by first choosing the most likely class distribution, *i.e.*, the mode distribution $\widetilde{\mathbf{m}}_o$ of $Dir(\widetilde{\boldsymbol{\alpha}}_o)$. Subsequently, we assign classes to o according to $\widetilde{\mathbf{m}}_o$, say, by taking the top k classes with the largest probabilities in $\widetilde{\mathbf{m}}_o$, or all classes above a given probability threshold. Since $\widetilde{\boldsymbol{\alpha}}_o[i] > 1, \forall i$, the mode has a closed form solution:

$$\widetilde{\mathbf{m}}_o = (\widetilde{\boldsymbol{\alpha}}_o - \mathbf{1})/\widetilde{\sigma}_o, \tag{4.7}$$

where $\widetilde{\sigma}_o$ is the normalized posterior confidence of o in Eq. 4.6.

Unlike previous regularization frameworks [113, 107, 13, 14], the prediction by $\widetilde{\mathbf{m}}_o$ is *confidence-aware*, as we will now demonstrate. Dividing both sides of Eq. 4.5 by $\widetilde{\sigma}_o$, we obtain that

$$\frac{\widetilde{\boldsymbol{\alpha}}_o - \mathbf{1}}{\widetilde{\sigma}_o} = \frac{\boldsymbol{\alpha}_o - \mathbf{1} + \sum_{o' \in N(o)} S(o, o')(\boldsymbol{\alpha}_{o'} - \mathbf{1})}{\widetilde{\sigma}_o S_o} \tag{4.8}$$

In Eq. 4.8, the left hand side is simply $\widetilde{\mathbf{m}}_o$. On the right hand side, $\boldsymbol{\alpha}_o - \mathbf{1} = \sigma_o \mathbf{m}_o, \forall o \in O$, where $\mathbf{m}_o$ is the mode distribution of $Dir(\boldsymbol{\alpha}_o)$. Moreover, taking Eq. 4.6 into consideration, Eq. 4.8 can be further expressed as follows:

$$\widetilde{\mathbf{m}}_o = \frac{\sigma_o \mathbf{m}_o + \sum_{o' \in N(o)} S(o, o')\sigma_{o'} \mathbf{m}_{o'}}{\sigma_o + \sum_{o' \in N(o)} S(o, o')\sigma_{o'}}. \tag{4.9}$$

Similar to the Gaussian random field method [113], Eq. 4.9 implies that the regularized mode $\widetilde{\mathbf{m}}_o$ is a weighted average of the original (unregularized) modes $\mathbf{m}_o$ and $\mathbf{m}_{o'}, \forall o' \in N(o)$.

However, unlike [113], the weight on each neighbor o' is $S(o, o')\sigma_{o'}$, which factors in both the similarity $S(o, o')$ and the confidence $\sigma_{o'}$. Hence, our model is confidence-aware, *i.e.*, objects with higher confidence $\sigma_{o'}$ will be weighed more in the regularization. As motivated in Section 4.1, such a property is desirable, since an object with low confidence would potentially introduce noise. Thus, we should downplay its influence. On the other hand, [113] and other regularization models [107, 13, 14] only assign weights according to the similarity between the objects, without utilizing the confidence.

Alternative interpretation. Our regularization (Eq. 4.5) is equivalent to the minimization of a cost function, the essence of many graph-based regularization frameworks [113, 107, 13, 14]. Specifically, we minimize the following function over $\widetilde{\boldsymbol{\alpha}}_o, \forall o \in O$:

$$\mathcal{E} = \frac{1}{2} \sum_{o \in O} \left(\|\widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_o\|^2 + \sum_{o' \in N(o)} S(o, o') \|\widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_{o'}\|^2 \right), \tag{4.10}$$

where $\|\cdot\|$ is the Euclidean distance. Intuitively, the regularization should not change the original belief on o too much, *i.e.*, we want $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ to be close to the prior $Dir(\boldsymbol{\alpha}_o)$. Additionally, $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ should be close to the prior $Dir(\boldsymbol{\alpha}_{o'})$ of each neighbor $o' \in N(o)$, from which observations are contributed towards o.

To see the equivalence, we minimize $\mathcal{E}$ by setting its derivative with respect to $\widetilde{\boldsymbol{\alpha}}_o$ to zero. The solution is algebraically equivalent to Eq. 4.5:

$$\frac{\partial \mathcal{E}}{\partial \widetilde{\boldsymbol{\alpha}}_o} = \widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_o + \sum_{o' \in N(o)} S(o, o')(\widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_{o'}) = 0. \tag{4.11}$$

4.3.2.2 Regularization by Indirectly Related Objects

In the above model (Eq. 4.5), an object o is only regularized by its neighbors on the graph. In most scenarios, indirectly related objects ("neighbors of neighbors") also provide additional evidence for o. The evidence from "neighbors of neighbors" can be captured by regularizing $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ again. In general, we propose an iterative regularization algorithm, which can potentially model the evidence from any object with a path to o on the graph.

Iterative regularization. As our regularized distribution $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ is also Dirichlet, we can feed it back as input for regularization again in the exact same way as we have done in Section 4.3.2.1, treating $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ as the new Dirichlet prior for o. The process can be repeated for any number of iterations.

Let $\boldsymbol{\alpha}_o^{(0)} \triangleq \boldsymbol{\alpha}_o$ denote the Dirichlet parameter of the initial prior for an object o. Furthermore, let $\boldsymbol{\alpha}_o^{(t)}$ be the regularized Dirichlet parameter after t iterations, $\forall t > 0$. Similar to Eq. 4.5, it is easy to see that $\forall t \geq 1$,

$$\boldsymbol{\alpha}_o^{(t)} - \mathbf{1} = \frac{1}{S_o}\left(\boldsymbol{\alpha}_o^{(t-1)} - \mathbf{1} + \sum_{o' \in N(o)} S(o, o')\left(\boldsymbol{\alpha}_{o'}^{(t-1)} - \mathbf{1}\right)\right). \tag{4.12}$$

As shown, we only need to know the Dirichlet parameters from the previous iteration to compute $\boldsymbol{\alpha}_o^{(t)}$. To start the iterative process, we bootstrap from $t = 0$ by specifying $\boldsymbol{\alpha}_o^{(0)}, \forall o$, for the initial prior of each object. How they are initialized is an orthogonal issue to our regularization framework. Different strategies may be applied depending on the application. For our query intent classification problem, we will describe an simple yet effective bootstrapping strategy in Section 4.4.

Finally, we make predictions using the mode distributions after t iterations of regularization, $\mathbf{m}_o^{(t)} = (\boldsymbol{\alpha}_o^{(t)} - \mathbf{1})/\sigma_o^{(t)}$, in the same manner as discussed in Section 4.3.2.1.

Alternative interpretation. The iterative regularization process can be interpreted as a form of *backward* random walk on the graph. The mode of the posterior distribution, $\mathbf{m}_o^{(t)}[i]$, captures the probability that a random surfer, who starts from the object o, reaches some object of class i after t random steps on the graph. In each step, the probability of moving from o to its neighbor o' is $S(o, o')\sigma_{o'}^{(t)} / \left(\sigma_o^{(t)} + \sum_{o' \in N(o)} S(o, o')\sigma_{o'}^{(t)}\right)$. We call this a backward random walk, since the surfer is walking from the node of interest (object o) back to some "reference" node (some object of class i), consistent with earlier chapters. However, unlike earlier chapters, this is a *time-inhomogeneous* Markov chain, since the transition probabilities vary with time due to changing confidence across iterations.

Number of iterations. The total number of iterations T is an important parameter to decide, with similar effects as the length parameter in random walks [101, 29]. In particular, when $T = 1$, it reduces to a simple weighted voting by neighbors. But as $T \to \infty$, $\boldsymbol{\alpha}_o^{(T)}$ becomes the same for every object o in each connected component of the graph, similar to what have been discussed elsewhere [101, 29].

Intuitively, if T is too small, only very short-range dependencies can be captured, ignoring potentially useful longer-range ones. On the other hand, if T is too large, all dependencies regardless of its range will be captured, making the result less discriminative. Thus, an optimal T would ensure a desirable trade-off between short and long-range dependencies. While the optimal T can be selected either manually [29] or heuristically [101], we propose to learn it in the next subsection.

4.3.3 Parameters Learning

In addition to the total number of iterations T, each relationship type τ has a parameter λ_τ that specifies the overall influence of τ on the classification of neighboring objects (see Eq. 4.2). Thus, our parameters are T and Λ, where $\Lambda \triangleq \{\lambda_\tau : \forall \tau\}$.

We propose to learn the parameters with a heuristic objective function defined on only T and Λ. Since $|\Lambda|$ is generally small as restricted by the number of possible relationships between objects (in our experiments $|\Lambda| = 2$), only a few labeled objects are needed for learning. In comparison, traditional supervised classification techniques train a model for each target class. In real-world applications where the number of target classes is large ($|C| = 2043$ in our experiments), an enormous amount of training data would be required, as we would expect at least a few labels for each class. On the other hand, with our heuristic objective function, we require far fewer than $|C|$ to achieve good classification accuracy.

Our first attempt defines a global error function to optimize the parameters. However, due to computational challenges, we subsequently propose a local error function for the dynamic selection of parameters in each iteration. As the parameters are chosen to minimize

the error functions, any existing numerical optimization technique can be used. In particular, we apply Powell's method [82] to solve the optimization problem.

4.3.3.1 Global Optimization

Given a labeled training set $O_L \subset O$, and an initialization for the Dirichlet parameters $\Omega = \{\boldsymbol{\alpha}_o^{(0)} : o \in O\}$ at $t = 0$, we can define an error function as the minimization objective:

$$G_{\text{err}}(O_L, \Omega, \Lambda, T) = \frac{1}{|O_L|} \sum_{o \in O_L} \left\| \mathbf{m}_o^{(T|\Omega,\Lambda)} - \mathbf{u}_o^* \right\|^2, \tag{4.13}$$

where $\mathbf{m}_o^{(T|\Omega,\Lambda)}$ denotes the mode after T iterations of regularization (using Eq. 4.12), with the initialization Ω and parameters Λ. $\mathbf{u}_o^*$ is the "gold standard" distribution derived from the labels of o. $\|\cdot\|$ is the Euclidean distance. (Note that alternative measures such as log-likelihood or KL-divergence may also be used.) We select parameters that minimize this global error, *i.e.*, $\{\Lambda^*, T^*\} = \arg\min_{\Lambda,T} G_{\text{err}}(O_L, \Omega, \Lambda, T)$.

When $T = 1$, $\mathbf{m}_o^{(T|\Omega,\Lambda)}$ can be computed for all $o \in O_L$ using only the neighbors of objects in the labeled training set O_L. When $T = 2$, we need to consider "neighbors of neighbors" as well. For larger T's, we potentially need to compute iteratively over the entire graph of labeled and unlabeled objects. This makes the computation of the error function very costly. As many optimization techniques require repeated computation of $G_{\text{err}}(\cdot)$ for different values of Λ and T, such an error function is computationally infeasible.

4.3.3.2 Iterative Optimization

As an efficient alternative to minimizing the global error, we propose an algorithm where we regularize the Dirichlet parameters $\boldsymbol{\alpha}_o^{(t)}$ (Eq. 4.12) and update the parameters Λ in alternating fashion. In each iteration, we regularize the classification using the Λ learned from the previous iteration. As the Dirichlet distribution on every object changes after regularization, we update Λ by minimizing the error function for the next iteration. This is similar in spirit to the Expectation-Maximization (EM) algorithm. However, unlike EM

which maximizes the likelihood function of the parameters, we are minimizing an error function. Our approach involves the following two steps in each iteration $t, \forall t \geq 0$.

(1) **Regularization step.** If $t = 0$, initialize $\boldsymbol{\alpha}_o^{(0)}, \forall o \in O$, according to the given $\Omega^{(0)} = \{\boldsymbol{\alpha}_o^{(0)} : \forall o \in O\}$. If $t > 0$, compute $\Omega^{(t)} = \{\boldsymbol{\alpha}_o^{(t)} : \forall o \in O\}$ from $\Omega^{(t-1)}$ (Eq. 4.12), using parameters $\Lambda^{(t)}$. Note that $\Omega^{(t-1)}$ and $\Lambda^{(t)}$ have already been computed from the previous iteration.

(2) **Minimization step.** To update Λ for the next iteration, we minimize a local error function over Λ:

$$L_{\text{err}}^{(t+1)}(O_L, \Lambda) \triangleq G_{\text{err}}(O_L, \Omega^{(t)}, \Lambda, 1), \tag{4.14}$$

which is the global error for $T = 1$ with initialization $\Omega^{(t)}$. For the next iteration, we find the parameters $\Lambda^{(t+1)} = \arg\min_\Lambda L_{\text{err}}^{(t+1)}(O_L, \Lambda)$. We continue the iteration until the minimum error converges.

Although in the regularization step, we still need to update the entire graph, it is acceptable as it is done only once per iteration. In contrast, to solve the optimization problem in each minimization step, the error function has to be computed numerous times. Thus it is important the error function involves only light-weight computation. Our local error function (Eq. 4.14) is defined using the global error function (Eq. 4.13) with $T = 1$, which can be easily computed using only the neighbors of the objects in the training set O_L. Typically, $|O_L| \ll |O|$. In most of our experiments, $|O_L|$ is less than 0.1% of $|O|$.

Another distinction in our iterative optimization is that we do not explicitly learn T, the maximum number of iterations. Instead, we terminate the iterations when the minimum local error $\min_\Lambda L_{\text{err}}^{(t)}(\cdot)$ converges. Although such convergence is guaranteed as established in the following, similar to EM, there is no guarantee that the global minimum $\min_{\Lambda,T} G_{\text{err}}(\cdot)$ can be achieved.

PROPOSITION 4.1 (CONVERGENCE OF LOCAL ERRORS). As defined in Eq. 4.14, the sequence of local minimum errors $\left(\min_{\Lambda} L_{\text{err}}^{(t)}(O_L, \Lambda)\right)_{t=1}^{\infty}$ converges to a finite limit.

4.4 Applications in Information Retrieval

We now discuss the applications of the regularization framework on real-world classification tasks in information retrieval. We first formalize what components of the regularization framework must be realized in any application. Next, we showcase a specific application on query intent classification in the shopping domain, followed by brief discussions of applications on other tasks. Finally, we describe the target scenarios of our applications.

4.4.1 Realization of the Framework

Realizing our regularization framework requires a *node model* and one or more *edge models*.

Node model. As we see in Eq. 4.12, to enable iterative regularization, we need an initial model for $t = 0$, characterized by the Dirichlet parameters $\boldsymbol{\alpha}_o^{(0)}, \forall o \in O$. Since each object o is a node in the graph, we call this initialization a *node model*. As $\boldsymbol{\alpha}_o^{(0)} = \sigma_o^{(0)} \mathbf{m}_o^{(0)} + \mathbf{1}$, which entails both the mode distribution and the confidence, we can equivalently set $\boldsymbol{\alpha}_o^{(0)}$ by initializing $\mathbf{m}_o^{(0)}$ and $\sigma_o^{(0)}$ separately, as we do in Section 4.4.2.

Edge model. In Section 4.3, the strength of a relationship between two objects o, o' is abstracted into a weight function $W(o, o', \tau)$, where τ is the relationship type. Since relationships are edges on the graph, we call the realization of the weight function $W(q, q', \tau)$ for a particular τ an *edge model*.

4.4.2 Example Applications

4.4.2.1 Query Intent Classification

We address the problem of query intent classification in the shopping domain. Given a query $q \in Q$ from a query log of an e-commerce website, and a set of predefined product categories C, the task is to map the query to a product category, with a limited number of labeled queries. Additional resources such as existing product metadata may also be leveraged. In the following, we present how we can realize the node and edge models for this task. Note that the set of queries Q in the given query log form the nodes on the graph instead of the generic objects O in Section 4.3.

Node models. Any conventional query classification algorithm can be used to initialize the mode $\mathbf{m}_q^{(0)}$ for each query $q \in Q$, as long as the output of the algorithm can be converted to a probability distribution over the $|C|$ categories, which is taken as the most likely distribution, *i.e.*, the mode distribution $\mathbf{m}_q^{(0)}$. We first introduce a node model based on unigram language models, followed by possible alternatives.

Language node model. As input, we leverage existing product metadata from online shopping sites such as Amazon [6] and Bing Shopping [78]. Each product is associated with metadata that include attributes such as name, brand and description, as well as the category to which the product belongs. From this data, we build a unigram language model θ_i for each category $i \in \{1, \ldots, |C|\}$ based on all observed words from the attribute values of the products in that category, weighted by product popularity. Given a query q, to estimate $\mathbf{m}_q^{(0)}$, we evaluate the query likelihood $p(q|\theta_i), \forall i$, and convert it to a probability distribution over categories by applying Bayes' rule: $p(\theta_i|q) \propto p(q|\theta_i)p(\theta_i)$, using the prior of the categories as estimated by the total popularity of all products in each category.

On the other hand, to initialize the confidence, we build a background model θ_b by considering all products in our metadata. The intuition is that queries with a lower background model likelihood are easier to classify, as a low likelihood means that the words in the query

are rare and more likely to be identified as belonging to those few categories that contain these rare words, resulting in higher classification confidence. On the other hand, a high likelihood implies that the query words are very common and appear in many categories, leading to lower confidence classification. Empirical study shows that the probability of a correct classification of a query is fairly correlated with its negative log likelihood. Thus, we initialize the confidence as follows:

$$\sigma_q^{(0)} = -\log p(q|\theta_b) \tag{4.15}$$

The language node model only requires weak supervision from a database of product metadata, which is available from many online shopping websites. No training labels for individual queries are needed.

Alternative node models. To initialize the mode distribution, we can also use supervised but substantially more accurate classifiers. The disadvantage of using a supervised classifier is that it requires a large number of labeled training queries to achieve good classification accuracy. However, after applying our regularization framework, using the weakly supervised unigram models can achieve comparable results to using a well-trained fully supervised classifier for the node model, as we shall observe in the experiments.

Eq. 4.15 is only a simple heuristic for the purpose of initializing the distribution confidence. Other heuristics can be used, such as the query length (longer queries provide additional features to guide its classification), or the consistency of the outputs from multiple classifiers (queries with more consistent outputs are arguably easier to classify). However, developing more theoretical confidence estimation methods, such as those explored by work on query difficulty [22], is beyond the scope of this chapter.

Edge models. Among many possible relationships between queries, we focus on the lexical (τ_{lex}) and co-click (τ_{click}) similarities.

