Result Diversification on Spatial, Multidimensional, Opinion, and Bibliographic Data

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Onur Küçüktunç, B.S., M.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2013

Dissertation Committee:

Prof. Ümit V. Çatalyürek, Advisor

Prof. Srinivasan Parthasarathy

Prof. Arnab Nandi

© Copyright by

Onur Küçüktunç

2013

Abstract

Similarity search methods in the literature produce results based on the ranked degree of similarity to the query. However, the results are typically unsatisfactory, especially if there is an ambiguity in the query, or the search space include redundantly repeating similar documents. Diversity in query results is preferred by a variety of applications since diverse results may give a complete view of the queried topic. In this study, we investigate the *result diversification* task in various application areas, such as opinion retrieval, paper recommendation, with different types of data, such as spatial, high-dimensional data, opinions, citation graph, and other networks. Although the definitions of diversity will differ from field to field, we propose techniques considering the general objective of *result diversification*, which is to maximize the similarity of search results to the query while minimizing the pairwise similarity between the results, without neglecting the efficiency.

For the diversity on spatial and high-dimensional data, we make an analogy with the concept of natural neighbors and propose geometric methods. We also introduce a diverse browsing method based on the popular distance browsing feature of R-tree index structures.

Next, we focus on search and retrieval of opinion data on certain entities, and start our analysis by looking at direct correlations between sentiments of opinions and the demographics (e.g., gender, age, education level, etc.) of people that generate those opinions. Based on the analysis, we argue that opinion diversity can be achieved by diversifying the sources of opinions.

Recommendation tasks on academic networks also suffer from the mentioned ambiguity and redundancy issues. To observe those effects, we present a paper recommendation framework called the**advisor** (http://theadvisor.osu.edu) which recommends new papers to researchers using only the reference-citation relationships between academic papers. We introduce *direction awareness* property to the recommendation process, which allows the users to reach either old, foundational (possibly well-cited and well-known) research papers or recent (most likely less-known) ones. We also present different implementations and ordering techniques for reducing the query processing time. Finally, we enhance various result diversification techniques with direction-awareness property for paper recommendation, propose new algorithms based on vertex selection and query refinement, and compare in this domain.

Based on our findings on diversifying citation recommendations, we further extend the diversity of graph-based recommendation algorithms for other types of graphs, such as social and collaboration networks, web and product co-purchasing graphs. Although the diversification problem is understandably addressed as a bi-criteria objective optimization problem over relevance and diversity, the sufficiency of the evaluations of such methods are questionable since a *query-oblivious* algorithm that returns most of its recommendations without considering the query may still perform the best on these commonly used measures. We show the deficiencies of commonly preferred evaluation techniques of diversification methods, propose a new measure called *expanded relevance* which combines relevance and diversity. Finally, we present a novel algorithm that optimizes the *expanded relevance* of the diversified results.

To my family

Acknowledgments

I would like to give my sincerest gratitude to my advisor, Ümit V. Çatalyürek, for his instructive comments, invaluable support, and encouragement during my doctoral study. I also owe many thanks to him and his family for their hospitality.

I am indebted to all my advisory committee members, Srinivasan Parthasarathy and Arnab Nandi, for spending their time and effort to read and comment on my dissertation; and also Tunç Aldemir for serving as the graduate school representative during my dissertation defense. I would like to express my cordial acknowledgment to my former advisor, Hakan Ferhatosmanoğlu, for guiding me to the right research area and for our invaluable discussions during the first years of my doctoral study.

In particular, I would like to thank my postdocs and mentors: Erik Saule, Kamer Kaya, and B. Barla Cambazoğlu, for their help and contributions to my research. I would also like to thank my fellow lab members, past and present, for their assistance and support over the years.

Last, but not least, many thanks to my family and Daniya Zamalieva for their care and support during these years.

Thanks all.

Vita

2007	B.S. Computer Engineering, Bilkent University.
2009	M.S. Computer Engineering, Bilkent University.
2009 – present	Graduate Research Associate, Computer Science and Engineering, The Ohio State University.

Publications

Research Publications

Onur Küçüktunç and Hakan Ferhatosmanoğlu. Diverse browsing for spatial data. Technical Report OSU-CISRC-1/11-TR02, OSU CSE, Feb 2011.