Lexical edge model. Given two queries q and q' from a query log, we can view each as a

set of words, denoted by $\phi(q)$ and $\phi(q')$, respectively. We can then define the lexical edge model as follows:

$$W(q, q', \tau_{\text{lex}}) = \begin{cases} 1 & \phi(q) \subseteq \phi(q') \text{ or } \phi(q') \subseteq \phi(q) \\ 0 & \text{else.} \end{cases} \tag{4.16}$$

In other words, we connect two queries with an edge if one of them contains all the words in the other, resulting in a symmetric and binary similarity measure. As an example, $W(\text{“canon”}, \text{“canon camera”}, \tau_{\text{lex}}) = 1$, whereas $W(\text{“canon printer”}, \text{“canon camera”}, \tau_{\text{lex}}) = 0$.

Although this edge model is simple, it is more effective in preliminary experiments than other more sophisticated models such as cosine similarity. One possible explanation is that our binary similarity results in less noise than cosine similarity.

Co-click edge model. We can also establish relationships between queries using co-click similarity. Intuitively, if two queries have more clicks that land on product pages belonging to the same category, they are more closely related. In many cases, we may only be able to deduce that two clicks lead to the same category (*e.g.*, they land on the same URL, or URLs with the same prefix such as `example.com/digital_camera/...`), but we may not know the actual category, or how to map the vendor category to ours. Formally, we define the co-click edge model as follows, adapted from the co-citation/click measures in [108, 54]:

$$W(q, q', \tau_{\text{click}}) = \log\left(1 + \sum_{c} \#(q, c)\#(q', c)\frac{N}{N_c}\right). \tag{4.17}$$

In this edge model, each c is a category, which could be a “virtual category” with no direct correspondence to a target category, for reasons discussed above. $\forall q \in Q$, $\#(q, c)$ denotes the number of clicks associated with query q that leads to category c. N is the total number of clicks across all queries, whereas N_c is the number of total clicks that lead to category c. Thus, N/N_c has an effect similar to inverse document frequency, *i.e.*, more popular categories contribute less to the similarity between two queries. We also use a logarithm function to model a sublinear growth of the similarity with respect to the number of clicks.

Additional edge models. The following relationships can also be considered, although they are not used in our experiments.

- User sessions. In the same user session, the second query is often a refinement of the first. Thus, whether two queries co-occur in the same session and the frequency of such co-occurrence can be another edge model.
- Search results. On an existing e-commerce system, a query can retrieve a set of related products (*i.e.*, the search results). Thus, one edge model could be the similarity of the search results of two queries. Alternatively, search results from a generic search engine can also be used, where similarity between retrieved pages can be measured instead.

4.4.2.2 Other Applications

We briefly discuss how we can realize our regularization framework on other applications in IR. As discussed in Section 4.4.2.1, the node model is often easy to realize using an existing classifier for the initial mode distribution and various heuristics for the initial confidence. Hence, for the following applications, we only discuss the choice of their edge models, which are often neglected in conventional classification.

Offer classification. Given a product offer from a vendor, which is presented as a list of attribute-value pairs (*e.g.*, product name, brand, image, description, as well as product-specific attributes such as focal length for cameras), the task is to map it to our predefined categories. (Imagine we are an e-commerce website receiving feeds on offers from different vendors, whose attribute schemas and category taxonomies vary.) The following edge models could be used: (a) the graphical similarity between two product images; (b) the overlap of product-specific attribute names, since products from the same category often share similar attribute names even across vendors; (c) lexical similarity between the names and descriptions of two products.

Text classification. This is a classic task in IR that assigns each document to a class.

With additional resources such as access statistics, social media and embedded hyperlinks, we suggest the following edge models: (a) co-readership, the number of readers who accessed both of the two documents within some time window; (b) social tagging, the consistency of user tags on social sharing sites; (c) hyperlinks in the documents, where similar documents likely point to similar pages; (d) lexical similarity in the contents of two documents.

4.4.3 Target Application Scenarios

As our regularization framework requires multiple iterations over the entire graph $G = (O, E)$, it is not intended for online classification. Instead, it is designed for offline computation, where the precomputed predictions for each object in O is stored in an index. Given a seen object o (*i.e.*, $o \in O$), its precomputed prediction is directly fetched from the index. If o is previously unseen (*i.e.*, $o \notin O$), we can use the node model to classify it on-the-fly. Alternatively, we may potentially look up o's neighbors on the graph using an inverted index, locality sensitive hash, or other means, and subsequently regularize the classification of o using its neighbors for one iteration online.

As an example, let us consider a real scenario involving a live search system on the task of query intent classification. In this situation, as time passes, the live system collects a growing amount of data, including queries and clickthroughs. Thus, it is ideal if our regularization framework can be applied periodically offline (say, on a daily or weekly basis), in order to leverage more data and update the precomputed index accordingly. As our experiments show (Section 4.5.3), DirGraph runs fast enough on a single machine to be computed daily for at least tens of millions of queries.

Alternatively, we can apply our framework and take objects with highly confident predictions as labeled training data. Using these automatically generated training data, we can iteratively train a supervised classifier, extending the approach described in [73].

4.5 Experiments

To analyze the performance of our regularization framework DirGraph, we conduct extensive experiments on the example application of query intent classification, using a large scale real-world dataset. The goal of our experiments is to validate that the proposed framework can effectively exploit heterogeneous contexts (including different relationship types and confidence levels) to ultimately improve classification accuracy, especially in a scenario with limited training data.

4.5.1 Experimental Settings

Dataset. We obtain a hierarchy of product categories from Bing Shopping [78], with a total of 2043 leaf categories (*i.e.*, $|C| = 2043$). Thus, our task is to predict the leaf category for each shopping query.

In our experiments, we use a search log containing 4 million distinct queries. Some of these queries have been labeled by human judges, which we split into two disjoint subsets for training and testing. Specifically, the training set consists of 10K labeled queries for parameter learning, and the test set consists of over 10K labeled queries for evaluation. Although we reserve 10K labeled queries for training, we only use 1K of them in all experiments except in Section 4.5.2.4, where we vary the number of training queries.

We also use clickthrough data for the co-click edge model between queries. There are 11 million clicks associated with about a quarter of all the queries. The fact that most queries do not have any click inspires the adoption of heterogeneous relationship types.

Node model. Unless otherwise stated, we use the *language node model* (Section 4.4.2.1) in all experiments, which is weakly supervised and requires no labeled training queries. However, to demonstrate the robustness of our framework, we also compare it with an alternative node model in Section 4.5.2.2.

(a) "canon 35"

canon 35 mm lens
canon 35 f 2
35 mm wide angle 1.4 canon lens

(b) "hp laptop hard drive"

hard drive 1tb
seagate harddrive
western digital 2tb external

Figure 4.2: Examples of different types of neighbor in graph regularization. (a) Neighbors by lexical similarity; (b) neighbors by co-click similarity.

Accuracy evaluation. For each query, we sort the categories according to their predicted probabilities, upon which we can apply the following metrics.

- Top-K accuracy: fraction of test queries which contain a correct result within the top K hypothesized categories.
- Precision-recall plot: a plot of precision against recall while varying the prediction cut-off threshold.
- Optimal F-score: the best F-score achievable while varying the cut-off threshold.
- Prec @ 0.5 recall: precision at a recall level of 0.5. In practice, precision is often more important than recall—we would rather not predict than suggest a wrong category.

4.5.2 Accuracy Study

4.5.2.1 Case Study

We first present some illustrative results to gain the intuition behind DirGraph.

Query 1: "canon 35". This query is misclassified as `camcorder` by the language model, since both words in the query are not sufficiently discriminating. However, its lexical neighbors reveal some interesting information, as shown in Figure 4.2(a). Given these neighbors, it is not surprising that DirGraph can correctly classify the query as `camera-lens`.

Query 2: "hp laptop hard drive". This query is misclassified as `laptop` by the language model, since the words "hp laptop" strongly suggest laptops. This time, its co-click neighbors, as illustrated in Figure 4.2(b), allow DirGraph to correctly classify it as `hard-drive`.

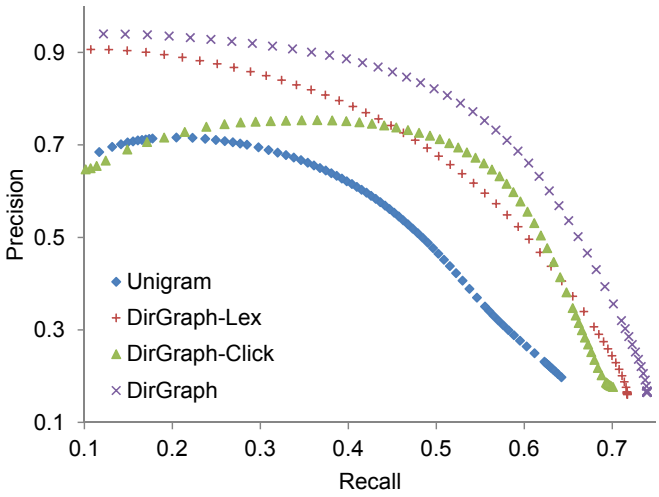


Figure 4.3: Precision-recall plot for different relationship types.

	Top-3 accuracy		Optimal F-score		Prec @ 0.5 recall	
Unigram	0.663		0.564		0.465	
DirGraph-Lex	0.735	(+10.8%)	0.660	(+17.1%)	0.676	(+45.4%)
DirGraph-Click	0.731	(+10.1%)	0.659	(+16.9%)	0.712	(+53.3%)
DirGraph	**0.763**	(+15.0%)	**0.728**	(+29.2%)	**0.821**	(+76.7%)

Figure 4.4: Accuracy comparison for different relationship types.

4.5.2.2 Effects of Heterogeneous Relationship Types

We compare our approach DirGraph to the node model, as well as two weaker schemes of DirGraph that only consider a single type of relationship.

- Unigram: the unigram language model.
- DirGraph-Lex: a weaker scheme of DirGraph, with only lexical similarity.
- DirGraph-Click: a weaker scheme of DirGraph, with only co-click similarity.

Overall results. The results are reported in Figure 4.3 and 4.4. While all graph methods generally outperform the unigram model especially at high recall levels, we make two further observations in the following.

First, neither DirGraph-Lex nor DirGraph-Click clearly dominates the other, each of which only uses a single type of relationship. Normally we would expect DirGraph-Click to

perform better than DirGraph-Lex, since the former employs co-click similarity, a form of user feedback that may be more robust than lexical similarity. For instance, many queries may be lexically similar yet unrelated (*e.g.*, "laptop" and "laptop bag"), or related but exhibit no lexical similarity (*e.g.*, "hp 3 in 1" and "canon printer"). Nevertheless, our results can be explained by the sparsity of the clickthrough data, where only a quarter of the queries have received at least one click, but almost all queries have lexically similar neighbors.

Second, by using both types of relationship, DirGraph performs substantially better than DirGraph-Lex and DirGraph-Click. Apart from the sparsity of a single relationship type, another reason is that the combined evidence of both types of relationship is more informative than any single one alone, thus directly improving queries that have both relationships and indirectly improving queries that connect to neighbors with both.

Result breakdown. We further analyze the performance on two subsets of queries—those with clicks and those without—as shown in Figure 4.5. As expected, DirGraph-Click performs well for queries with clicks, but has no effect on queries without clicks. On the other hand, DirGraph-Lex improves the accuracy for both subsets of queries. It is not surprising that DirGraph outperforms both DirGraph-Click and DirGraph-Lex for queries with clicks, since in this subset, both edge models can improve the classification accuracy. However, we also outperform them for queries without clicks, although using co-click similarity alone has no effect on this subset. In this case, because classification evidence is iteratively propagated across both edge models in DirGraph, improvements to queries with clicks will influence their lexical neighbors, not all of which may have clicks. By combining different types of relationship, we can extend the reach of each individual relationship to influence and improve the performance of more queries via heterogeneous long-range dependencies.

Alternative node model. To demonstrate the robustness of our framework, we also evaluate our approach with an alternative *regression node model* (Regression), based on a supervised logistic regression model trained from a large number of labeled queries, with

(a) Accuracy comparison for queries with clicks

	Top-3 accuracy		Optimal F-score		Prec @ 0.5 recall	
Unigram	0.721		0.609		0.593	
DirGraph-Lex	0.795	(+10.2%)	0.704	(+15.6%)	0.780	(+31.6%)
DirGraph-Click	0.832	(+15.3%)	0.798	(+30.9%)	0.920	(+55.1%)
DirGraph	**0.836**	(+15.9%)	**0.801**	(+31.5%)	**0.925**	(+56.0%)

(b) Accuracy comparison for queries without clicks

	Top-3 accuracy		Optimal F-score		Prec @ 0.5 recall	
Unigram	0.573		0.495		0.238	
DirGraph-Lex	0.641	(+11.9%)	0.583	(+17.9%)	0.445	(+ 87.2%)
DirGraph-Click	0.573	(+ 0.0%)	0.495	(+ 0.0%)	0.238	(+ 0.0%)
DirGraph	**0.648**	(+13.1%)	**0.592**	(+19.7%)	**0.483**	(+102.9%)

Figure 4.5: Result analysis for queries with and without clicks.

	Top-3 accuracy		Optimal F-score		Prec @ 0.5 recall	
Regression	0.719		0.680		0.739	
DirGraph-Lex	0.746	(+3.9%)	0.697	(+2.5%)	0.773	(+ 4.6%)
DirGraph-Click	0.773	(+7.5%)	0.722	(+6.2%)	0.809	(+ 9.5%)
DirGraph	**0.784**	(+9.1%)	**0.729**	(+7.3%)	**0.823**	(+11.4%)

Figure 4.6: Accuracy comparison with regression node model.

query words as features. To initialize the confidence, we use the query length as a heuristic, as our empirical study suggests a correlation between longer queries and better classification accuracy, since longer queries contain more words to guide the classification.

The results of using the regression node model are shown in Figure 4.6. Similar to using the language node model (Figure 4.4), DirGraph outperforms the rest by a clear margin, even though the improvements are not as substantial. Note that Regression is a much stronger node model than Unigram. We hypothesize that this phenomenon is analogous to ensemble methods in machine learning, where it is generally easier to improve over weak learners than strong learners. Interestingly, despite initializing from a much stronger node model, the final results are only slightly better than those initializing from the language node model. Thus, in the subsequent experiments, we only use the language node model, which enjoys the additional benefit of weak supervision.

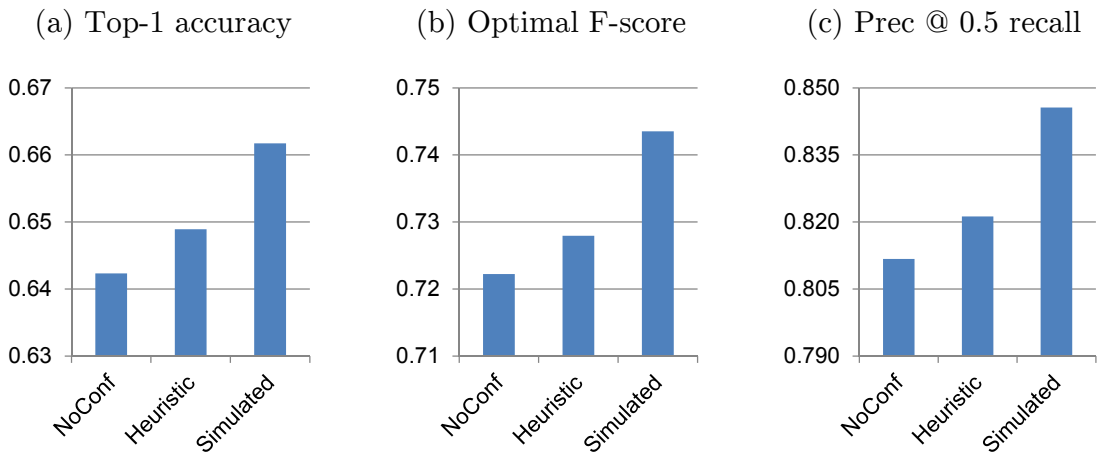


Figure 4.7: Accuracy with different confidence initializations.

4.5.2.3 Effect of Confidence

Next, we investigate the effect of the confidence initialization on regularization. Specifically, we compare our heuristic method in Eq. 4.15 (Heuristic) with the following two strategies.

We call the first strategy NoConf, which applies uniform confidence to each query. That is, no confidence information is utilized.

We label the second strategy Simulated, which generates more reliable confidence values via a simulation. On the one hand, for each labeled query, we sample a random value x from two Gaussian distributions. If the top-1 category predicted by the unigram model is correct, then $x \sim N(0.8, 0.1)$; otherwise, $x \sim N(0.2, 0.1)$. Subsequently, we initialize the confidence of the labeled query with x, clipped to the range $[0, 1]$. On the other hand, for each unlabeled query, we initialize its confidence to 0.512, which is the top-1 accuracy of the unigram model on the test queries.

The results of the different strategies are presented in Figure 4.7. Although Heuristic only uses a simple heuristic (Eq. 4.15) to initialize the confidence, it performs better than NoConf, which is not confidence-aware. On the other hand, Simulated performs the best, since it initializes confidence in a more reliable way. Thus, with more sophisticated confidence estimation, we can expect even better results from our framework.

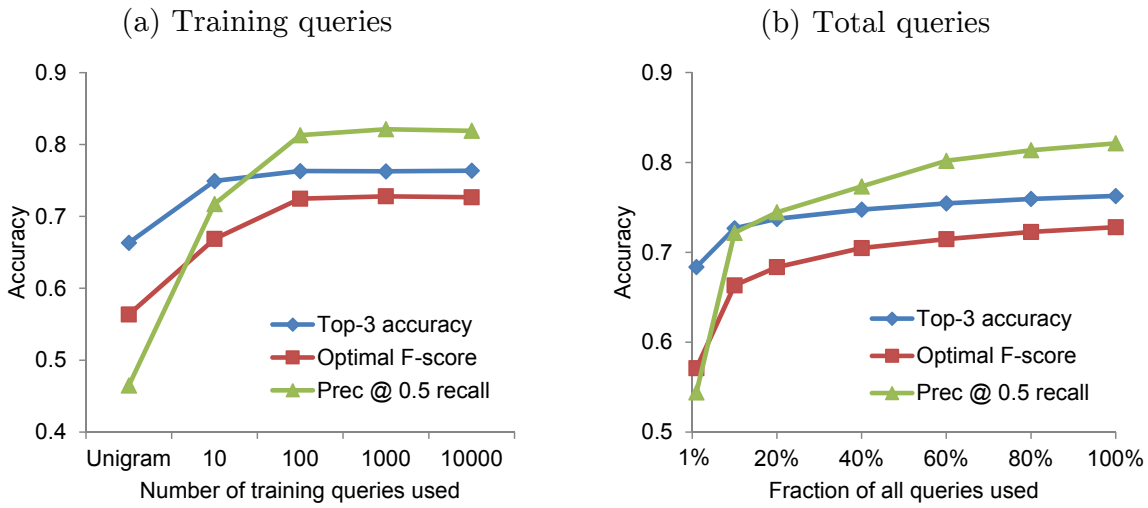


Figure 4.8: Accuracy versus number of queries.

4.5.2.4 Varying Amount of Data

In the following experiments, we vary the amount of data in three ways: the number of labeled training queries, total queries, and clickthroughs. These experiments are important, since in practical scenarios data are collected from a live system continuously.

Number of training queries. Recall from Section 4.3.3 that we require some labeled queries as training data in order to learn the parameters Λ. We examine the number of training queries needed to reach convergence on accuracy. Varying from 10 to 10K training queries, we record the corresponding performance in Figure 4.8(a). The results reveal that with merely 10 training queries, we can already achieve a substantial improvement over the unigram baseline. Furthermore, with as few as 100 training queries, the performance has nearly converged. Note that 100 training queries are much fewer than the 2043 categories in this classification task. This is not surprising as we are not directly modeling the categories, which would require 2000+ training queries at the minimum, one for each category. Instead, we use the training queries only to learn Λ, the weights of different relationships. Coupled with the weakly supervised language node model, DirGraph reaches convergence with very few training queries in total.

Number of total queries. The majority of our queries are unlabeled and not directly used for parameter learning. However, as we increase the number of unlabeled queries, we also improve the connectivity of the graph. To evaluate the effect, we randomly sample a subset of all queries, and apply DirGraph using this subset as the total set of queries. The sample size varies between 1% and 100% of all queries. To eliminate any differences due to labeled training queries, our samples include the same set of 1K training queries throughout. The results are presented in Figure 4.8(b). Despite using the same set of training queries, the accuracy of DirGraph increases as the number of total queries used increases. Although most of the improvement is achieved when the number of queries is initially increased from 1% to 20%, further increases result in smaller but continual improvements. From this trend, we expect that the accuracy can be further improved, but to a lesser extent, with an even larger query set.

Number of clickthroughs. Finally, we investigate the effect of varying the number of clickthroughs. Specifically, we randomly sample between 0% and 100% of clickthroughs from our query log, while using all queries and the same set of training queries. Figure 4.9(a) and (b) illustrate the effect of clickthroughs on Reg-Click and our approach, respectively. (The number of clickthroughs does not affect the performance of Reg-Lex, which only utilizes the lexical edge model.) In both methods, using a larger number of clickthroughs improves accuracy. Similar to the effect of increasing the number of total queries, the improvements diminish as more clickthroughs are used. Nevertheless, the improvements have not converged, suggesting that additional clickthrough data will further increase performance.

4.5.3 Efficiency Study

Although DirGraph is designed to be applied offline periodically (Section 4.4.3), efficiency is still important. For instance, to apply it daily, it should not take longer than a day. The following experiments evaluate the efficiency of our approach.

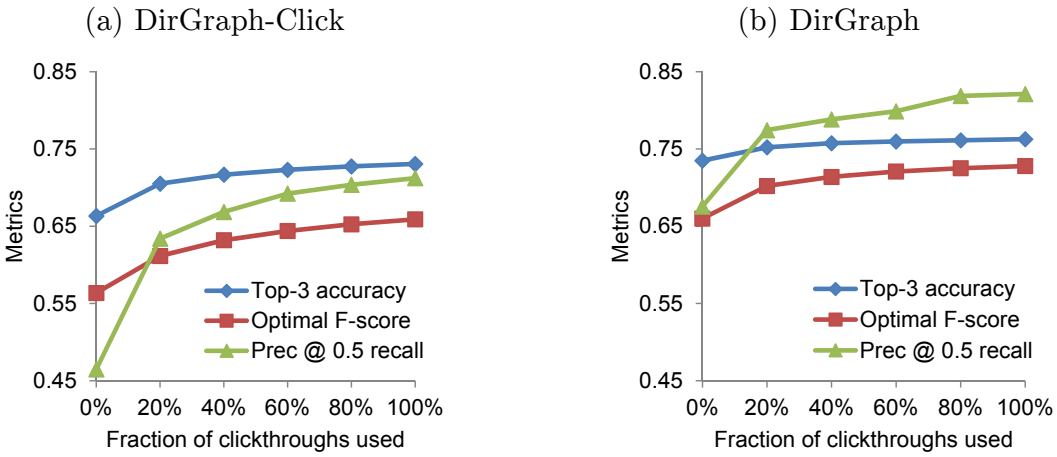


Figure 4.9: Accuracy versus the number of clickthroughs.

4.5.3.1 Rate of Convergence

As DirGraph is iterative, the number of iterations required to achieve convergence is crucial to efficiency. Our experiments show that convergence is fast, with 2 to 4 iterations in most settings. In particular, when using all available data, Figure 4.10(a) illustrates the convergence of classification accuracy with a precision-recall plot. Iteration-0 corresponds to the initial node model. Regularization in Iteration-1 results in a larger improvement, whereas Iteration-2 shows smaller improvement. There is virtually no improvement from Iteration-3 (not shown in the figure), which indicates that the accuracy has converged. From another perspective, Figure 4.10(b) illustrates the sequence of minimum errors in each minimization step (Eq. 4.14), which converges after only 3 iterations.

4.5.3.2 Execution Time

We start with a simple complexity analysis. As we scan all edges and nodes of the graph once in each iteration, the complexity is $O((E+V)T)$, where T is the number of iterations, E is the number of edges and V is the number of nodes (*i.e.*, queries). E potentially increases faster than V—the number of edges introduced by a new query potentially grows with the number of existing queries. Hence, the overall execution time would grow faster than a linear rate as we introduce more queries. Note that we do not consider the time to build the

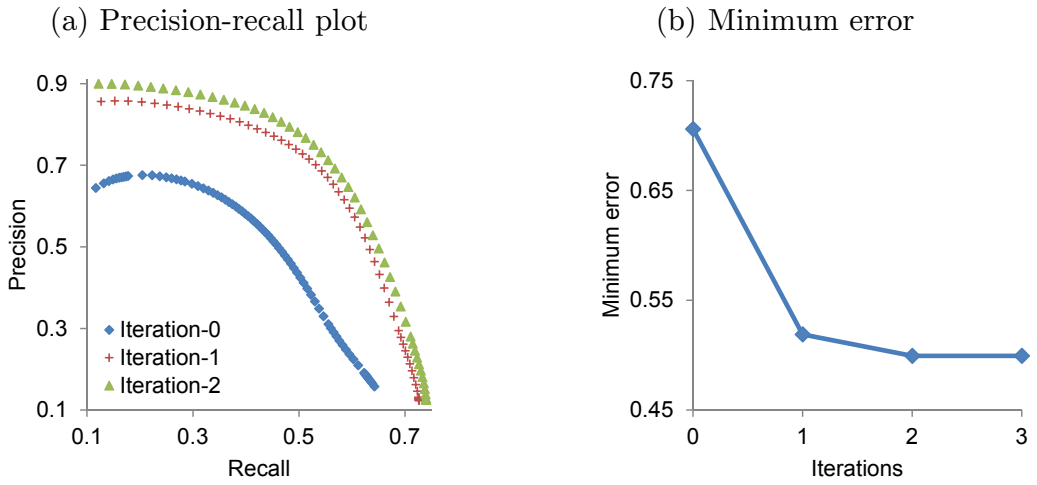


Figure 4.10: Convergence of (a) accuracy; (b) minimum error.

graph, which can be incrementally constructed whenever a new query is collected. Thus, it is unnecessary to build the graph from scratch every time.

In Figure 4.11, we record the execution time on a single 8-core PC with 32GB memory. Consistent with our analysis above, the execution time grows slightly faster than a linear rate against the number of total queries. Nevertheless, the execution time of under 20 minutes when all 4 million queries are used is still quite reasonable, as it means our framework can be applied frequently (such as on a daily basis) at least on the scale of tens of millions of queries. Furthermore, as the number of new queries within a fixed time interval tends to decrease over time, the total number of queries should grow in a sublinear rate. Lastly, as Section 4.5.2.4 shows, including more queries brings in gradually less boost in accuracy. Thus, we do not expect to handle an arbitrarily large number of queries, since it would not achieve a noticeable benefit beyond a certain point.

4.6 Conclusion

In this chapter, we guided the task of classification with heterogeneous contexts—different relationship types as well as confidence levels. To incorporate both dimensions of heterogeneity, we developed a Dirichlet-based graph regularization framework called DirGraph, which

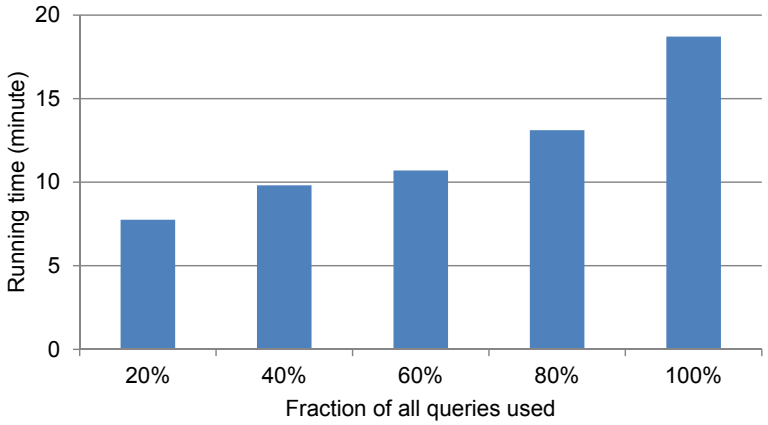


Figure 4.11: Execution time versus number of total queries.

corresponds to a special form of backward random walk. Building on the proposed framework, we discussed example realizations of several real-world scenarios, particularly query intent classification in the shopping domain. Finally, we conducted extensive experiments on a large-scale real-world dataset to demonstrate the advantages of our approach.

Chapter 5

Graph-based Smoothness Framework

As attempted in previous chapters, forward and backward random walks can be used to address different graph-based searching and mining problems. In this chapter, we will investigate the fundamental principle governing and unifying these two forms of random walks, in a generic graph-based semi-supervised learning setting. In particular, random walk techniques boil down to the notion of *smoothness*. As the central notion in semi-supervised learning, smoothness is often realized on a graph representation of the data. In this chapter, we study two complementary dimensions of smoothness: its pointwise nature and probabilistic modeling. While no existing graph-based work exploits them in conjunction, we encompass both in a novel framework of Probabilistic Graph-based Pointwise Smoothness (PGP), building upon two foundational models of data closeness and label coupling. Interestingly, forward and backward random walks simply correspond to two different "modes" of smoothness, respectively, under the same unifying framework. Theoretically, we provide an error and robustness analysis of PGP. Empirically, we conduct extensive experiments to show the advantages of PGP.

5.1 Introduction

As labeled data is often scarce, semi-supervised learning (SSL) that can exploit unlabeled data in addition to labeled data is often desirable. Consider a random tuple (X, Y), where a data point $X \in \mathcal{X} = \{x_1, \ldots, x_{|\mathcal{X}|}\}$ has a label $Y \in \mathcal{Y}$. We observe labeled data $\mathcal{L}$ comprising i.i.d. samples of (X, Y), and unlabeled data $\mathcal{U}$ comprising i.i.d. samples of X.

Typically $|\mathcal{U}| \gg |\mathcal{L}|$. Potentially, we may only observe a partial $\mathcal{X}$ via $\mathcal{L}$ and $\mathcal{U}$. Our task is to predict the label for every $x_i \in \mathcal{U}$.

Towards effective SSL, graph-based smoothness has attracted much research interest. In particular, the *smoothness* statement is central to SSL [111, 24]: *if two points x_i, x_j are close, their respective labels y_i, y_j are likely to be the same.* The literature further suggests that it is more effective to consider smoothness on the low-dimensional manifold, where the high-dimensional data roughly live. As widely recognized, graphs are often used as a proxy for the manifold [16, 113, 107, 14, 65, 98]. Specifically, each point x_i is a node on a graph, and two points x_i, x_j may be connected by an edge weighted W_{ij}. The weight matrix W aims to capture the pairwise geodesic distance on the manifold. In other words, the graph reveals the structures on the manifold.

Unfortunately, although graph-based methods universally hinge on smoothness, their realizations fall short. As the key insight of this chapter, we advocate that smoothness shall be pointwise in nature and probabilistic in modeling.

Nature: Pointwise smoothness. Smoothness shall inherently occur "everywhere," to relate the behavior of *each point* to that of its close points. We call this the *pointwise* nature of smoothness. As recently identified [84], precisely expressing the pointwise nature boils down to the two following aspects.

- How do we decide if a data point is close to another? The smoothness statement lacks a concrete definition of closeness. Thus, we need a *data closeness model* to explicitly define this. (P1)

- How is the behavior of two close points related? The smoothness statement requires that "their labels are likely to be the same", which is rather vague. Thus, we need a *label coupling model* to explicitly relate their label behavior. (P2)

Surprisingly, to date, no existing graph-based method realizes pointwise smoothness. While it has been studied in non-graph based settings [84, 71, 96], previous graph-based

methods treat smoothness in an *aggregate*, rather than pointwise, manner. Most of the existing graph-based approaches optimize an energy function in a random field [113, 112, 48] or a cost function [107, 14, 98] over the graph. An energy or cost function *aggregates* all pairwise differences between neighboring points across the entire graph. By minimizing the aggregated difference, some "average" smoothness is achieved. However, such aggregation is not designed for and thus does not necessarily enforce smoothness at every point—it is unclear how an aggregate function can precisely express the pointwise nature of smoothness, in terms of the two aspects (P1 & P2). After all, there exist different cost functions varying greatly in actual forms (*e.g.*, squared error, soft margin loss, or probability divergence), with limited justification to favor one over another.

Modeling: Probabilistic smoothness. Pointwise smoothness shall be modeled *probabilistically* in both aspects (P1 & P2), to ultimately infer $p(Y|X)$. First, how close is sufficiently close is difficult to be reliably captured by deterministic binary decisions (P1). Second, the smoothness statement that "their labels are likely to be the same" is meaningless [84] unless it is exploited in probabilistic terms (P2). Within a probabilistic framework, eventually each point can be classified based on $p(Y|X)$, given i.i.d. samples. Furthermore, probabilistic modeling conveys some concrete benefits, such as integrating class priors $p(Y)$ in a more principled way, naturally supporting multi-class tasks, and facilitating client applications that require probabilities as input.

We note that existing probabilistic modeling in graph-based settings [98, 30, 53, 8] only supports aggregate, but not pointwise, smoothness.

Proposed framework. We propose the framework of Probabilistic Graph-based Pointwise Smoothness (PGP), hinging on two foundations that address the pointwise nature of smoothness probabilistically on a graph.

To begin with, we need a *data closeness model* to determine if a point is close to another (P1). Since the graph captures the pairwise geodesic distance on the manifold, a random

walk on the graph [76]—which moves from X to X' in each step—naturally "connects" X and X' as close points on the manifold. Hence, for a *pair* of random points (X, X') such that X is close to X', we can describe their distribution $p(X, X')$ using the *second-order* stationary distribution of the random walk. In contrast, the distribution of a *single* point $p(X)$ has been traditionally represented by the *first-order* stationary distribution.

Next, we also need a *label coupling model* to relate the label behavior of a point x_i to that of its close points (P2). We leverage the notion of *statistical indistinguishability* [50]. In particular, whether X is x_i, or X is some point close to x_i, the label Y of X shall be produced in an indistinguishable manner. In other words, we cannot tell apart the distributions of Y in these two cases.

Together, these two foundations constitute our smoothness framework, which further entails a solution to SSL. While the given labels naturally constrain the labeled data, our smoothness framework axiomatizes a set of probability constraints on the unlabeled data. Solving these constraints eventually infers $p(Y|X)$ for class prediction. Note that the constraints can be either discriminative over $p(Y|X)$ or generative over $p(X|Y)$, resulting in two different "modes" of smoothness, which can be interpreted as the backward and forward random walks, respectively.

Finally, we present a theoretical analysis of our solution. First, to see that PGP can utilize both labeled and unlabeled data to improve its performance, we derive a generalization error in $\mathcal{L}$ and $\mathcal{U}$. Second, to show that PGP is not sensitive to noisy input graphs, we assess the robustness of our solution against graph perturbations.

Our contributions. We summarize the contributions in this chapter as follows.

- **Framework:** We propose a unifying smoothness framework PGP, the first work to realize pointwise smoothness on a graph probabilistically.
- **Analysis:** We conduct an error and robustness analysis of PGP.
- **Experiments:** We demonstrate the advantages of PGP empirically.

5.2 Smoothness Framework

To express the pointwise nature of smoothness, we must address its two aspects. Under a probabilistic graph-based framework, we propose a data closeness model to capture how a point is close to another (Section 5.2.1), as well as a label coupling model to conceptualize how the label behavior of a point is related to that of its close points (Section 5.2.2).

5.2.1 Data Closeness Model (P1)

We first propose a probabilistic model for capturing data closeness on the graph.

Graph. For a set of points $\mathcal{X} = \{x_1, \ldots, x_{|\mathcal{X}|}\}$, we construct a graph G to capture the pairwise geodesic distance on the underlying manifold. Each point $x_i \in \mathcal{X}$ is a vertex of G, and each pair of points (x_i, x_j) form an edge of G with a weight W_{ij}. W_{ij} is also known as the *affinity* between x_i and x_j, an approximate description of the geodesic distance between the two points. W is often defined as follows:

$$W_{ij} = \begin{cases} \exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right) & i \neq j \\ 0 & i = j, \end{cases} \tag{5.1}$$

where $\|\cdot\|$ is a symmetric distance function, and σ is a scaling parameter. In our notation, unquantified indices such as i, j belong to $\{1, \ldots, |\mathcal{X}|\}$, unless stated otherwise.

Random walk-based closeness. As argued in Section 5.1, it is difficult to reliably capture how close is sufficiently close in a deterministic manner. In order to develop probabilistic closeness, we need to represent the event that two points, say x_i and x_j, are close.

We capture the closeness event based on a random walk on the graph. Consider a random walk on G visiting a sequence of points $\{V_t : t = 0, 1, \ldots\}$. While traditionally a *visit* at x_i ($V_t = x_i$) models a single point x_i, a *traversal* walking from a point x_i to another x_j ($V_t = x_i, V_{t+1} = x_j$) naturally "connects" x_i and x_j to imply that x_i is close to x_j on the underlying manifold.

Note that our use of random walk serves a novel purpose. It specifically models the first pointwise aspect (P1) of relating the points X through the closeness event, which, together with the second aspect (P2) of relating the labels Y in Section 5.2.2, defines smoothness pointwisely. On the contrary, existing use of random walk [101, 8, 104] models the "propagation" of label Y among X altogether, without treating the two aspects explicitly.