Onur Küçüktunç and Hakan Ferhatosmanoğlu. λ-diverse nearest neighbors browsing for multidimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):481–493, Mar 2013.

Onur Küçüktunç, B. Barla Cambazoğlu, Ingmar Weber, and Hakan Ferhatosmanoğlu. A large-scale sentiment analysis for Yahoo! Answers. In *Proc. ACM Int'l Conf. Web Search and Data Mining (WSDM)*, 2012.

Kemal Eren, Mehmet Deveci, Onur Küçüktunç, and Ümit V. Çatalyürek. A comparative analysis of biclustering algorithms for gene expression data. *Briefings in Bioinformatics*, 14(3):279–292, May 2013.

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Direction awareness in citation recommendation. In *Proc. Int'l Workshop on Ranking in Databases (DBRank'12) in conjunction with VLDB'12*, 2012.

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Recommendation on academic networks using direction aware citation analysis. Technical Report arXiv:1205.1143, ArXiv, Apr 2012.

Onur Küçüktunç, Kamer Kaya, Erik Saule, and Ümit V. Çatalyürek. Fast recommendation on bibliographic networks. In *Proc. IEEE/ACM Int'l Conf. Advances in Social Networks Analysis and Mining (ASONAM)*, Aug 2012.

Onur Küçüktunç, Kamer Kaya, Erik Saule, and Ümit V. Çatalyürek. Fast recommendation on bibliographic networks with sparse-matrix ordering and partitioning. *Social Network Analysis and Mining (SNAM)*, 2013. (to appear).

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Diversifying citation recommendations. Technical Report arXiv:1209.5809, ArXiv, Sep 2012.

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Diversifying citation recommendations. *ACM Trans. Information Systems and Technology*, 2013. (under review).

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Result diversification in automatic citation recommendation. In *iConference Workshop on Computational Scientometrics: Theory and Applications*, 2013.

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Diversified recommendation on graphs: Pitfalls, measures, and algorithms. In *Proc. Int'l Conf. World Wide Web (WWW)*, 2013.

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. TheAdvisor: A webservice for academic recommendation. In *Proc. ACM/IEEE Joint Conf. Digital Libraries (JCDL)*, 2013. (poster).

Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Towards a personalized, scalable, and exploratory academic recommendation service. In *Proc. IEEE/ACM Int'l Conf. Advances in Social Networks Analysis and Mining (ASONAM)*, Aug 2013.

Fields of Study

Major Field: Computer Science and Engineering

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Similarity search methods in the literature produce results based on the ranked degree of similarity to the query. However, the results are typically unsatisfactory, especially if there is an ambiguity in the query, or the search space include redundantly repeating similar documents. Diversity in query results is preferred by a variety of applications since diverse results may give a complete view of the queried topic.

In this study, we investigate the *result diversification* task in various application areas (such as opinion retrieval, paper recommendation) with different types of data (such as spatial, high-dimensional data, opinions, citation graph, and other networks). Although the definitions of diversity will differ from field to field, we propose techniques considering the general objective of *result diversification*, which is to maximize the similarity of search results to the query while minimizing the pairwise similarity between the results, without neglecting the efficiency.

1.1 Diversity on Spatial and Multidimensional Data

When querying and browsing **spatial data**, diversity of the results can be very powerful. With the popularity of location-based services, most smartphones today come with GPS and solid state compasses, and people use this technology for navigation and searching nearby restaurants, gas stations, etc. A location-based service application that defines user's position with a number of point-of-interests (POI) can be affected adversely with the information overload. Instead of returning the closest POIs, close but a more diverse result set is preferred for such an application.

Considering the diversification problem in the spatial domain, it is possible to present an intuitive solution based on clustering. Data can be initially clustered and then representatives of clusters around the query point can be returned as search results. Although clustering can be computationally expensive, there are methods to generate those representatives with tree-based approaches [94]. The problem of clustering-based methods is that initial clusters may be unsatisfactory depending on the settings of the query. Furthermore, if data needs to be clustered for each query, the method is obviously not scalable. Today, most database management systems support a spatial index (e.g., R-tree or one of its variants). Therefore, diversity can be obtained by taking advantage of the spatial index without the extra cost of clustering [68, 69].