Formally, let $(X = x_i, X' = x_j)$ denote the event that x_i is close to x_j, which follows the distribution of observing a random walk traversal from x_i to x_j in the long run as $t \to \infty$. Hence, (X, X') is a pair of limiting random variables in the sense that a traversal (V_t, V_{t+1}) converges *in distribution* to (X, X') jointly:

$$(V_t, V_{t+1}) \xrightarrow{d} (X, X'). \tag{5.2}$$

In other words, (X, X') describes the closeness between two points with the joint *second-order* limit, while X describes a single point with the marginal *first-order* limit. Their convergence will be shown later.

Probability space of closeness. We describe the probability space of the random walk-based closeness model.

Sample space. An outcome that x_i is close to x_j is a pair of points (x_i, x_j), which corresponds to a random walk traversal from x_i to x_j. Hence, the sample space is $\Omega = \mathcal{X}^2$. An outcome can be denoted by a pair of random variables $(X, X') \in \Omega$, as defined in Eq. 5.2.

Events. As discussed, each outcome (x_i, x_j) itself is an event that x_i is close to x_j, *i.e.*, $\{(X, X') \in \Omega : X = x_i, X' = x_j\}$ or denoted $(X = x_i, X' = x_j)$. In order to relate the behavior of two close points, we are also interested in the following two events. First, $\{(X, X') \in \Omega : X = x_i\}$, or denoted $X = x_i$, is the event of *observing X as a point x_i*. It corresponds to a traversal from x_i to some point. Second, $\{(X, X') \in \Omega : X' = x_i\}$, or denoted $(X, X' = x_i)$, is the event of *observing X as some point close to x_i* (*i.e.*, X is implicitly constrained by $X' = x_i$). It corresponds to a traversal from some point to x_i.

Without loss of generality, here we treat X as the random variable of interest, and our ultimate goal is to estimate $p(Y|X)$. However, we could also treat X' as the variable of interest, and find $p(Y'|X')$ in a symmetric manner given that W is symmetric.

Probability measure. Finally, we evaluate the probability of the events. The random walk can be formally represented by a transition matrix Q such that

$$Q_{ij} = W_{ij}/Z_i, \quad \text{where } Z_i \triangleq \textstyle\sum_j W_{ij}. \tag{5.3}$$

As (V_t, V_{t+1}) converges in distribution to (X, X'), the closeness event $(X = x_i, X' = x_j)$ obeys the *second-order* stationary distribution of the random walk:

$$p(X = x_i, X' = x_j) = \lim_{t\to\infty} p(V_t = x_i, V_{t+1} = x_j). \tag{5.4}$$

As established in the following, a unique second-order stationary distribution exists. As a further consequence, the probability of the events can also be computed.

PROPOSITION 5.1 (PROBABILITY OF EVENTS).

(a) The limit of $p(V_t, V_{t+1})$ as $t \to \infty$ exists uniquely.

(b) Given that $(V_t, V_{t+1}) \xrightarrow{d} (X, X')$,

$$\forall ij, \quad p(X = x_i, X' = x_j) \propto W_{ij}, \tag{5.5}$$

$$\forall i, \quad p(X = x_i) = p(X, X' = x_i) \propto Z_i. \tag{5.6}$$

Intuitively, Eq. 5.5 means that the stronger affinity between x_i and x_j, the more likely they are close. Second, Eq. 5.6 implies that observing x_i is as likely as observing a point close to x_i, which is not surprising given that two close points lie near each other.

5.2.2 Label Coupling Model (P2)

Next, we propose a label coupling model to relate the label behavior of two close points. In our realization, the label Y of X distributes similarly whether X is x_i itself, or X is some point close to x_i. That is, $p(Y|X = x_i)$ and $p(Y|X, X' = x_i)$ shall be alike.

Indistinguishability. We leverage the concept of *statistical indistinguishability* [50]: two distributions are statistically indistinguishable if they cannot be told apart to some extent.

DEFINITION 5.1 (STATISTICAL INDISTINGUISHABILITY). Two distributions D_1 and D_2 are ϵ-*statistically indistinguishable* if and only if $\frac{1}{2}\left\|D_1 - D_2\right\|_1 \leq \epsilon$.

In our context, $p\left(Y|X = x_i\right)$ and $p\left(Y|X, X' = x_i\right)$ shall be statistically indistinguishable. In other words, the label Y of X is produced in an indistinguishable manner regardless of X being x_i or a point close to x_i.

Label Coupling. To achieve indistinguishability, x_i's label shall distribute similarly to that of a point close to x_i. At the same time, some "distrust" of the close points shall be allowed, as small variances in their labels are still expected. These factors can be accounted for by a simple mixture:

$$p\left(Y|X = x_i\right) = (1 - \alpha)p\left(Y|X, X' = x_i\right) + \alpha D, \tag{5.7}$$

where $\alpha \in (0, 1)$ is a parameter, and D is the distribution to fall back on when the close points are not trusted. In the distrust case, we assign x_i to an "unknown" class $\phi \notin \mathcal{Y}$, *i.e.*, $D(y) = 0, \forall y \in \mathcal{Y}$ and $D(\phi) = 1$. Our label coupling model represented by this mixture formally satisfies statistical indistinguishability.

PROPOSITION 5.2 (LABEL COUPLING). Given that Eq. 5.7 holds, the label distribution of x_i, $p\left(Y|X = x_i\right)$, is α-*statistically indistinguishable* from the label distribution of some point close to x_i, $p\left(Y|X, X' = x_i\right)$.

Note that Eq. 5.7 couples the label distributions in a discriminative form of $p\left(Y|X\right)$. To model the generative probability $p\left(X|Y\right)$, we also derive its generative counterpart below.

$\forall y \in \mathcal{Y}, \forall x_i \in \mathcal{X}$,

$$\begin{aligned} & p\left(X = x_i | Y = y\right) \\ = & \, p\left(Y = y | X = x_i\right) p\left(X = x_i\right) / p\left(Y = y\right) \\ = & \, (1-\alpha) p\left(Y = y | X, X' = x_i\right) p\left(X, X' = x_i\right) / p\left(Y = y\right) \\ = & \, (1-\alpha) p\left(X, X' = x_i | Y = y\right). \end{aligned} \tag{5.8}$$

In particular, D is eliminated since $D(y) = 0, \forall y \in \mathcal{Y}$, *i.e.*, points of class $y \in \mathcal{Y}$ cannot be generated from D. The intuition is that indistinguishability slowly "fades" along a "chain" of close points due to the $1 - \alpha$ factor.

5.3 Probability Constraint-based Learning

Under the smoothness framework in Section 5.2, we develop a set of probability constraints. The constraints can either assume the generative form over $p\left(X|Y\right)$, or the discriminative form over $p\left(Y|X\right)$. Although the ultimate goal is $p\left(Y|X\right)$, generative models that learn $p\left(X|Y\right)$ and $p\left(Y\right)$ are often favorable in SSL [24]. Thus, we focus on the generative constraints, and show that a unique solution satisfying the constraints exists, which can be leveraged for class prediction.

Nevertheless, we also include a preliminary discussion on the discriminative counterpart under the same smoothness framework. As it turns out, forward and backward random walks represent two different "modes" of smoothness, which simply correspond to two different probabilistic formulations (*i.e.*, generative and discriminative models) under one unifying framework for smoothness.

Finally, we contrast with existing methods to highlight the major novelty in PGP.

5.3.1 Generative Probability Constraints

For each $y \in \mathcal{Y}$, we aim to learn the generative distribution

$$\pi_y \triangleq (\pi_{y1}, \ldots, \pi_{y|\mathcal{X}|}), \tag{5.9}$$

where $\pi_{yi} \triangleq p\,(X = x_i|Y = y)$. To find π_y, we develop and solve a set of constraints on π_y. On the one hand, for $x_i \in \mathcal{L}$ the constraints can be modeled using the known labels. On the other hand, while there is no known label for $x_i \notin \mathcal{L}$, the constraints can be modeled using points close to x_i, based on our smoothness framework.

Labeled points. We rewrite $p\,(X = x_i|Y = y)$ for $x_i \in \mathcal{L}$, relating it to $p\,(Y = y|X = x_i)$ which can be estimated from the known labels (as we will see in Section 5.3.2). For a particular $y \in \mathcal{Y}$,

$$\begin{aligned} & p\,(X = x_i|Y = y) \\ = \; & p\,(Y = y|X = x_i)\, p\,(X = x_i)\, /p\,(Y = y) \\ \propto \; & p\,(Y = y|X = x_i)\, Z_i. \end{aligned} \tag{5.10}$$

The proportionality follows from the conclusion that $p\,(X = x_i) \propto Z_i$ (Proposition 5.1), for a given $y \in \mathcal{Y}$. We can transform this result into a set of constraints on π_y below, where K is the sum of π_{yi} for labeled points, and θ_{yi} is the proportion each π_{yi} gets from the sum K according to Eq. 5.10. Note that we write $p\,(y|x_i)$ as a shorthand for $p\,(Y = y|X = x_i)$, when there is no ambiguity.

Constraints on Labeled Data:

$$\pi_{yi} = K \cdot \theta_{yi}, \quad \forall i : x_i \in \mathcal{L}. \tag{5.11}$$

$$\text{where} \quad K = \sum_{i:x_i \in \mathcal{L}} \pi_{yi},$$

$$\theta_{yi} = \frac{p\,(y|x_i)\, Z_i}{\sum_{k:x_k \in \mathcal{L}} p\,(y|x_k)\, Z_k}.$$

Unlabeled points. We also rewrite $p(X = x_i|Y = y)$ for unlabeled points $x_i \notin \mathcal{L}$, relating it to that of its close points. Specifically, for a given $y \in \mathcal{Y}$,

$$\begin{aligned}
& p(X = x_i|Y = y) \\
\overset{1}{=} & (1-\alpha) p(X, X' = x_i|Y = y) \\
\overset{2}{=} & (1-\alpha) \textstyle\sum_j p(X = x_j, X' = x_i|Y = y) \\
\overset{3}{=} & (1-\alpha) \textstyle\sum_j p(X' = x_i|X = x_j, Y = y)\, p(X = x_j|Y = y) \\
\overset{4}{=} & (1-\alpha) \textstyle\sum_j p(X' = x_i|X = x_j)\, p(X = x_j|Y = y) \\
\overset{5}{=} & (1-\alpha) \textstyle\sum_j W_{ji}/Z_j \cdot p(X = x_j|Y = y)
\end{aligned} \tag{5.12}$$

Step 1 is the generative form of smoothness (Eq 5.8). In step 2, we relate x_i to each x_j through their second-order (joint) distributions, where each x_j has a different probability of being close to x_i. In step 3, we apply Bayes' rule. In step 4, based on our closeness model, given $X = x_j$, $X' = x_i$ only depends on W and is conditionally independent of Y. In step 5, $p(X' = x_i|X = x_j)$ is simply the transition probability $Q_{ji} = W_{ji}/Z_j$. This result imposes another set of constraints on π_y.

Constraints on Unlabeled Data:

$$\pi_{yi} = (1-\alpha) \textstyle\sum_j W_{ji}/Z_j \cdot \pi_{yj}, \quad \forall i : x_i \notin \mathcal{L}. \tag{5.13}$$

5.3.2 Solving the Generative Constraints

The goal is to solve π_y that satisfies the constraints on labeled and unlabeled data. In particular, we can show that π_y is the stationary distribution of some Markov chain with $\mathcal{X}$ as its state space. Intuitively, the unlabeled constraint (Eq. 5.13) already tells us how state x_j transitions to each $x_i \notin \mathcal{L}$. Thus, we only need to deduce the transition to each state $x_i \in \mathcal{L}$. Proposition 5.3 establishes the exact transition between the states.

PROPOSITION 5.3 (SOLUTION TO GENERATIVE CONSTRAINTS). $\forall y \in \mathcal{Y}$, suppose π_y satisfies the constraints in Eq. 5.11 and 5.13. The following can be established.
(a) π_y is the stationary distribution of a Markov chain $\mathcal{C}$ with states $\mathcal{X}$ and transition matrix P, where

$$P_{ji} = \begin{cases} \dfrac{\sum_{k:x_k \in \mathcal{L}} W_{jk} + \alpha \sum_{k:x_k \notin \mathcal{L}} W_{jk}}{Z_j} \cdot \theta_{yi} & i : x_i \in \mathcal{L} \\ \dfrac{(1-\alpha) W_{ji}}{Z_j} & i : x_i \notin \mathcal{L}, \end{cases} \tag{5.14}$$

(b) The stationary distribution of $\mathcal{C}$ exists uniquely.

In fact, Eq. 5.14 means that $\pi_y = \pi_y P$. If we rewrite it as element-wise operations, we see that a constraint is placed on every individual point.

Class prediction. Based on π_y, we predict the label y_i for x_i as follows:

$$\begin{aligned} y_i &= \arg\max_{y \in \mathcal{Y}} p\left(Y = y | X = x_i\right) \\ &= \arg\max_{y \in \mathcal{Y}} p\left(X = x_i | Y = y\right) p\left(Y = y\right). \end{aligned} \tag{5.15}$$

Here $p\left(X = x_i | Y = y\right)$ is simply π_{yi}, and $p\left(Y = y\right)$ is the class prior which can be estimated from $\mathcal{L}$ or given by domain experts.

Solution computation. Next, we discuss how π_y can be computed. Proposition 5.3 entails that π_y can be found iteratively, if the transition matrix P is *known*:

$$\pi_y^{(t+1)} = \pi_y^{(t)} P, \qquad t = 0, 1, 2, \ldots, \tag{5.16}$$

where $\pi_y^{(t)}$ converges uniquely as $t \to \infty$ for an arbitrary initial distribution $\pi_y^{(0)}$.

Solution estimation. We can only find a solution estimator $\hat{\pi}_y$ since P is *unknown*—P is a function of W and θ_y, both of which can only be estimated, resulting in two types of error. First, there is *data sampling error*. W is defined on $\mathcal{X}$, but only a partial $\hat{\mathcal{X}}$ is observed through $\mathcal{L}$ and $\mathcal{U}$. Thus, we can only construct an estimator $\hat{W}$ using the incomplete $\hat{\mathcal{X}}$. Second, there is *label sampling error*. θ_y is defined by $p\left(y | x_i\right), \forall x_i \in \mathcal{L}$, but

$p(y|x_i)$ is unknown. We can only estimate it as the sample mean based on the given labels, $\hat{p}(y|x_i) = |\mathcal{L}_y \cap \mathcal{L}_{x_i}|/|\mathcal{L}_{x_i}|$ where $\mathcal{L}_y$ is the set of all samples with y in $\mathcal{L}$, and $\mathcal{L}_{x_i}$ is the set of all samples with x_i in $\mathcal{L}$. (As the samples are i.i.d., there can be different samples with y or x_i.) Subsequently, we obtain an estimator $\hat{\theta}_y$ based on $\hat{p}(y|x_i)$.

Efficiency. While efficiency is not our focus in this chapter, PGP can be solved efficiently using standard iterative techniques, and its complexity is comparable to, if not better than, most existing SSL methods. In terms of time efficiency, if we use a widely adopted kNN graph, the cost is $O(k|\mathcal{X}|s)$, where s is the number of iterations needed for convergence. In typical settings, $k \sim 10$ and $s \sim 100$. Although constructing an exact kNN graph can take quadratic time, an approximate graph is often adequate in most scenarios [25]. In terms of space efficiency, we can store the kNN graph in a sparse representation (as an adjacency list), thus needing only $O(k|\mathcal{X}|)$ space.

5.3.3 Discussion: Discriminative Constraints and Interpretation

Our smoothness framework can accommodate both generative and discriminative formulations. In other words, our framework unifies two different "modes" of smoothness, each corresponding to one of the probabilistic formulations. In particular, parallel to our earlier derivation, we are able to derive a discriminative counterpart to the generative model, in terms of probability constraints over $p(Y|X)$.

To develop the discriminative constraints, we start from the indistinguishability-based label coupling model below, which is already in the discriminative form. (D is eliminated as $D(y) = 0, \forall y \in \mathcal{Y}$.)

$$\begin{aligned} p(Y|X = x_i) &= (1-\alpha)p(Y|X, X' = x_i) + \alpha D, \\ &= (1-\alpha)p(Y|X, X' = x_i). \end{aligned} \tag{5.17}$$

Subsequently, a corresponding set of probability constraints in terms of $p(Y|X)$ can be

developed for unlabeled data. For $x_i \in \mathcal{U}$ and a particular $y \in \mathcal{Y}$,

$$\begin{aligned} p\left(Y=y|X=x_i\right) &\overset{1}{=} (1-\alpha)p\left(Y=y|X, X'=x_i\right) \\ &\overset{2}{=} (1-\alpha)\sum_j p\left(Y=y, X=x_j|X, X'=x_i\right) \\ &\overset{3}{=} (1-\alpha)\sum_j p\left(Y=y|X=x_j, X'=x_i\right) p\left(X=x_j|X'=x_i\right) \\ &\overset{4}{=} (1-\alpha)\sum_j p\left(Y=y|X=x_j\right) p\left(X=x_j|X'=x_i\right) \\ &\overset{5}{=} (1-\alpha)\sum_j p\left(Y=y|X=x_j\right) W_{ji}/Z_i \end{aligned} \tag{5.18}$$

Step 1 is given by the indistinguishability model. Step 2 expands into the joint distributions. Step 3 is an application of Bayes' rule. In step 4, the label Y of X, given $X = x_j$, is conditionally independent of X'. In step 5, we have $p\left(X=x_j|X'=x_i\right) = p\left(X=x_j, X'=x_i\right)/p\left(X'=x_i\right) = W_{ji}/Z_i$. Thus, $p\left(Y=y|X=x_i\right)$ is rewritten, relating the label distribution of x_i to that of its close points x_j.

The generative and discriminative constraints on unlabeled data in Eq. 5.13 and 5.18, respectively, appear symmetric. They differ in their normalizations—the generative constraints (Eq. 5.13) normalize each summand by Z_j, while the discriminative constraints (Eq. 5.18) normalize each summand by Z_i. Interestingly, such different normalizations correspond to two different but symmetric forms of random walk, namely the *forward* and *backward* random walks. Loosely speaking, on the one hand, the generative constraints correspond to the forward walk, which starts from a labeled point and we are interested in the probability of reaching each x_i. On the other hand, the discriminative constraints correspond to the backward walk, which starts from x_i and we are interested in the probability of reaching a labeled point. These two forms of random walk not only travel in "opposite" directions, but also convey symmetric and complementary semantics: probabilistic recall and precision in Chapter 2, or importance and specificity in Chapter 3.

In summary, the generative and discriminative formulations of smoothness, corresponding

to forward and backward walks, can be unified under one framework. However, in the rest of our discussion, we only assume the solution based on the generative formulation. Developing a concrete solution based on the discriminative constraints is left to future work.

5.3.4 Discussion: Comparison to Existing Methods

Our constraints on unlabeled points (Eq. 5.13) may appear similar to existing works, in particular GRF [113] of the following formulation:

$$F_i = \sum_j W_{ij}/Z_i \cdot F_j, \tag{5.19}$$

where $F_i \in [0, 1]$ is the label function at x_i.

Although they resemble in the surface form, their exact forms are still disparate. We stress that such resemblance—expressing x_i's label as some function of its neighbors x_j—is quite expected, since it is a common insight of graph-based SSL to relate a point and its neighbors on the graph [107, 98]. Nonetheless, our exact function still differs in that PGP normalizes each x_j differently by Z_j and has a damping factor $1-\alpha$, whereas GRF normalizes each x_j by the same Z_i and has no damping factor. Beneath the surface resemblance, there also exist some fundamental differences.

First, most existing cost function [107, 14] or random walk [101, 8, 104] approaches, including GRF, do not correspond to an explicit formulation of pointwise smoothness. For instance, GRF boils down to the energy function of a Gaussian field, which is the aggregated sum of pairwise losses. Such aggregation is not designed for or derived from requiring smoothness at every individual point. Thus, smoothness does not necessarily occur "everywhere." Even though GRF eventually leads to a local weighted average of neighbors (Eq. 5.19), it is a consequence of minimizing an aggregate loss function, rather than originating from the pointwise nature in terms of the two aspects P1 & P2. In contrast, PGP is not derived from an aggregate function, but directly builds on the data closeness model (for P1) and label coupling model (for P2).

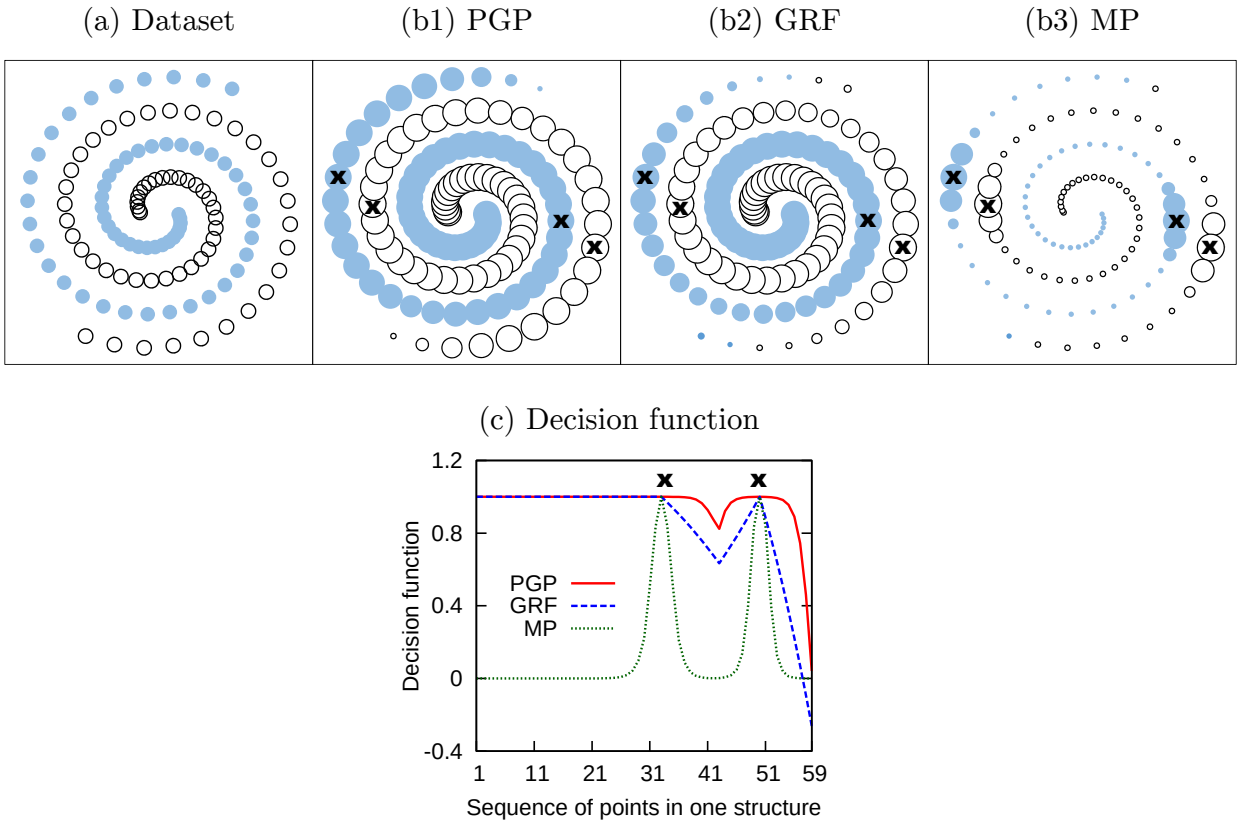


Figure 5.1: Toy problem on the two-spiral dataset. (a) The two-spiral dataset; (b1–3) Visualization of smoothness by different methods; (c) Decision function over a structure.

Second, while many approaches [113, 104] have probabilistic interpretations, they do not directly model $p(Y|X)$ or $p(X|Y)$. In GRF, $\{F_i\}$ represents the most probable configuration of a Gaussian field. F_i is also the probability of hitting a positively labeled node in a random walk starting from x_i. Both interpretations do not explicitly capture $p(Y|X)$ or $p(X|Y)$. In contrast, in PGP, the solution π_y directly corresponds to $p(X|Y=y)$.

Lastly, we use a two-spiral dataset [97] to illustrate that PGP results in better smoothness. The dataset shown in Figure 5.1(a) consists of two spiral structures as the two classes (*i.e.*, $\mathcal{Y} = \{1, 2\}$). We compare the smoothness of PGP with the well-known GRF and the state-of-the-art MP [98], which are graph-based methods with different cost functions.

Smoothness implies that the *decision function* of a classifier should change slowly on a coherent structure [107], *i.e.*, on each spiral in this toy problem. One previously proposed

decision function is $h(x_i) \triangleq (H_{1i} - H_{2i})/(H_{1i} + H_{2i})$, where H_{yi} is x_i's "score" for class y (assigned by each method). The corresponding decision rule is $\text{sign}(h(x_i))$, as employed by all the methods compared here.

In Figure 5.1(b1–3), we visualize the predictions made by the three methods, respectively. All methods use the same four points marked by $\times$ as labeled, whereas the rest are unlabeled. Their respective optimal parameters are adopted. The *color* of a point x_i represents the predicted class y_i, and the *size* of a point x_i represents the magnitude of the decision function at x_i, $|h(x_i)|$. Thus, a smoother decision function shall result in a sequence of points in more uniform sizes over each structure. Clearly, among the three methods, PGP generates points of the most uniform sizes, and is smooth nearly everywhere. Alternatively, we plot the decision function over the sequence of points in one of the structures in Figure 5.1(c), which shows that PGP has a smoother decision function.

With better smoothness, PGP achieves a perfect result against the ideal classification in Figure 5.1(a) (where the size of each point has no significance). In contrast, GRF and MP misclassify 4 and 2 points, respectively.

5.4 Theoretical Analysis

We present an theoretical analysis of PGP, in terms of its error and robustness.

5.4.1 Error in Solution

It is crucial that we can bound the error in the solution estimator $\hat{\pi}_y$, which is estimated from the samples $\mathcal{L}$ and $\mathcal{U}$.

We show that the expected error, $\mathbb{E}\big[\|\hat{\pi}_y - \pi_y\|_1\big]$, can be bounded by two terms, corresponding to the two types of error discussed in Section 5.3.2. Formally, as our solution is the stationary distribution of a Markov chain, the proof can be established based on the perturbation theory of Markov chains [27, 92].

PROPOSITION 5.4 (ERROR ANALYSIS). Given the two constraints (Eq. 5.11 and 5.13), for any constant $\epsilon \in (0, 1)$,

$$\mathbb{E}\left[\left\|\hat{\pi}_y - \pi_y\right\|_1\right] \leq O\left((1-\lambda_1)^{|\mathcal{U}|}\right) + O\left(\exp\left(-2\epsilon^2\lambda_2 \min_{x_i \in \mathcal{L}, p(y|x_i)>0} |\mathcal{L}_{x_i}|\right)\right), \tag{5.20}$$

where $\lambda_1, \lambda_2 \in (0, 1]$ are constants such that

$$\lambda_1 = \min_{x_i \in \mathcal{X}, p(x_i)>0} p\left(x_i\right), \qquad \lambda_2 = \min_{x_i \in \mathcal{L}, p(y|x_i)>0} p\left(y|x_i\right)^2.$$

This result presents two major implications. First, both labeled and unlabeled data can help, as the bound improves when $\mathcal{L}$ or $\mathcal{U}$ grows. Second, the bound is fundamentally limited by $\mathcal{L}$. Given a fixed set of $\mathcal{L}$, even as $|\mathcal{U}| \to \infty$, we can achieve no better than the second error term. In other words, unlabeled data can only help so much. While our analysis is tailored to PGP, the result is consistent with previous analysis [84].

5.4.2 Robustness of Solution

Until now, we have assumed that the graph construction function (Eq. 5.1) is perfect. If the graph were constructed differently (*i.e.*, perturbed), can we assess the robustness of our solution? That is, do small perturbations only cause a small change in the solution? Note that our goal is not to improve the graph construction itself studied elsewhere [26].

In our perturbation model, the pairwise affinity W_{ij} can be perturbed by some scale factor s. We show that the solution derived from the perturbed matrix $\tilde{W}$ does not change much if s is small, based on the perturbation theory of Markov chains [27, 92].

PROPOSITION 5.5 (ROBUSTNESS ANALYSIS). Suppose a matrix $\tilde{W}$ is perturbed from W, such that for some $s > 1$, $W_{ij}/s \leq \tilde{W}_{ij} \leq W_{ij} \cdot s$, $\forall ij$. Let $\tilde{\pi}_y$ be the the solution vector based on $\tilde{W}$. It holds that

$$\left\|\tilde{\pi}_y - \pi_y\right\|_1 \leq O(s^2 - 1). \tag{5.21}$$

Name	Task	Points	Features	Classes	Balanced
Digit1	synthetic digits recognition	1500	241	2	yes
Text	newsgroup classification	1500	11960	2	yes
ISOLET	spoken letters recognition	1200	617	4	yes
Cancer	breast cancer diagnostics	569	30	2	no
USPS	handwritten digits recognition	1500	241	2	no
Yeast	protein sites localization	721	8	4	no

Figure 5.2: Summary of the datasets.

5.5 Experiments

In this section, we empirically evaluate our approach PGP and show that PGP outperforms various existing graph-based SSL algorithms.

5.5.1 Experimental Settings

Datasets. We use six public datasets shown in Figure 5.2. Three of them, Digit1, Text and USPS, come from a benchmark [24]. As the benchmark datasets are mostly balanced, we also use three datasets from UCI repository [9], namely, Yeast, ISOLET and Cancer. Note that Cancer is originally known as "Breast Cancer Wisconsin (Diagnostic)" in the UCI repository. Only a subset of Yeast (classes *cyt*, *me1*, *me2*, *me3*) and of ISOLET (classes a, b, c, d) are used. The benchmark datasets are taken without further processing. For the UCI datasets, feature scaling is performed so that all features have zero mean and unit variance.

Graph. We construct a kNN graph [24], where k is a parameter to be selected. To instantiate Eq. 5.1, we use Euclidean distance for all datasets except Text, and Cosine distance for Text. σ is set to the average distance of all neighboring pairs on the graph.

Labeling. For a given $|\mathcal{L}|$, we sample 200 runs, where in each run $|\mathcal{L}|$ points are randomly chosen as labeled, and the rest are treated as unlabeled. The sampling ensures at least one labeled point for each class. 5% of the runs are reserved for model selection, and the remaining are for testing.

Evaluation. We evaluate the mean performance over the testing runs on each dataset. As classification accuracy is not a truthful measure of the predictive power on imbalanced datasets, we adopt *macro F-score* [44] as the performance metric.

5.5.2 Comparison to Baseline Algorithms

We compare PGP to five state-of-the-art SSL algorithms, which have been shown [113, 14, 98] to significantly outperform earlier ones such as TSVM [63] and SGT [64].

- Gaussian Random Fields (GRF) [113]: a pioneering method based on Gaussian fields, equivalent to optimizing the squared loss.
- LapSVM (LSVM) [14]: an effective graph-based extension of SVM.
- Graph-based Generative SSL (GGS) [53]: a probabilistic generative approach.
- Measure Propagation (MP) [98]: a divergence-based optimization formulation over probability distributions.
- Partially Absorbing Random Walk (PARW) [104]: a random walk method on graphs.

An existing implementation [77] is used for LSVM, whereas our own implementations are used for the others. Each algorithm integrates class priors as in their respective work, if any. Model selection is performed on the reserved runs. For each algorithm, we search $k \in \{5, 10, 15, 20, 25\}$ to construct the kNN graph. GRF and GGS has no other parameters. For LSVM, we search $\gamma_A \in \{1\text{e}{-6}, 1\text{e}{-4}, .01, 1, 100\}$, $r \in \{0, 1\text{e}{-4}, .01, 1, 100, 1\text{e}4, 1\text{e}6\}$. For MP, we search $\alpha \in \{.5, 1, 5, 20, 100\}$, $u \in \{1\text{e}{-8}, 1\text{e}{-6}, 1\text{e}{-4}, .01, .1, 1, 10\}$, as well as $v \in \{1\text{e}{-8}, 1\text{e}{-6}, 1\text{e}{-4}, .01, .1\}$. For PARW, we search $\alpha \in \{1\text{e}{-8}, 1\text{e}{-6}, 1\text{e}{-4}, .01, 1, 100\}$. Lastly, for our method PGP, we search $\alpha \in \{.01, .02, .05, .1, .2, .5\}$.

The mean macro F-scores on the testing runs are reported in Figure 5.3, leading to three major findings. First, PGP performs the *best* or *not significantly different* from the best in 15 out of the 18 cases (*i.e.*, columns), whereas GRF, LSVM, GGS, MP and PARW perform

(a) Number of labeled points $|\mathcal{L}| = 10$

	Digit1	Text	ISOLET	Cancer	USPS	Yeast
GRF	.894	.451	.627	.871	.638	.510
LSVM	.833	.428	.719	.886	**.698**	.562
GGS	.855	.567	.677	.867	.666	.540
MP	**.901**	.558	.692	.898	**.713**	.574
PARW	.881	**.587**	.721	.893	**.706**	.575
PGP	**.910**	**.592**	**.734**	**.910**	**.704**	**.593**

(b) Number of labeled points $|\mathcal{L}| = 20$

	Digit1	Text	ISOLET	Cancer	USPS	Yeast
GRF	**.932**	.467	.686	.913	.682	.569
LSVM	**.935**	.472	**.780**	.914	.780	.614
GGS	.886	**.648**	.771	.918	.758	.578
MP	**.940**	.611	.735	**.924**	**.794**	.617
PARW	.923	**.640**	**.782**	.920	**.791**	.613
PGP	**.939**	.634	**.786**	**.927**	**.796**	**.633**

(c) Number of labeled points $|\mathcal{L}| = 150$

	Digit1	Text	ISOLET	Cancer	USPS	Yeast
GRF	**.979**	.744	.849	**.958**	.902	.718
LSVM	**.979**	**.771**	.901	.956	.908	**.729**
GGS	.965	**.771**	**.905**	.948	.906	**.727**
MP	**.979**	.746	.854	**.957**	.913	.718
PARW	.975	.729	.897	.955	.916	.715
PGP	**.978**	.732	**.902**	**.958**	**.931**	.721

Figure 5.3: Performance comparison of SSL methods. In each column, the *best* result and those *not significantly different* ($p > .05$ in t-test) are bolded.

as such in only 3, 6, 4, 7, 5 cases, respectively. Second, while PGP has relatively stable performance across all the cases, the baselines can be volatile. In particular, when PGP is not the best, there is no consistent best method, which varies between LSVM, GGS and MP. Third, PGP is especially advantageous with limited labeled data (*e.g.*, $|\mathcal{L}| = 10$), which is the very motivation of SSL. In contrast, when abundant data are labeled (*e.g.*, $|\mathcal{L}| = 150$), all of the algorithms perform better, and thus not surprisingly, the margin between them becomes smaller.

5.5.3 Integrating Class Priors

A concrete benefit of probabilistic modeling is to enable better integration of class priors, which is also probabilistic in nature. We demonstrate that principled integration of class priors is more effective than heuristics, and integrating more accurate priors helps.

Integration of priors. We compare two different methods of integrating class priors:

- BAYES: integrating in PGP in a Bayesian way (Eq. 5.15);
- CMN: integrating in GRF using the popular heuristic Class Mass Normalization [113].

Note that BAYES and CMN respectively apply to a different algorithm as they are originally intended for, and they are not designed for the other algorithm. The priors are approximated in the same way for both methods, using the labeled points with add-one smoothing.

We study the corrective power of each method: integrating priors can be seen as "corrections" to the *base model* that does not incorporate priors. Directly assessing the improvement over the base model is unfair, since the base performances of PGP and GRF differ. Instead, we compute the F-score from the precision and recall of the corrections:

$$\text{precision} = \frac{\#\text{ true corrections}}{\#\text{ corrections made}}; \qquad \text{recall} = \frac{\#\text{ true corrections}}{\#\text{ corrections needed}}. \tag{5.22}$$

The results are presented in Figure 5.4(a) on the imbalanced datasets, which are more interesting given their non-uniform class priors. In all but one case, BAYES possesses much better corrective power than CMN.

More accurate priors. Using more accurate priors is expected to improve the performance. Suppose we know the exact priors by considering the labels of all points. We then apply the approximate and exact priors to PGP. We directly measure the performance with or without priors, given the same base model. The results are presented in Figure 5.4(b), which illustrate that, while the estimated priors are effective in many cases, the supposedly more accurate exact priors are able to further improve the performance.

(a) Corrective power of different prior integration

	$\|\mathcal{L}\| = 10$			$\|\mathcal{L}\| = 20$			$\|\mathcal{L}\| = 150$		
	Cancer	USPS	Yeast	Cancer	USPS	Yeast	Cancer	USPS	Yeast
CMN	**.449**	.264	.310	.255	.275	.307	.087	.250	.084
BAYES	.333	**.564**	**.388**	**.373**	**.692**	**.504**	**.475**	**.781**	**.607**

(b) Accuracy of PGP using different estimations of priors on

	$\|\mathcal{L}\| = 10$			$\|\mathcal{L}\| = 20$			$\|\mathcal{L}\| = 150$		
	Cancer	USPS	Yeast	Cancer	USPS	Yeast	Cancer	USPS	Yeast
None	.923	.636	.568	.927	.722	.600	.950	.873	.678
Approx.	.910	.704	.593	.927	.796	.633	.958	.931	.721
Exact	**.929**	**.733**	**.622**	**.937**	**.805**	**.647**	**.960**	**.932**	**.730**

Figure 5.4: Effect of incorporating class priors in prediction.