In Chapter 2, we first give a geometric definition of *diversity* by making an analogy with the concept of natural neighbors and propose a natural neighbor-based method and an incremental browsing algorithm based on Gabriel graph for diverse nearest neighbor search problem. We also introduce a diverse browsing method based on the popular distance browsing feature of R-tree index structures. The method maintains a priority-queue with the ranks of the objects depending on both relevancy and diversity, and efficiently prunes non-diverse items and nodes. Providing a measure that captures both relevancy and diversity, we show that pruning internal nodes with respect to their diversity from the items in the result set helps us achieve more diverse results. The contributions of this work can be summarized as follows:

- We formalize the **λ-diverse k-nearest neighbor search problem** based on angular similarity and develop measures to evaluate the relevancy and diversity of the retrieved results.
- We propose two geometric diverse browsing approaches for static databases, each of which effectively captures the spatial distribution around a query point and hence gives a diverse set of results.
- Extending the distance browsing feature, we introduce an efficient λ-diverse k-nearest neighbor search algorithm on R-trees, diverse browsing, and prove its correctness.
- We conduct experiments on 2D and high-dimensional datasets to evaluate the performance of the proposed methods.

Key advantages of the proposed geometric and index-based methods are: (1) geometric methods are appropriate for static databases and perform very efficiently once the graphs are built, (2) index-based diverse browsing does not require any change in the index structure, therefore, it can easily be integrated into various databases, and (3) with effective pruning, diverse browsing performs more efficiently than k-nearest neighbor (k-NN) search on R-trees and also performs better than the state-of-the-art techniques.

Results suggest that geometric approaches are suitable for static data, and index-based diverse browsing is for dynamic databases. Our index-based diverse browsing method performed more efficient than k-NN search with distance browsing on R-tree (in terms of the number of disk accesses) and more effective than other methods found in the literature (in terms of Maximal Marginal Relevance). In addition, Gabriel graph-based method performed well in high dimensions, which can be investigated more and applied to other research fields where search in high dimensional space is required.

1.2 Sentiment Analysis and Opinion Diversification

Increasing popularity of personal web content via blogs, tweets, and other types of social media has boosted the generation of public **opinions** on certain famous people (politicians, singers, etc.), locations (cities, countries, etc.) and brands (products, companies, organizations, etc.). Search and retrieval of opinions on a subject is extremely useful for a number of application areas, including reputation management. However, due to the fact that there might be enormous number of opinions on a subject, or the opinions are skewed towards a viewpoint in some cases, the task of sampling and ranking of the opinions by representing all the viewpoints is a complex task.

In Chapter 3, we start our analysis by looking at direct correlations between sentiments of opinions and the demographics of people (e.g., gender, age, education level, etc.) that generate those opinions. The details of a large-scale sentiment analysis on Yahoo! Answers data is given in [67]. We start our analysis by looking at direct correlations, e.g., we observe more positive sentiments on weekends, very neutral ones in the Science & Mathematics topic, a trend for younger people to express stronger sentiments, or people in military bases to ask the most neutral questions. We then extend this basic analysis by investigating how properties of the (asker, answerer) pair affect the sentiment present in the answer. Among other things, we observe a dependence on the pairing of some inferred attributes estimated by a user's ZIP code. We also show that the best answers differ in their sentiments from other answers, e.g., in the Business & Finance topic, best answers tend to have a more neutral sentiment than other answers. Finally, we report results for the task of predicting the attitude that a question will provoke in answers. We believe that understanding factors influencing the mood of users is not only interesting from a sociological point of view, but also has applications in advertising, recommendation, and search.

Since the availability of a topical classification makes it possible to differentiate sentiments attached to a particular entity according to the context, we investigate this property to obtain a more faceted representation of the opinions about an entity. Based on the hypothesis of *network diversity is positively associated with receiving more diverse and less redundant information*, we argue that **opinion diversity** can also be achieved by diversifying the sources of opinions, which is basically finding a set of opinions from various opinion holders with diverse demographics. We discuss the outlines of such an opinion diversification framework and leave the practical application as a future work for those who have access to user profiles of social networks.

1.3 Graph-based Paper Recommendation and Diversity

Next, we focus on the paper recommendation problem on **academic networks**, and how the mentioned ambiguity and redundancy issues interacts with the users' preferences and satisfiability of the results. To observe those effects, we built a paper recommendation service called the**advisor**[1] which recommends new papers to researchers using only the reference-citation relationships between academic papers [79].

Chapter 4 gives the details of the methods used in the service and how the recommendations are scored with *direction awareness* if the intent of the research if to find

[1]http://theadvisor.osu.edu

recent or tradition papers [75, 72]. Our aim in this work is to evaluate the existing algorithms and to explain the new algorithms that power the**advisor**. We introduce a class of parametric algorithms, said to be *direction aware*, which allow to give more importance to either the citation of papers or their references. They make the citation suggestion process easily tunable for finding either recent or traditional relevant papers. In particular we extend two eigenvector based methods into direction-aware algorithms, namely DaRWR and DaKatz. These algorithms are compared to state-of-the-art citation-based algorithms for bibliographic recommendation and their adequation to the problem is studied. In our experiments, direction-aware algorithms outperform the existing algorithms when the objective is to find either traditional or recent papers. We deployed one of the algorithms in our web service which allows any researcher to upload a bibliography file and obtain suggestions. We believe that our service will become a tool of major interest for researchers.

We also focus on different implementation and ordering techniques for reducing the query processing time for the recommendations [70, 71]. We propose an efficient implementation of the sparse matrix-dense vector multiplication (SpMxV)-type problem which arises in our publicly available citation, venue, and expert recommendation service, the**advisor**. We propose compression and bandwidth reduction techniques to reduce the memory usage and hence, the bandwidth required to bring the matrix from the memory at each iteration. We also use matrix ordering techniques to reduce the number cache misses. Our contribution is two-fold:

- We propose techniques to efficiently store the matrix used by our algorithm,
- We propose an efficient implementation of the algorithm and investigate several matrix ordering techniques based on a hypergraph partitioning model and

ordering heuristics, such as Approximate Minimum Degree (AMD) [6], Reverse Cuthill-McKee (RCM) [31], and SlashBurn [62].

We give a thorough evaluation of the proposed approach and measure the efficiency of implementation and matrix storing/ordering techniques used in the**advisor**. The combination of all the techniques improved the response time of our service by 67% (3x). We believe that the techniques proposed here can also be useful for SpMxV-related sparse-matrix problems in social network analysis.

Next, we investigate the result diversification task in paper recommendation problem [77, 78]. We target the bibliographic search and diversifying the results of the citation/paper recommendation process with the following objectives in mind: (1) the direction awareness property is kept, (2) the method should be efficient enough to be computable in real time, and (3) the results are relevant to the query and also diverse within the set. The contribution of this work is three-fold:

- We survey various random walk-based diversity methods (i.e., GrassHopper [147], DivRank [101] variants, and Dragon [128]) and relevancy/diversity measures. We enhance these algorithms with the direction awareness property.
- We propose new algorithms based on vertex selection (LM, γ-RLM) and query refinement (GSparse).
- We perform a rigorous set of experiments with various evaluation criteria and show that the proposed γ-RLM algorithm is suitable in practice for real-time diverse bibliographic search.

All of the algorithms in this study are implemented and tested within the**advisor** service. Our experiments with various relevancy and diversity measures show that the proposed γ-RLM algorithm can be preferred for both its efficiency and effectiveness.

In Chapter 5, we show how relevance feedback and result diversification affect and improve the service in practice, and how those different techniques complement each other to provide a powerful document discovery engine [79, 80].

1.4 Graph Diversity and Common Pitfalls in its Evaluation

Based on our findings on diversifying citation recommendations, in Chapter 6, we further extend the **diversity of graph-based recommendation algorithms** for other types of graphs, such as social and collaboration networks, web and product co-purchasing graphs. Result diversification has gained a lot of attention as a way to answer ambiguous queries and to tackle the redundancy problem in the results.

In this work, we assume that the graph itself is the only existing information, no categories or intents are available. We are interested in providing recommendation to the user based on a set of objects of interest. The recommended items should be related to the user's interest while being dissimilar to each other. This particular problem has attracted a lot of attention recently; many algorithms and evaluations have been proposed [147, 101, 128, 90, 37, 146, 27].