5.5.4 Validating Theoretical Analysis

We empirically validate the analysis in Section 5.4. First, while the expected effect of increasing labeled points is shown in Figure 5.3, our findings on unlabeled data also match the error bound analysis. That is, more unlabeled data reduces the error, but the rate of reduction decreases as unlabeled data are further increased. Second, the findings also support our robustness conclusion. That is, small perturbations in W only cause small changes in the solution π_y.

5.5.4.1 Effect of unlabeled data

We investigate unlabeled points here to validate the error analysis in Proposition 5.4. For each dataset, we sample $|\mathcal{L}| + |\mathcal{U}|$ data points, where we fix $|\mathcal{L}| = 10$ while $|\mathcal{U}|$ is varied from Δ to 5Δ. Δ is different for different datasets in order to visualize the performance trend clearer. Among these points, we draw 10 points as labeled $\mathcal{L}$, and treat the others as unlabeled $\mathcal{U}$.

We present the mean performance over 10 such samples in Figure 5.5. The empirical results are consistent with our analysis. As expected, the performance improves when we use more unlabeled data even though the labeled data remain the same. The result also

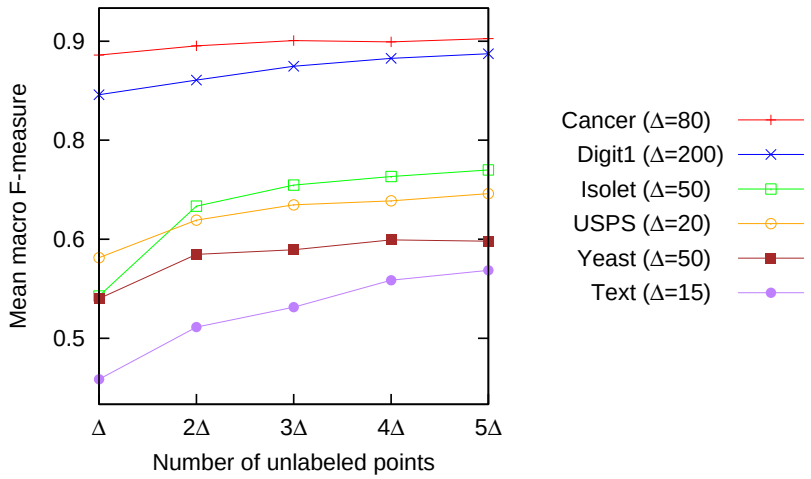


Figure 5.5: Effect of unlabeled data.

shows that the rate of improvement slows down as we introduce more unlabeled data, eventually hitting a ceiling. This means growing unlabeled data alone cannot reduce the error indefinitely, since it is also limited by labeled data as explained by the second error term in our analysis. In more intuitive words, unlabeled data can only help so much.

5.5.4.2 Effect of graph perturbation

Lastly, we study the robustness of our solution against graph perturbation. As Proposition 5.5 has established, the change in the solution π_y can be bounded by $O(s^2 - 1)$, where $s \geq 1$ is the degree of perturbation. Given s, we independently generate a random $\tilde{W}_{ij} \in [W_{ij}/s, W_{ij} \cdot s]$, $\forall ij$, and use $\tilde{W}$ as the perturbed affinity matrix.

We illustrate the effect of perturbation on USPS with $|\mathcal{L}| = 10$. In Figure 5.6, when the graph is perturbed to various degrees ($s \in \{1.01, 1.02, \ldots, 1.1\}$), the change in π_y (averaged over the testing runs with standard deviation bars) for both $y = 1$ and $y = 2$ is linear in s. This result is better than (yet still consistent with) the theoretical bound $O(s^2 - 1)$. The reason is that we consider the worst case scenario in deriving the bound, but in our experiments here the perturbations are generated randomly within the degree s. Furthermore, we observe that the change in the actual predictive power in terms of macro

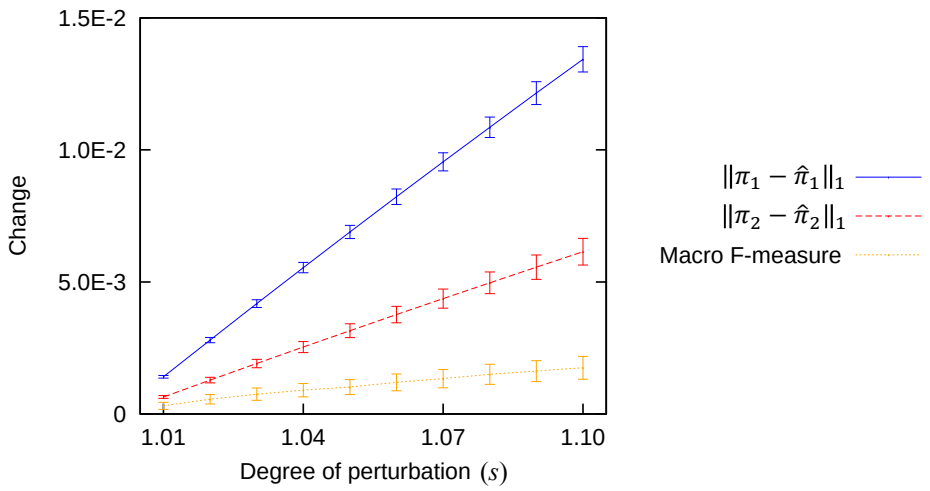


Figure 5.6: Effect of graph perturbation.

F-measure is highly correlated with the change in π_y. The same trend is observed on all datasets with different $|\mathcal{L}|$'s. Thus, our solution is robust—small degrees of perturbation cause small changes in π_y and the predictive power.

5.6 Conclusion

We proposed a novel framework of Probabilistic Graph-based Pointwise Smoothness (PGP), hinging on the foundational data closeness and label coupling models. Our framework can accommodate both generative and discriminative formulations, unifying two different "modes" of smoothness which correspond to forward and backward random walks on the graph. We focused on the generative formulation and developed a set of probability constraints over $p(X|Y)$. Solving these constraints ultimately enables us to infer $p(Y|X)$ for class prediction. We also studied the theoretical properties of PGP in terms of its generalization error and robustness. Finally, we empirically demonstrated that PGP is superior to existing state-of-the-art baselines.

Chapter 6

Conclusion: Unification and Beyond

In this chapter, we first highlight the significance of this dissertation, and then discuss a few future research directions.

6.1 Conclusion: Unifying Framework

Graph representations of data enable many crucial tasks, including the extraction, ranking and classification of objects addressed in earlier chapters. Despite of the widely varying problem formulations, they all boil down to a measure of similarity between nodes on the graph. As their common insight, *similar objects (or neighboring nodes in graph terminology) should behave similarly.* To summarize, in pattern-based relation extraction, similar tuples could be extracted by a similar set of patterns; in networked object ranking, similar objects should appear near each other in the ranked list; in graph-based classification, similar objects would be more likely to have the same label.

To realize this insight, a great number of graph-based techniques have been proposed. As we have explored in this dissertation, one popular and effective family of approaches are based on random walks. Even though they are fundamentally built upon the same insight, many different forms and variations of random walk exist, including the forward and backward walks. Given these varying forms of random walk, their differences and similarities are not thoroughly investigated in the existing literature. Apart from using them to solve different graph-based searching and mining problems, which we also contributed to in the first part of this dissertation (Chapters 2–4), little understanding has been developed regarding how

they are unified by the same underlying insight, and what makes them different within the unifying framework. Such understanding is crucial for two reasons. First, it helps us identify the "right" forms of random walk in different scenarios. Second, a unifying framework encourages the development of new solutions for novel purposes in a systematic and principled manner.

Thus, in the second part of this dissertation (Chapter 5), we attempted to investigate the fundamental principle governing two symmetric forms of random walk: the forward and backward random walks, focused on a framework unifying them as well as their differences within the framework. In particular, we identified the core mechanism of random walks as *smoothness* on the graph, *i.e.*, the principle governing the common insight of graph-based searching and mining—the behavior of two close nodes on the graph tends to be the same—as we just remarked earlier.

Subsequently, we showed that the forward and backward random walks simply represent two different "modes" of smoothness under one framework. In particular, the forward random walk correspond to the generative modeling, whereas the backward walk correspond to the discriminative modeling, both under the same smoothness framework. In other words, the former captures the likelihood and the latter captures posterior probabilities, respectively. Thus, it becomes natural that the backward random walk is appropriate for object classification in Chapter 4. Furthermore, probabilistic precision and recall in Chapter 2 can also be interpreted as the posterior distribution and likelihood, and thus correspond to the discriminative and generative models of smoothness, respectively. Let us consider a pattern. On the one hand, given a tuple extracted by the pattern (observable evidence), the probability that it is a relevant tuple (latent variable) is our definition of probabilistic precision. On the other hand, given a relevant tuple, the probability that it is extracted by the pattern is probabilistic recall. While the notions of specificity and importance in Chapter 3 seem to intuitively parallel precision and recall, and can be respectively represented by backward and forward random walks, their direct correspondence to the two modes of smoothness still

remains open for future work. Nevertheless, our attempt at the unification sheds light on the "right" form of random walk for different scenarios.

The unifying smoothness framework also provides avenue to extend to or develop new solutions systematically. While our framework can adopt either discriminative or generative models, so far they are only explored on individual nodes. Note that this is not a restriction of the framework; potentially, we can model the joint probability of a set of nodes under the same framework, which may be useful for novel scenarios (for example, we may have additional constraints on a set of user nodes belonging to the same community in a social network). In general, the smoothness framework is a foundation upon which different probabilistic formulations can be adopted.

6.2 Future Agenda

While this dissertation studied various graph-based searching and mining problems and their underlying principle, there are some promising future directions to explore.

Strengthening the unification. Even though our smoothness framework has unified the forward and backward random walks, it would be interesting to extend or adapt the framework to also incorporate other forms of random walk or graph-based techniques in general. It is also important to identify and interpret the semantics conveyed by different solutions, parallel to what we have studied in this dissertation (for instance, the concepts of probabilistic precision and recall, as well as specificity and importance).

Collective and structured measures. In this dissertation, we only considered different graph-based measures for individual objects, whether in extraction, ranking, or classification. However, it is sometimes useful to measure the quality of a set of nodes. For instance, what are the best K patterns *in conjunction* for extracting a given relation? Note that the best K patterns often differ from the K best patterns, due to the "overlaps" between

different patterns. Hence, it is crucial that we can define a "collective" measure for a set of objects which need to be assessed together. The collective measure can be further extended to become a "structured" measure, in order to measure a substructure on the graph. For instance, a community on a social network is a substructure with internal connections, which is more complex than a "flat" collection of objects.

Learning graph structure. We always assumed the full knowledge of a graph in this dissertation. However, for many real-world graphs such as social networks, this is often untrue due to various reasons. The missing information could be edges in the graph. For example, public access to social connections between users may be blocked for privacy considerations. Node attributes could also be missing or noisy. For instance, users may not supply their hometowns in their profiles, or fill in uninformative values such as "earth" or "home." Thus, it is desirable if we can learn the missing information on a graph, in particular through a mutually reinforcing mechanism—inferring more edges can improve the inference of node attributes, and vice versa. A more complete graph would ultimately enable us to improve the effectiveness of searching and mining on this graph.

References

[1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003. (Cited on page 1, 7, 48, 70)

[2] G. Agarwal, G. Kabra, and K. C.-C. Chang. Towards rich query interpretation: walking back and forth for mining query templates. In *Proceedings of the International World Wide Web Conference*, pages 1–10, 2010. (Cited on page 6, 16, 24, 28, 29, 48, 67, 73)

[3] E. Agichtein. Confidence estimation methods for partially supervised information extraction. In *Proceedings of the SIAM International Conference on Data Mining*, 2006. (Cited on page 14, 16)

[4] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of the ACM Conference on Digital Libraries*, pages 85–94, 2000. (Cited on page 4, 7, 12, 14, 15, 16, 19, 36)

[5] E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. In *Proceedings of the International Conference on Data Engineering*, pages 113–124, 2003. (Cited on page 12, 15, 36)

[6] Amazon. `http://www.amazon.com/`. Accessed: 2013-11-08. (Cited on page 5, 95)

[7] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the International Conferences on Very Large Data Bases*, pages 586–597, 2002. (Cited on page 7)

[8] A. Azran. The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. In *Proceedings of the International Conference on Machine Learning*, pages 49–56, 2007. (Cited on page 113, 116, 125)

[9] K. Bache and M. Lichman. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. `http://archive.ics.uci.edu/ml`, 2013. Accessed: 2013-11-08. (Cited on page 129)

[10] A. Balmin, V. Hristidis, and Y. Papakonstantinou. ObjectRank: Authority-based keyword search in databases. In *Proceedings of the International Conferences on Very Large Data Bases*, pages 564–575, 2004. (Cited on page 6, 7, 43, 44, 48, 49)

[11] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007. (Cited on page 16)

[12] S. M. Beitzel, E. C. Jensen, O. Frieder, D. D. Lewis, A. Chowdhury, and A. Kolcz. Improving automatic query classification via semi-supervised learning. In *Proceedings of the International Conference on Data Mining*, pages 42–49, 2005. (Cited on page 83)

[13] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proceedings of the Annual Conference on Learning Theory*, pages 624–638, 2004. (Cited on page 80, 83, 88, 89)

[14] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006. (Cited on page 9, 80, 83, 88, 89, 112, 113, 125, 130)

[15] P. Berkhin. Bookmark-coloring approach to personalized pagerank computing. *Internet Mathematics*, 3(1):41–62, 2006. (Cited on page 61, 62, 152)

[16] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the International Conference on Machine Learning*, pages 19–26, 2001. (Cited on page 112)

[17] P. Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, pages 1170–1182, 1987. (Cited on page 2)

[18] S. Brin. Extracting patterns and relations from the world wide web. In *Proceedings of the International Workshop on the Web and Databases*, pages 172–183, 1998. (Cited on page 4, 7, 12, 14, 15, 16, 19, 27)

[19] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. (Cited on page 1)

[20] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 231–238, 2007. (Cited on page 82)

[21] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 3–10, 2009. (Cited on page 82)

[22] D. Carmel and E. Yom-Tov. Estimating the query difficulty for information retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–89, 2010. (Cited on page 96)

[23] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *Proceedings of the International World Wide Web Conference*, pages 571–580, 2007. (Cited on page 6, 7, 49, 75)

[24] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-supervised learning*. MIT Press, Cambridge, MA, 2006. (Cited on page 8, 112, 119, 129)

[25] J. Chen, H.-R. Fang, and Y. Saad. Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10:1989–2012, 2009. (Cited on page 123)

[26] H. Cheng, Z. Liu, and J. Yang. Sparsity induced similarity measure for label propagation. In *Proceedings of the International Conference on Computer Vision*, pages 317–324, 2009. (Cited on page 128)

[27] G. E. Cho and C. D. Meyer. Comparison of perturbation bounds for the stationary distribution of a Markov chain. *Linear Algebra and its Applications*, 335(1):137–150, 2001. (Cited on page 127, 128, 159, 160)

[28] C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 358–365, 2001. (Cited on page 12)

[29] N. Craswell and M. Szummer. Random walks on the click graph. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 239–246, 2007. (Cited on page 48, 80, 83, 91)

[30] D. Das and N. A. Smith. Graph-based lexicon expansion with sparsity-inducing penalties. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 677–687, 2012. (Cited on page 113)

[31] DBLP. http://www.informatik.uni-trier.de/~ley/db/. Accessed: 2013-11-08. (Cited on page 3, 67)

[32] F. Diaz. Regularizing query-based retrieval scores. *Information Retrieval*, 10(6):531–562, 2007. (Cited on page 83)

[33] F. D. Diaz. Improving relevance feedback in language modeling with score regularization. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 807–808, 2008. (Cited on page 83)

[34] S. T. Dumais, M. Banko, E. Brill, J. J. Lin, and A. Y. Ng. Web question answering: is more always better? In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 291–298, 2002. (Cited on page 12)

[35] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowItAll: (preliminary results). In *Proceedings of the International World Wide Web Conference*, pages 100–110, 2004. (Cited on page 12, 15, 16)

[36] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 391–398, 2004. (Cited on page 16)

[37] Y. Fang and K. C.-C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 825–834, 2011. (Cited on page 9, 48, 73)

[38] Y. Fang, K. C.-C. Chang, and H. W. Lauw. RoundTripRank: Graph-based proximity with importance and specificity. In *Proceedings of the International Conference on Data Engineering*, pages 613–624, 2013. (Cited on page 9)

[39] Y. Fang, K. C.-C. Chang, and H. W. Lauw. Graph-based semi-supervised learning: Realizing pointwise smoothness probabilistically. In *Proceedings of the International Conference on Machine Learning*, pages 406–414, 2014. (Cited on page 10)

[40] Y. Fang and M.-W. Chang. Entity linking on microblogs with spatial and temporal signals. *Transactions of the Association for Computational Linguistics*, 2014. *To appear*. (Cited on page 1)

[41] Y. Fang, B.-J. P. Hsu, and K. C.-C. Chang. Confidence-aware graph regularization with heterogeneous pairwise features. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 951–960, 2012. (Cited on page 10)

[42] J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 363–370, 2005. (Cited on page 4)

[43] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005. (Cited on page 50)

[44] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003. (Cited on page 130)

[45] S. Fortunato. Community detection in graphs. *Computing Research Repository*, abs/0906.0612, 2009. (Cited on page 8)

[46] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1979. (Cited on page 2)

[47] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 7(2):3–12, 2005. (Cited on page 8)

[48] G. Getz, N. Shental, and E. Domany. Semi-supervised learning–a statistical physics approach. In *ICML Workshop on Learning with Partially Classified Training Data*, 2005. (Cited on page 113)

[49] C. L. Giles. The future of CiteSeer: Citeseer^x. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, page 2, 2006. (Cited on page 1, 67)

[50] O. Goldreich. *A primer on pseudorandom generators*, volume 55. American Mathematical Society, 2010. (Cited on page 114, 118)

[51] M. S. Gupta, A. Pathak, and S. Chakrabarti. Fast algorithms for top-k personalized pagerank queries. In *Proceedings of the International World Wide Web Conference*, pages 1225–1226, 2008. (Cited on page 49, 61, 63, 75)

[52] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003. (Cited on page 54, 55, 154)