The objective evaluation of the quality of the algorithms is one of the interest of this work. Most commonly, algorithms are evaluated by expressing the problem as a bi-criteria objective optimization problem. One objective is selected for relevancy, most commonly the sum of personalized PageRank of the recommended items, and another one is selected for diversity, most commonly the density of the subgraph formed by the recommended set or its expansion ratio. The two objectives are either aggregated (often with a simple linear aggregation) or both objectives are considered simultaneously and Pareto dominance is considered (where the solutions are in the relevancy-diversity objective space). The first contribution of this work is to show that such an evaluation is inappropriate. Indeed we design *query-oblivious* algorithms for the two commonly used combination of objectives that return most of its recommendations without considering the user's interest, yet, perform best on these commonly used measures.

We argue that a result diversification algorithm should be evaluated under a measure which tightly integrates the query in its value. The *goodness* measure proposed in [128] has such a property; however, it is shown to be dominated by the relevance. We then propose a new measure called *expanded relevance* ($exprel_\ell$) which measures the coverage of the relevant part of the graph. We show that the previous query-oblivious algorithms cannot optimize $exprel_\ell$ [76].

We also investigate various quality indices by computing their pairwise correlation. This highlights that the *goodness* measure is highly correlated with the sum of ranking scores, meaning that algorithms that perform best on the *goodness* measure do not return results that are much different from a simple PageRank computation. The *exprel* metric we propose appears to have no high correlation with other metrics.

Based on the intuition behind $exprel_\ell$ measure, we propose a greedy algorithm, `BestCoverage`, to optimize $exprel_\ell$. Because of submodular properties of the $exprel_\ell$ objective, `BestCoverage` is an $(1 - 1/e)$-approximation algorithm for $exprel_\ell$ with complexity $O(kn\Delta^\ell)$ where n is the number of vertices in the graph and Δ is the maximum degree of the graph. We propose a relaxation of `BestCoverage` with complexity $O(k\bar{\delta}^\ell\Delta^\ell)$ where k is the number of recommended objects and $\bar{\delta}$ is the average degree of the graph. We experimentally show that the relaxation carries no significant harm to the *expanded relevance* of the solution.

Chapter 2: Diverse Browsing for Spatial and Multidimensional Data

Database search techniques try to obtain the most relevant information and rank it according to the degree of similarity to the queries. However, diversity in query results is also preferred by a variety of applications since results very similar to each other cannot capture all aspects of the queried topic. In this work, we focus on the **λ-diverse k-nearest neighbor search** problem on spatial and multi-dimensional data. Unlike the common approach of diversifying query results in a post-processing step, we naturally obtain diverse results with the proposed geometric and index-based methods.

In this work, we first make an analogy with the concept of natural neighbors and propose a natural neighbor-based method for 2D and 3D data and an incremental browsing algorithm based on Gabriel graphs for higher dimensional spaces. We then introduce a diverse browsing method based on the popular distance browsing feature of spatial index structures, such as R-trees. The algorithm maintains a priority queue with the ranks of the objects depending on both relevancy and angular diversity and efficiently prunes non-diverse items and nodes. We experiment with a number of spatial and high-dimensional datasets, including Factual[2]'s US points-of-interest

[2]http://www.factual.com/

dataset of 13M entries. With effective pruning, our diverse browsing method is shown to be more efficient (regarding disk accesses) than k-NN search on R-trees, and more effective (regarding Maximal Marginal Relevance) than the diverse nearest neighbor search techniques found in the literature.

2.1 Introduction

Most similarity search methods in the literature produce results based on the ranked degree of similarity to the query. However, the results could be unsatisfactory, especially when there is an ambiguity in the query or when the search results include redundantly similar data.

To resolve ambiguity, it would be better to answer the query with diverse search results instead of homogeneous results representing similar cases. For example, the query *Barcelona* is ambiguous since the system cannot decide whether it represents a city, a football team, or a movie [4]. A reasonable strategy for responding to an ambiguous query is to return a mixture of results covering all aspects of the query. Redundantly repeating search results is another problem of conventional similarity search techniques, particularly for search spaces that include many duplicate data. In this case, similar but homogeneous information will fill up the top results. This situation has been discussed in several application areas, such as recommender systems [149], online shopping [130], and web search [30].