[53] J. He, J. G. Carbonell, and Y. Liu. Graph-based semi-supervised learning as a generative model. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2492–2497, 2007. (Cited on page 113, 130)

[54] X. He and P. Jhala. Regularized query classification using search click information. *Pattern Recognition*, 41(7):2283–2288, 2008. (Cited on page 83, 97)

[55] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the International Conference on Computational Linguistics*, pages 539–545, 1992. (Cited on page 12)

[56] V. Hristidis, H. Hwang, and Y. Papakonstantinou. Authority-based keyword search in databases. *ACM Transactions on Database Systems*, 33(1), 2008. (Cited on page 7, 44, 48, 72, 73)

[57] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23, 2000. (Cited on page 8)

[58] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 538–543, 2002. (Cited on page 7, 48, 70)

[59] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the International World Wide Web Conference*, pages 271–279, 2003. (Cited on page 43, 48, 49, 52)

[60] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (1)*, pages 570–586, 2010. (Cited on page 7)

[61] M. Ji, J. Yan, S. Gu, J. Han, X. He, W. V. Zhang, and Z. Chen. Learning search tasks in queries and web pages via graph regularization. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 55–64, 2011. (Cited on page 7, 80, 83)

[62] W. Jiang, J. Vaidya, Z. Balaporia, C. Clifton, and B. Banich. Knowledge discovery from transportation network data. In *Proceedings of the International Conference on Data Engineering*, pages 1061–1072, 2005. (Cited on page 1)

[63] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 200–209, 1999. (Cited on page 130)

[64] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning*, pages 290–297, 2003. (Cited on page 130)

[65] R. Johnson and T. Zhang. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288, 2008. (Cited on page 112)

[66] R. D. King, S. H. Muggleton, A. Srinivasan, and M. Sternberg. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93(1):438–442, 1996. (Cited on page 8)

[67] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998. (Cited on page 2)

[68] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 245–255, 2006. (Cited on page 7, 48)

[69] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *Proceedings of the International World Wide Web Conference*, pages 150–161, 2001. (Cited on page 12)

[70] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001. (Cited on page 4)

[71] J. D. Lafferty and L. A. Wasserman. Statistical analysis of semi-supervised regression. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 801–808, 2007. (Cited on page 9, 112)

[72] J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 177–187, 2005. (Cited on page 65)

[73] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 339–346, 2008. (Cited on page 80, 83, 99)

[74] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 572–579, 2009. (Cited on page 83)

[75] D. Liben-Nowell and J. M. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007. (Cited on page 8)

[76] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993. (Cited on page 3, 114)

[77] S. Melacci and M. Belkin. Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research*, 12:1149–1184, 2011. (Cited on page 130)

[78] Microsoft. Bing shopping. `http://www.bing.com/shopping/`. Accessed: 2011-06-01. (Cited on page 95, 100)

[79] R. Motwani and P. Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010. (Cited on page 154)

[80] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: bringing order to web objects. In *Proceedings of the International World Wide Web Conference*, pages 567–574, 2005. (Cited on page 6, 7, 43, 48)

[81] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report SIDL-WP-1999-0120, Stanford University, 1999. (Cited on page 2, 7, 43, 48, 49, 54)

[82] W. H. Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge University Press, 2007. (Cited on page 92)

[83] D. Ravichandran and E. H. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 41–47, 2002. (Cited on page 16)

[84] P. Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research*, 8:1369–1392, 2007. (Cited on page 9, 112, 113, 128)

[85] M. Rosenberg. Capitals of every independent country. http://geography.about.com/od/countryinformation/a/capitals.htm. Accessed: 2013-11-08. (Cited on page 36)

[86] J.-F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062):1173–1178, 2005. (Cited on page 1)

[87] K. Salem and H. Garcia-Molina. Disk striping. In *Proceedings of the International Conference on Data Engineering*, pages 336–342, 1986. (Cited on page 66)

[88] P. Sarkar and A. W. Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 335–343, 2007. (Cited on page 48, 72, 73)

[89] P. Sarkar and A. W. Moore. Fast nearest-neighbor search in disk-resident graphs. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 513–522, 2010. (Cited on page 61, 64, 75)

[90] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In *Proceedings of the International Conference on Machine Learning*, pages 896–903, 2008. (Cited on page 49, 58, 64, 67, 69, 72)

[91] S. Sekine. On-demand information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2006. (Cited on page 16)

[92] E. Seneta. Sensitivity of finite markov chains under perturbation. *Statistics & probability letters*, 17(2):163–168, 1993. (Cited on page 127, 128, 159, 160)

[93] D. Shen, Y. Li, X. Li, and D. Zhou. Product query classification. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 741–750, 2009. (Cited on page 82, 83)

[94] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 131–138, 2006. (Cited on page 82)

[95] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, 2006. (Cited on page 16)

[96] A. Singh, R. D. Nowak, and X. Zhu. Unlabeled data: Now it helps, now it doesn't. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 1513–1520, 2008. (Cited on page 9, 112)

[97] S. Singh. 2D spiral pattern recognition with possibilistic measures. *Pattern Recognition Letters*, 19(2):141–147, 1998. (Cited on page 126)

[98] A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12:3311–3370, 2011. (Cited on page 9, 112, 113, 125, 126, 130)

[99] Y. Sun, J. Han, J. Gao, and Y. Yu. itopicmodel: Information network-integrated topic modeling. In *Proceedings of the International Conference on Data Mining*, pages 493–502, 2009. (Cited on page 68)

[100] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011. (Cited on page 7, 68)

[101] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 945–952, 2001. (Cited on page 83, 91, 116, 125)

[102] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 747–756, 2007. (Cited on page 7, 48)

[103] R. C. Wang and W. W. Cohen. Language-independent set expansion of named entities using the web. In *Proceedings of the International Conference on Data Mining*, pages 342–350, 2007. (Cited on page 4)

[104] X.-M. Wu, Z. Li, A. M.-C. So, J. Wright, and S.-F. Chang. Learning with partially absorbing random walks. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 3086–3094, 2012. (Cited on page 116, 125, 126, 130)

[105] Yahoo. Yahoo! BOSS search API. `http://developer.yahoo.com/search/boss/`. Accessed: 2010-07-01. (Cited on page 35)

[106] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the Web. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 1048–1052, 2007. (Cited on page 16)

[107] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 321–328, 2003. (Cited on page 5, 9, 80, 81, 83, 88, 89, 112, 113, 125, 126)

[108] D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 1633–1640, 2004. (Cited on page 97)

[109] M. Zhou, T. Cheng, and K. C.-C. Chang. Data-oriented content query system: searching for data into text on the web. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 121–130, 2010. (Cited on page 12, 13, 15, 39)

[110] F. Zhu, Y. Fang, K. C.-C. Chang, and J. Ying. Incremental and accuracy-aware personalized pagerank through scheduled approximation. *Proceedings of the VLDB Endowment*, 6(6):481–492, 2013. (Cited on page 49)

[111] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin-Madison, 2005. (Cited on page 8, 112)

[112] X. Zhu and Z. Ghahramani. Towards semi-supervised classification with markov random fields. Technical Report CMU-CALD-02-106, School of Computer Science, Carnegie Mellon University, 2002. (Cited on page 113)

[113] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, pages 912–919, 2003. (Cited on page 5, 9, 80, 81, 82, 83, 88, 89, 112, 113, 125, 126, 130, 132)

Appendix A Proofs of Propositions

Proof of Proposition 2.1 (Equivalence Sampling)

We first establish the result for precision.

$$\begin{aligned}
\sum_{t_i \in \tau(p_j)} \mathcal{P}(t_i) \frac{W_{ij}}{|\tau(p_j)|} &\overset{1}{=} \sum_{k:\tau(p_j)\cap T_k \neq \emptyset} \left(\sum_{t_i \in T_k} \mathcal{P}(t_i) \frac{W_{ij}}{|\tau(p_j)|} \right) \\
&\overset{2}{=} \sum_{k:\tau(p_j)\cap T_k \neq \emptyset} \left(\rho \sum_{t_i \in T_k} \mathcal{P}(t_i) \frac{W_{ij}}{\rho|\tau(p_j)|} \right) \\
&\overset{3}{=} \sum_{k:\tau(p_j)\cap T_k \neq \emptyset} \left(\rho \sum_{t_i \in T_k} \mathcal{P}(t_i) \frac{W_{ij}}{|\hat{\tau}(p_j)|} \right) \\
&\overset{4}{=} \sum_{k:\tau(p_j)\cap T_k \neq \emptyset} \left(\rho|T_k| \mathcal{P}(t_i) \frac{W_{ij}}{|\hat{\tau}(p_j)|} \right) \\
&\overset{5}{=} \sum_{k:\tau(p_j)\cap T_k \neq \emptyset} \left(|\hat{T}_k| \mathcal{P}(t_i) \frac{W_{ij}}{|\hat{\tau}(p_j)|} \right) \\
&\overset{6}{=} \sum_{k:\tau(p_j)\cap \hat{T}_k \neq \emptyset} \left(\sum_{t_i \in \hat{T}_k} \mathcal{P}(t_i) \frac{W_{ij}}{|\hat{\tau}(p_j)|} \right) \\
&\overset{7}{=} \sum_{t_i \in \tau(p)\cap \hat{T}} \mathcal{P}(t_i) \frac{W_{ij}}{|\hat{\tau}(p_j)|}. \qquad \text{(A.1)}
\end{aligned}$$

Step 1 partitions the tuples in $\tau(p_j)$ into equivalent classes T_k. Step 2 rewrites the equation to introduce ρ. In step 3, we have $\rho|\tau(p_j)| = |\hat{\tau}(p_j)|$ due to Eq. 2.13. Step 4 follows since $\mathcal{P}(t_i)$ and W_{ij} are the same for tuples in the same equivalence class (Eq. 2.13 and 2.14), where t_i is any tuple in T_k. Step 5 is due to $|\hat{T}_k| = \rho|T_k|$. Step 6 rewrites the multiplication into a summation over $\hat{T}_k$. Step 7 combines the sampled equivalence classes into $\hat{T}$.

The proof for recall can be derived quite similarly. □

Proof of Proposition 3.2 (Linearity of RoundTripRank)

$$
\begin{aligned}
r(Q,v) &\triangleq p\left(W_L = v | W_0 = W_{L+L'}, W_0 \in Q\right) \\
&\overset{1}{=} \sum_{q \in Q} p\left(W_L = v, W_0 = q | W_0 = W_{L+L'}, W_0 \in Q\right) \\
&\overset{2}{=} \sum_{q \in Q} p\left(W_L = v | W_0 = q, W_0 = W_{L+L'}, W_0 \in Q\right) p\left(W_0 = q | W_0 = W_{L+L'}, W_0 \in Q\right) \\
&\overset{3}{=} \sum_{q \in Q} p\left(W_L = v | W_0 = q, W_0 = W_{L+L'}\right) p\left(W_0 = q | W_0 = W_{L+L'}, W_0 \in Q\right) \\
&\overset{4}{=} \sum_{q \in Q} r(q,v) \lambda(q)
\end{aligned} \tag{A.2}
$$

Step 1 expands to the joint distribution with W_0 for each $q \in Q$. In step 2, we apply Bayes' rule. In step 3, $W_0 \in Q$ is removed from the condition given that $W_0 = q$. □

Proof of Proposition 3.3 (Decomposition of RoundTripRank)

$$
\begin{aligned}
r(q,v) &\triangleq p\left(W_L = v | W_0 = W_{L+L'}, W_0 = q\right) \\
&\overset{1}{=} p\left(W_L = v, W_0 = W_{L+L'} | W_0 = q\right) / p\left(W_0 = W_{L+L'} | W_0 = q\right) \\
&\overset{2}{\propto} p\left(W_L = v, W_{L+L'} = q | W_0 = q\right) \\
&\overset{3}{=} p\left(W_{L+L'} = q | W_L = v, W_0 = q\right) p\left(W_L = v | W_0 = q\right) \\
&\overset{4}{=} p\left(W_{L+L'} = q | W_L = v\right) p\left(W_L = v | W_0 = q\right) \\
&\overset{5}{=} p\left(W_{L'} = q | W_0 = v\right) p\left(W_L = v | W_0 = q\right)
\end{aligned} \tag{A.3}
$$

Step 1 applies Bayes' rule. In step 2, the v-independent denominator is dropped without altering ranking. Step 3 is another application of Bayes' rule. In step 4, given W_L, $W_{L+L'}$ is conditionally independent of W_0 in a Markovian walk. In step 5, a shift in time by $-L$ does not affect the transition probability. □

Proof of Proposition 3.4 (Decomposition of RoundTripRank+)

As each surfer walks independently, we can factor $r_\Omega(q, v)$ into the product of individual surfers. Then, for each surfer, apply Proposition 3.3. Since some surfers shortcut the round trip, *e.g.*, $p\left(W^\omega_{L+L'} = q | W^\omega_L = v\right) = 1$ for $\omega \in \Omega_{10}$, and similarly for $\omega \in \Omega_{01}$, we will get

$$r_\Omega(q, v) \propto f(q, v)^{|\Omega_{11}|+|\Omega_{10}|} \cdot b(q, v)^{|\Omega_{11}|+|\Omega_{01}|}. \tag{A.4}$$

We can further normalize the exponents in Eq. A.4 without altering ranking, since the power function is monotonic:

$$\begin{aligned} r_\Omega(q, v) &\propto \left(f(q, v)^{|\Omega_{11}|+|\Omega_{10}|} \cdot b(q, v)^{|\Omega_{11}|+|\Omega_{01}|}\right)^{\frac{1}{|\Omega|+|\Omega_{11}|}} \\ &= f(q, v)^{\frac{|\Omega_{11}|+|\Omega_{10}|}{|\Omega|+|\Omega_{11}|}} \cdot b(q, v)^{\frac{|\Omega_{11}|+|\Omega_{01}|}{|\Omega|+|\Omega_{11}|}} \\ &= f(q, v)^{1-\beta} \cdot b(q, v)^{\beta}. \end{aligned} \tag{A.5}$$

The above result concludes the proof. □

Proof of Proposition 3.5 (Bounds of F-Rank)

It is already known that Eq. 3.23 holds [15]. Thus, we only need to prove Eq. 3.22, upon which Eq. 3.24 can also be built. For any node v, consider an upper bound U for the sum of residual that is already at v or may spread to v for the first time. An α portion of U adds to $\rho(q, v)$, while the remaining $(1 - \alpha)$ spreads to v's out-neighbors. Then, the same mechanism repeats—residual spread to neighbors may come back to v for a second or third time, adding α portion to $\rho(q, v)$ each time, totaling to

$$U \cdot [\alpha + (1 - \alpha)^2\alpha + (1 - \alpha)^4\alpha + \ldots] = U/(2 - \alpha). \tag{A.6}$$

Thus, $U/(2 - \alpha)$ is the maximum residual that may add to $\rho(q, v)$ for any node v, *i.e.*,

an unseen upper bound. It can be further transformed to become Eq. 3.22, since

$$\begin{aligned} U &\leq \mu(q,v) + (1-\alpha)\sum_{u \neq v} \mu(q,u) \\ &= \mu(q,v) + (1-\alpha)\sum_{u \in V} \mu(q,u) - (1-\alpha)\mu(q,v) \\ &= \alpha\mu(q,v) + (1-\alpha)\sum_{u \in V} \mu(q,u) \\ &\leq \alpha \max_{u \in V} \mu(q,u) + (1-\alpha)\sum_{u \in V} \mu(q,u). \end{aligned} \tag{A.7}$$

Note that in the last step, we apply $\mu(q,v) \leq \max_{u \in V} \mu(q,u)$ to make the upper bound independent of v.

Finally, Eq. 3.24 can be obtained by adding a seen node v's current $\rho(q,v)$ to $\hat{f}^{(0)}(q)$. □

Proof of Proposition 4.1 (Convergence of Local Errors)

Let Λ_0 denote that all parameters λ_τ are zero. Then,

$$\min_{\Lambda} L_{\text{err}}^{(t)}(O_L, \Lambda) \leq L_{\text{err}}^{(t)}(O_L, \Lambda_0) \overset{*}{=} \min_{\Lambda} L_{\text{err}}^{(t-1)}(O_L, \Lambda). \tag{A.8}$$

Note that $(*)$ holds since Λ_0 implies that no regularization is done—we get the same error as the minimum error of the previous iteration. Thus, the sequence is monotonically non-increasing. Additionally, the minimum error is bounded from below by zero. Any bounded monotone sequence converges to a finite limit. □

Proof of Proposition 5.1 (Probability of events)

(a) First, $p\left(V_t, V_{t+1}\right) = p\left(V_{t+1}|V_t\right) p\left(V_t\right)$. On the one hand, since a random walk is a time-homogeneous Markov chain, $p\left(V_{t+1}|V_t\right)$ is constant as t varies. On the other hand, the limit of $p\left(V_t\right)$ as $t \to \infty$ is the (first-order) stationary distribution of the random walk, which exists uniquely for an arbitrary initial state V_0, subject to the irreducibility and aperiodicity

of the graph [79]. Thus, $\lim_{t\to\infty} p\left(V_t, V_{t+1}\right) = p\left(V_{t+1}|V_t\right) \lim_{t\to\infty} p\left(V_t\right)$, which also exists uniquely regardless of the initial state V_0.

Now we discuss the irreducibility and aperiodicity of the graph. In particular, our graph (Eq. 5.1) satisfies such conditions. As a minor caveat, it is a popular practice to construct a kNN graph instead, where two points x_i and x_j are connected only if x_i is among the k nearest neighbors of x_j or x_j is among the k nearest neighbors of x_i. The two conditions are generally satisfied in a kNN graph as well; in the rare case where the conditions are not met, a simple tweak is to add some dummy edges of small weights [52].

(b) Since $(V_t, V_{t+1}) \xrightarrow{d} (X, X')$, we have

$$\begin{aligned} p\left(X = x_i, X' = x_j\right) &\overset{1}{=} \lim_{t\to\infty} p\left(V_t = x_i, V_{t+1} = x_j\right) \\ &\overset{2}{=} \lim_{t\to\infty} p\left(V_{t+1} = x_j | V_t = x_i\right) p\left(V_t = x_i\right) \\ &\overset{3}{=} \frac{W_{ij}}{Z_i} \lim_{t\to\infty} p\left(V_t = x_i\right) \\ &\overset{4}{=} \frac{W_{ij}}{Z_i} \frac{Z_i}{\sum_{uv} W_{uv}} \\ &\overset{5}{=} \frac{W_{ij}}{\sum_{uv} W_{uv}} \quad \overset{6}{\propto} W_{ij} \end{aligned} \tag{A.9}$$

In step 3, $p\left(V_{t+1} = x_j | V_t = x_i\right) = W_{ij}/Z_i$ is given by the transition probability. In step 4, $\lim_{t\to\infty} p\left(V_t = x_i\right) = Z_i / \sum_{uv} W_{uv}$ is the (first-order) stationary distribution of the random walk, which is established elsewhere [79].

Next, we find $p\left(X = x_i\right)$ by marginalizing the joint distribution $p\left(X, X'\right)$.

$$p\left(X = x_i\right) = \sum_j p\left(X = x_i, X' = x_j\right) = \sum_j \frac{W_{ij}}{\sum_{uv} W_{uv}} = \frac{Z_i}{\sum_{uv} W_{uv}} \propto Z_i \tag{A.10}$$

Finally, $p\left(X, X' = x_i\right)$ can be found similarly. □

Proof of Proposition 5.2 (Label coupling)

We can bound the L_1 difference as follows:

$$\begin{aligned}
&\frac{1}{2}\left\|p\left(Y|X=x_i\right)-p\left(Y|X,\,X'=x_i\right)\right\|_1 \\
&=\frac{1}{2}\left\|(1-\alpha)p\left(Y|X,\,X'=x_i\right)+\alpha D-p\left(Y|X,\,X'=x_i\right)\right\|_1 \\
&=\frac{1}{2}\alpha\left\|D-p\left(Y|X,\,X'=x_i\right)\right\|_1 \qquad \leq \alpha.
\end{aligned} \tag{A.11}$$

The inequality in the last step follows since the L_1 difference between any two distributions is at most 2. Thus, the proof is concluded. □

Proof of Proposition 5.3 (Solution of Generative Constraints)

(a) Our goal is to derive the transition matrix P. As the unlabeled constraint (Eq. 5.13) already tells us the transition from each state x_j to $x_i \notin \mathcal{L}$, we now need to find out the transition from each x_j to $x_i \in \mathcal{L}$, *i.e.*, to express K (Eq. 5.11) as a function of π_{yj}.

$$\begin{aligned}
K &\overset{1}{=} \sum_{k:x_k\in\mathcal{L}} \pi_{yk} \overset{2}{=} 1-\sum_{k:x_k\notin\mathcal{L}} \pi_{yk} \\
&\overset{3}{=} 1-\sum_{k:x_k\notin\mathcal{L}}\left((1-\alpha)\sum_j \frac{W_{jk}}{Z_j}\pi_{yj}\right) \\
&\overset{4}{=} 1-(1-\alpha)\sum_j\sum_{k:x_k\notin\mathcal{L}}\frac{W_{jk}}{Z_j}\pi_{yj} \\
&\overset{5}{=} 1-\sum_j\sum_{k:x_k\notin\mathcal{L}}\frac{W_{jk}}{Z_j}\pi_{yj}+\alpha\sum_j\sum_{k:x_k\notin\mathcal{L}}\frac{W_{jk}}{Z_j}\pi_{yj} \\
&\overset{6}{=} \sum_j\sum_k\frac{W_{jk}}{Z_j}\pi_{yj}-\sum_j\sum_{k:x_k\notin\mathcal{L}}\frac{W_{jk}}{Z_j}\pi_{yj}+\alpha\sum_j\sum_{k:x_k\notin\mathcal{L}}\frac{W_{jk}}{Z_j}\pi_{yj} \\
&\overset{7}{=} \sum_j\sum_{k:x_k\in\mathcal{L}}\frac{W_{jk}}{Z_j}\pi_{yj}+\alpha\sum_j\sum_{k:x_k\notin\mathcal{L}}\frac{W_{jk}}{Z_j}\pi_{yj} \\
&\overset{8}{=} \sum_j\frac{\sum_{k:x_k\in\mathcal{L}}W_{jk}+\alpha\sum_{k:x_k\notin\mathcal{L}}W_{jk}}{Z_j}\pi_{yj}
\end{aligned} \tag{A.12}$$

Step 2 follows since π_y is a distribution and its entries sum up to 1. Step 3 applies the constraint on unlabeled data (Eq. 5.13). In step 6, $\sum_j \sum_k \frac{W_{jk}}{Z_j} \pi_{yj} = \sum_j \left(\pi_{yj} \sum_k \frac{W_{jk}}{Z_j} \right) = \sum_j \pi_{yj} = 1$. Thus, $\forall i : x_i \in \mathcal{L}$, we have

$$\pi_{yi} = K \cdot \theta_{yi} = \sum_j \frac{\sum_{k:x_k \in \mathcal{L}} W_{jk} + \alpha \sum_{k:x_k \notin \mathcal{L}} W_{jk}}{Z_j} \pi_{yj} \cdot \theta_{yi}. \tag{A.13}$$

It is easy to verify that $\pi_y P = \pi_y$ and $\sum_i P_{ji} = 1$, concluding the first part of the proof.

(b) An irreducible and aperiodic Markov chain has a unique stationary distribution. P is irreducible and aperiodic except in one case below. When $\theta_{yi} = 0$ for some $i : x_i \in \mathcal{L}$, P is not irreducible since $P_{ji} = 0$. When this happens, it means $p(y|x_i) = 0$, which also implies $\pi_{yi} = 0$. Thus, we can do a minor tweak by excluding x_i from the state space $\mathcal{X}$, and derive a new transition matrix P' over $\mathcal{X} \backslash \{x_i\}$, which is always irreducible. The solution would then be π'_y corresponding to P', coupled with $\pi_{yi} = 0$. Alternatively, we can assume $p(y|x_i) \geq \epsilon, \forall x_i \in \mathcal{X}$, where ϵ is a very small positive constant. □

Proof of Proposition 5.3 (Solution to Generative Constraints)

Our solution estimator $\hat{\pi}_y$ can differ from the true π_y due to insufficient samples, which produce two types of error as follows.

First, there exists *data sampling error*. We only observe a potentially partial $\hat{\mathcal{X}}$ for $\mathcal{X}$, through samples in $\mathcal{L}$ and $\mathcal{U}$. Hence, the affinity matrix W would not be correctly constructed, resulting in a different transition matrix P. (Here we suppose that the graph construction function itself is perfect.)

Second, there exists *label sampling error*. We estimate θ_y as $\hat{\theta}_y$ based on $\mathcal{L}$, which is potentially erroneous, also resulting in a different P.

Corresponding to the two types of error, we consider two scenarios $\hat{\mathcal{X}} \subset \mathcal{X}$ and $\hat{\mathcal{X}} = \mathcal{X}$. On the one hand, when $\hat{\mathcal{X}} \subset \mathcal{X}$, we do not need to consider the second type of error caused

by $\hat{\theta}_y$, since regardless of the error in $\hat{\theta}_y$, the error in $\hat{\pi}_y$ can be as large as 2 (the maximal L_1 difference between two distributions). On the other hand, when $\hat{\mathcal{X}} = \mathcal{X}$, the first type of error does not exist, and we only need to investigate the error caused by $\hat{\theta}_y$. Formally,

$$\begin{aligned}\mathbb{E}\left[\left\|\hat{\pi}_y - \pi_y\right\|_1\right] &= \mathbb{E}\left[\left\|\hat{\pi}_y - \pi_y\right\|_1 \middle| \hat{\mathcal{X}} \subset \mathcal{X}\right] p\left(\hat{\mathcal{X}} \subset \mathcal{X}\right) + \mathbb{E}\left[\left\|\hat{\pi}_y - \pi_y\right\|_1 \middle| \hat{\mathcal{X}} = \mathcal{X}\right] p\left(\hat{\mathcal{X}} = \mathcal{X}\right) \\ &\leq 2p\left(\hat{\mathcal{X}} \subset \mathcal{X}\right) + \mathbb{E}\left[\left\|\hat{\pi}_y - \pi_y\right\|_1 \middle| \hat{\mathcal{X}} = \mathcal{X}\right]. \end{aligned} \tag{A.14}$$

Now, we only need to bound $p\left(\hat{\mathcal{X}} \subset \mathcal{X}\right)$ and $\mathbb{E}\left[\left\|\hat{\pi}_y - \pi_y\right\|_1 \middle| \hat{\mathcal{X}} = \mathcal{X}\right]$ for the two scenarios, respectively. Let $\mathcal{L}_{x_i}$ (or $\mathcal{U}_{x_i}$) denote the set that contains all the samples with x_i in $\mathcal{L}$ (or $\mathcal{U}$). Similarly, let $\mathcal{L}_y$ denote the set that contains all the samples with y in $\mathcal{L}$. Note that since the samples are i.i.d, there can be different samples with x_i or y.

The first scenario $\hat{\mathcal{X}} \subset \mathcal{X}$ happens only when there exists some $x_i \in \mathcal{X}$, such that $p(x_i) > 0$ but it is not sampled in $\mathcal{L}$ or $\mathcal{U}$. Hence,

$$\begin{aligned} p\left(\hat{\mathcal{X}} \subset \mathcal{X}\right) &\overset{1}{=} p\left(\bigcup_{x_i \in \mathcal{X}, p(x_i)>0} |\mathcal{L}_{x_i}| + |\mathcal{U}_{x_i}| = 0\right) \\ &\overset{2}{\leq} \sum_{x_i \in \mathcal{X}, p(x_i)>0} p\left(|\mathcal{L}_{x_i}| + |\mathcal{U}_{x_i}| = 0\right) \\ &\overset{3}{=} \sum_{x_i \in \mathcal{X}, p(x_i)>0} \left(1 - p(x_i)\right)^{|\mathcal{L}|+|\mathcal{U}|} \\ &\overset{4}{=} O\left(\left(1 - \min_{x_i \in \mathcal{X}, p(x_i)>0} p(x_i)\right)^{|\mathcal{L}|+|\mathcal{U}|}\right). \end{aligned} \tag{A.15}$$

Step 2 is an application of Bonferroni inequalities. Step 3 follows since the samples in $\mathcal{L}$ and $\mathcal{U}$ are i.i.d. In step 4, $\min_{x_i \in \mathcal{X}, p(x_i)>0} p(x_i)$ is a constant in $(0, 1]$, which we denote as λ_1. Furthermore, if we consider $|\mathcal{U}| \gg |\mathcal{L}|$, we can rewrite Eq. A.15 as follows:

$$p\left(\hat{\mathcal{X}} \subset \mathcal{X}\right) \leq O\left(\left(1 - \lambda_1\right)^{|\mathcal{U}|}\right). \tag{A.16}$$

In the second scenario $\hat{\mathcal{X}} = \mathcal{X}$, we investigate the second type of error due to $\hat{\theta}_y$. Note that θ_{yi} itself is defined in terms of $p(y|x_i)$ (Eq. 5.11). Thus, we need to quantify the error

in $p(y|x_i)$, further translate it to the error in $\hat{\theta}_y$, and finally derive the error in $\hat{\pi}_y$.

We first bound the error $|\hat{p}(y|x_i) - p(y|x_i)|$. Note that we only need to consider x_i such that $p(y|x_i) > 0$, since when $p(y|x_i) = 0$, we have $|\hat{p}(y|x_i) - p(y|x_i)| = 0$ almost surely. Recall that $\hat{p}(y|x_i)$ is estimated as the sample mean $|\mathcal{L}_y \cap \mathcal{L}_{x_i}|/|\mathcal{L}_{x_i}|$. By Hoeffding's inequality, the error in the sample mean can be bounded. That is, for any constant $\epsilon \in (0, 1)$,

$$p\left(|\hat{p}(y|x_i) - p(y|x_i)| > p(y|x_i)\,\epsilon\right) \leq 2\exp\left(-2p(y|x_i)^2\,\epsilon^2|\mathcal{L}_{x_i}|\right). \tag{A.17}$$

To obtain a bound on the error in $\hat{\theta}_y$, we need $|\hat{p}(y|x_i) - p(y|x_i)|$ to be bounded for all $x_i \in \mathcal{L}$ in conjunction, whose probability can be computed as follows.

$$\begin{aligned}
&p\left(\bigcap_{x_i\in\mathcal{L},p(y|x_i)>0} |\hat{p}(y|x_i) - p(y|x_i)| \leq p(y|x_i)\,\epsilon\right) \\
&\overset{1}{=} 1 - p\left(\bigcup_{x_i\in\mathcal{L},p(y|x_i)>0} |\hat{p}(y|x_i) - p(y|x_i)| > p(y|x_i)\,\epsilon\right) \\
&\overset{2}{\geq} 1 - \sum_{x_i\in\mathcal{L},p(y|x_i)>0} p\left(|\hat{p}(y|x_i) - p(y|x_i)| > p(y|x_i)\,\epsilon\right) \\
&\overset{3}{\geq} 1 - 2\sum_{x_i\in\mathcal{L},p(y|x_i)>0} \exp\left(-2p(y|x_i)^2\,\epsilon^2|\mathcal{L}_{x_i}|\right) \\
&\overset{4}{=} 1 - \rho
\end{aligned} \tag{A.18}$$

Step 1 follows from De Morgan's laws. Step 2 is an application of Bonferroni inequalities. Step 3 follows from Eq. A.17. In step 4 we denote $\rho \triangleq 2\sum_{x_i\in\mathcal{L},p(y|x_i)>0} \exp\left(-2p(y|x_i)^2\,\epsilon^2|\mathcal{L}_{x_i}|\right)$.

Next, we can translate the error $|\hat{p}(y|x_i) - p(y|x_i)|$ to $\left|\hat{\theta}_{yi} - \hat{\theta}_{yi}\right|$. Eq. A.18 means that there is a probability of at least $1 - \rho$ such that $\forall x_i \in \mathcal{L}, p(y|x_i) > 0$,

$$1 - \epsilon \leq \frac{\hat{p}(y|x_i)}{p(y|x_i)} \leq 1 + \epsilon. \tag{A.19}$$

Hence, with probability at least $1 - \rho$, $\forall x_i \in \mathcal{L}, p(y|x_i) > 0$,

$$\frac{1-\epsilon}{1+\epsilon} \leq \frac{\hat{\theta}_{yi}}{\theta_{yi}} \leq \frac{1+\epsilon}{1-\epsilon} \quad \Rightarrow \quad \left|\hat{\theta}_{yi} - \theta_{yi}\right| \leq \frac{2\epsilon}{1-\epsilon}\theta_{yi} \tag{A.20}$$

Finally, we can translate the error $\left|\hat{\theta}_{yi} - \hat{\theta}_{yi}\right|$ to $\|\hat{\pi}_y - \pi_y\|_1$. Based on the perturbation theory of Markov chains [27, 92], we can bound $\|\hat{\pi}_y - \pi_y\|_1$ in terms of the ∞-norm of the error matrix $\hat{P} - P$. Specifically, when $\hat{\mathcal{X}} = \mathcal{X}$, with probability at least $1 - \rho$,

$$\begin{aligned}
\|\hat{\pi}_y - \pi_y\|_1 &\overset{1}{\leq} C \left|\hat{P} - P\right|_\infty \\
&\overset{2}{=} C \max_j \sum_i \left|\hat{P}_{ji} - P_{ji}\right| \\
&\overset{3}{=} C \max_j \sum_{i:x_i \in \mathcal{L}} \frac{\sum_k \alpha^{\mathbf{1}\{k:x_k \notin \mathcal{L}\}} W_{jk}}{Z_j} \left|\hat{\theta}_{yi} - \theta_{yi}\right| \\
&\overset{4}{\leq} C \max_j \sum_{i:x_i \in \mathcal{L}} \left|\hat{\theta}_{yi} - \theta_{yi}\right| \\
&\overset{5}{\leq} C \max_j \sum_{i:x_i \in \mathcal{L}} \frac{2\epsilon}{1-\epsilon} \theta_{yi} \\
&\overset{6}{=} \frac{2C\epsilon}{1-\epsilon}.
\end{aligned} \tag{A.21}$$

Step 1 follows from the perturbation theory of Markov chains, where C is called a condition number and is a constant for a given P. In step 3, we only sum over i where $x_i \in \mathcal{L}$, since $\hat{P}_{ji} = P_{ji}$ when $x_i \notin \mathcal{L}$. Step 4 follows since $Z_j = \sum_k W_{jk} \geq \sum_k \alpha^{\mathbf{1}\{k:x_k \notin \mathcal{L}\}} W_{jk}$ for $0 < \alpha < 1$. Step 5 applies Eq. A.20.

As $\|\hat{\pi}_y - \pi_y\|_1$ is also bounded by 2 (the maximal L_1 difference between two distributions), when $\hat{\mathcal{X}} = \mathcal{X}$, we can determine with probability at least $1 - \rho$,

$$\|\hat{\pi}_y - \pi_y\|_1 \leq \min\left\{2, \frac{2C\epsilon}{1-\epsilon}\right\}. \tag{A.22}$$

Let $\delta \triangleq \min\left\{2, \frac{2C\epsilon}{1-\epsilon}\right\}$, which is a constant. It is easy to see that

$$\begin{aligned}
\mathbb{E}\left[\|\hat{\pi}_y - \pi_y\|_1 \middle| \hat{\mathcal{X}} = \mathcal{X}\right] &\leq \delta(1-\rho) + 2\rho \\
&= \delta + (2-\delta)\rho \\
&= O(\rho) \\
&= O\left(\exp\left(-2\epsilon^2 \min_{x_i \in \mathcal{L}, p(y|x_i) > 0} p\left(y|x_i\right)^2 \min_{x_i \in \mathcal{L}} |\mathcal{L}_{x_i}|\right)\right).
\end{aligned} \tag{A.23}$$

Here $\min_{x_i \in \mathcal{L}, p(y|x_i)>0} p\left(y|x_i\right)^2$ is a constant in $(0, 1]$, which we denote as λ_2. Plugging Eq. A.16 and A.23 into Eq. A.14, we conclude the proof. □

Proof of Proposition 5.5 (Robustness Analysis)

Let $\tilde{P}$ be an estimator of the transition matrix based on $\tilde{W}$. As $W_{ij}/s \leq \tilde{W}_{ij} \leq W_{ij} \cdot s$, we can derive that

$$P_{ji}/s^2 \leq \tilde{P}_{ji} \leq P_{ji} \cdot s^2 \quad \Rightarrow \quad \left|\tilde{P}_{ji} - P_{ji}\right| \leq (s^2 - 1)P_{ji}. \tag{A.24}$$

From the sensitivity theory of Markov chains [27, 92], for a condition number C which is a constant for a given P,

$$\begin{aligned} L_{\text{err}}^{(\tilde{\pi}_y - \pi_y)} &\leq C \left|\tilde{P} - P\right|_\infty && \text{(A.25)} \\ &= C \max_j \sum_i \left|\tilde{P}_{ji} - P_{ji}\right| \\ &\leq C \max_j \sum_i (s^2 - 1)P_{ji} \\ &= C(s^2 - 1). && \text{(A.26)} \end{aligned}$$

Hence, the proof is concluded. □