Similar problems also exist when querying and browsing spatial data. In some applications, diversity is preferred over similarity due to the information overload (see Fig. 2.1). Suppose a criminal is spotted in NYC by a camera at 6th Ave & 33rd St (**C0**). Back in police department, the police have access to a number of cameras in

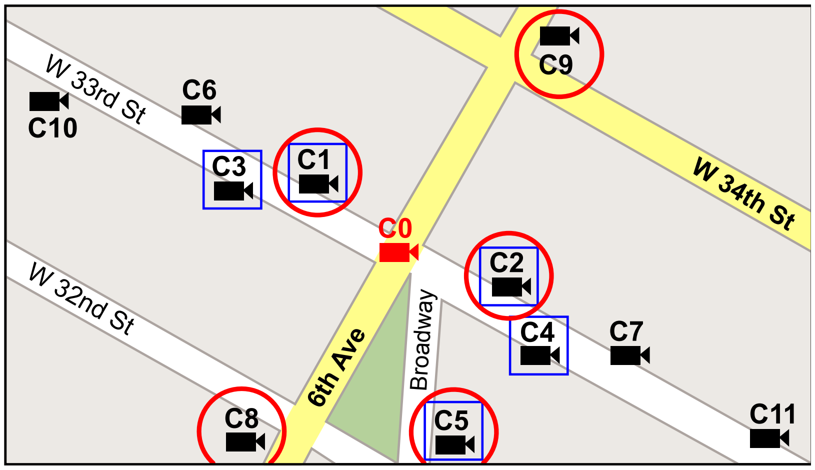


Figure 2.1: A location-based service application can be adversely affected with the information overload. A conventional similarity search (e.g., k-nearest neighbor search) returns four cameras in W 33rd street and one from Broadway (blue squares); whereas, diverse browsing can capture spatial distribution around the first camera and provide superior results (red circles).

the city, but have limited screens (say $k = 5$) to display the view of different cameras. The cameras are labeled in order by their distance from **C0**. Instead of returning the closest point-of-interests (POIs), a result set containing close yet diverse results is preferred for such an application.

Diversity in k-NN search is not limited to the spatial domain. A diverse k-NN classifier can be potentially useful for medical diagnosis since it is more likely to unveil minority reports by grouping and eliminating similar cases. Suppose that a number of medical records are labeled with '+' and '-' labels depending on whether a patient has disease D or not, respectively. Given a patient's medical records q, the aim of k-NN classifier is to classify the patient as D^+ or D^- by finding the majority class

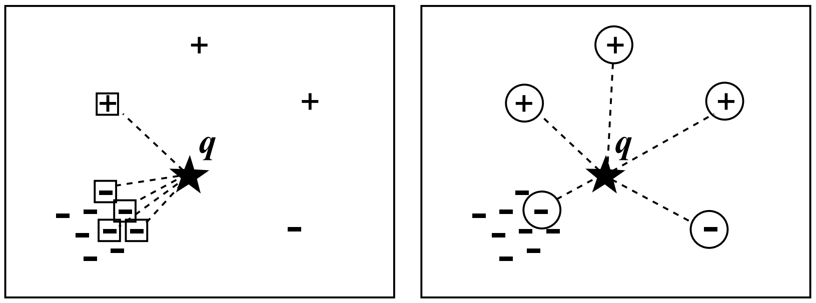


Figure 2.2: A classification example with k-NN (*left*) and diverse k-NN (*right*) classifiers for $k=5$. Although four out of five closest medical data points to q are D^-, a diverse perspective unveils minority reports and classifies the patient as D^+ with 60% confidence.

label for the closest k records. Fig. 2.2 depicts the classification results obtained from the k-NN classifier and diverse k-NN classifier.

The relation between diversity and relevance was investigated before, especially in text retrieval and summarization. Researchers have proposed linear combinations of diversity and relevance [18]. However, maximizing diversity of a result set is known to be NP-hard [19, 20]. Hence, some studies develop heuristic techniques [59, 141] to optimize the results.

Considering the diversification problem in the spatial domain, it is possible to present an intuitive solution based on clustering. Data can be initially clustered and then representatives of clusters around the query point can be returned as search results. Although clustering can be computationally expensive, there are methods to generate those representatives with tree-based approaches [94]. The problem of clustering-based methods is that initial clusters may be unsatisfactory depending on the settings of the query. Furthermore, if data needs to be clustered for each query, the method is obviously not scalable. Today, most database management systems support a spatial index (e.g., R-tree or one of its variants). Therefore, diversity can be obtained by taking advantage of the spatial index without the extra cost of clustering.

In this work, we first give a geometric definition of *diversity* by making an analogy with the concept of natural neighbors and propose a natural neighbor-based method and an incremental browsing algorithm based on Gabriel graph for diverse nearest neighbor search problem. We also introduce a diverse browsing method based on the popular distance browsing feature of R-tree index structures. The method maintains a priority-queue with the ranks of the objects depending on both relevancy and diversity, and efficiently prunes non-diverse items and nodes. Providing a measure that captures both relevancy and diversity, we show that pruning internal nodes with respect to their diversity from the items in the result set helps us achieve more diverse results. The contributions of this work can be summarized as follows:

- We formalize the **λ-diverse k-nearest neighbor search problem** based on angular similarity and develop measures to evaluate the relevancy and diversity of the retrieved results.
- We propose two geometric diverse browsing approaches for static databases, each of which effectively captures the spatial distribution around a query point and hence gives a diverse set of results.
- Extending the distance browsing feature, we introduce an efficient λ-diverse k-nearest neighbor search algorithm on R-trees, namely diverse browsing, and prove its correctness.

- We conduct experiments on 2D and high-dimensional datasets to evaluate the performance of the proposed methods.

Key advantages of the proposed geometric and index-based methods are:

- Geometric methods are appropriate for static databases and perform very efficiently once the graphs are built,
- Index-based diverse browsing does not require any change in the index structure, therefore, it can easily be integrated into various databases,
- With effective pruning, diverse browsing performs more efficiently than k-NN search on R-trees and also performs better than the state-of-the-art techniques regarding MMR metric.

2.2 Related work

There are notable works on diverse ranking in the literature. Carbonell and Goldstein [18] describe the Maximal Marginal Relevance (MMR) method for text retrieval and summarization. MMR attempts to find a result set by maximizing the query relevance and also minimizing the similarity between documents in the result set. The proposed method combines relevancy and novelty with a user-defined parameter (λ), which affects the importance of relevancy and diversity of the results.

Since the problem of finding diverse results is known to be NP-hard, Jain et al. [52, 59] investigate the k-nearest diverse neighbor search problem and develop two greedy approaches to optimize the results in terms of both relevancy and diversity. Both proposed methods employ the advantages of an available R-tree index. *Immediate Greedy* (IG) incrementally grows the result set R by including nearest points only if they are diverse enough from the data points already in R. *Buffered Greedy* (BG) tries to overcome some deficiencies of IG. They use the R-tree index only for getting the query's nearest neighbors in the dataset. Yu et al. [140, 141] address the issue of diversification in recommendation systems and introduce two heuristic algorithms to maximize the diversity under different relevance constraints. They state that maximizing diversity is about finding a balance between relevance and diversity. The proposed *Swap algorithm* basically tries to swap elements which are less likely to contribute to the set diversity with diverse ones. *Greedy algorithm*, similar to IG, includes the next most relevant item to the result set only if that item is diverse with respect to the items already in the result set.

Some other studies attack the diversity problem in various ways. Liu and Jagadish [94] employ the idea of clustering to find a solution to the Many-Answers Problem. They suggest that taking one representative from each cluster results in more diverse results. They propose a tree-based approach for efficiently finding the representatives, even if the search space is constrained at runtime. Halvey et al. [51] compare dissimilarity and clustering-based diverse re-ranking methods to introduce diversity in video retrieval results.

The notions of diversity and novelty are generally discussed in the context of information retrieval and recommendation systems. Clarke et al. [30] investigate the problems of ambiguity in queries and redundancy in results and propose an evaluation framework. Chen and Karger [25] describe a retrieval method which assigns negative feedback to the documents that are included in the result list for maximizing diversity. Vee et al. [130] present inverted list algorithms for computing diverse query results in online shopping applications. Ziegler et al. [148, 149] present an algorithmic framework to increase the diversity of a top-k list of recommended products. In order to show its efficiency, they also introduce a new intra-list similarity metric.

There are also works on content diversity over continuous data of publish/subscribe systems, such as news alerts, RSS feeds, social network notifications [35], and the diverse skyline [119]. Greedy heuristics were proposed for the problems although the discussions on how relevance and diversity should be combined, and how well greedy approaches approximate the optimal solution is fairly useful. Interested readers may refer to [96] for further information on diversity.

2.3 Preliminaries

Before we state our diversity and λ-diverse k-nearest neighbor search definitions, let us first analyze the approach used by KNDN-IG and KNDN-BG [52, 59]. For KNDN-IG the objective is to find a fully diverse set of results R close to the query point q. This means that for all $r_1, r_2 \in R$,

$$\text{divdist}(r_1, r_2, V(q)) = \sum_{j=1}^{L} (W_j \times \delta_j) > MinDiv,$$

where $V(q)$ is the diversity attributes, L is the number of dimensions to be diversified (in our case $L = d$), W is the set of weighting factors for the differences $(\delta_1, \ldots, \delta_L)$ sorted in decreasing order, and $MinDiv$ is the minimum diversity distance. The diversity computation in KNDN-IG is simply based on the Gower coefficient [49] with monotonically decaying weights W calculated with

$$W_j = \frac{a^{j-1} \times (1-a)}{1-a^L}, \qquad \text{for } 1 \leq j \leq L,$$

where a is the rate of decay.

KNDN-BG applies the same technique; however, it stores the eliminated points. Then the method checks if two of those points (say p' and p'') are pairwise-diverse, and also eliminated because of the same resulting point $r_i \in R$. If it finds such a pair, R is updated as $(R \setminus \{r_i\}) \cup \{p', p''\}$.

$MinDiv$ setting assumes that data is in $[0, 1]^d$ space; otherwise, it must be set with a knowledge of density and the range of data. When a is selected around 0.1, the dimension with the highest difference is overrated. This suggests that KNDN favors the points along the axes. Hence, it tries to find 2^d diverse points in a d-dimensional space. This behavior is problematic since k could be any number. Consequently, the results do not guarantee that KNDN accurately captures the distribution around the query point. Finally, these methods do not allow to select how diverse/relevant the results will be, unless the importance of diversity is embedded into either $MinDiv$ or a parameters.

Our diversity definition employs the angular similarity between two points regarding the query point. In other words, a diverse k-nearest neighbor search method should maximize the pairwise angular similarity while minimizing the overall distance. Angular similarity and diverse k-nearest neighbor search is defined in Definitions 2.3.1 and 2.3.2, respectively.

Definition 2.3.1. Angular similarity. *Given a query point q, two points p_1 and p_2, and an angle θ,* angular similarity *(sim_θ) of p_1 with respect to q and another point p_2 is:*

$$sim_\theta(p_1, q, p_2) = \begin{cases} 1 - \widehat{p_1 q p_2}/\theta, & \text{if } \widehat{p_1 q p_2} < \theta \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

sim_θ results in 0 if the angle $\widehat{p_1 q p_2}$ is greater than θ. It becomes 1 if both of them point at the same direction.

Table 2.1: Notation for diversity on spatial and multidimensional data

Symbol	Description
S	dataset
n	$\|S\|$, size of the dataset
k	# search results
q	query point
λ	importance of diversity over relevance
R	set of result points
K	set of result points for k-NN search
d	# dimensions
sim_θ	angular similarity of two points regarding q
$\mathrm{DT}(S)$	Delaunay triangulation of points in S
$\mathrm{GG}(S)$	Gabriel graph of points in S
k'	# natural neighbors for q in $S \cup \{q\}$
W	weights w_i of each natural neighbor
$\mathrm{adj}[p]$	adjacent nodes/points of p in a graph
$l_{GG}(k)$	# layers required to obtain k Gabriel neighbors
$\circledast_p^q$	pruning sector from q towards p
$\overrightarrow{qp}$	vector from query point q in the direction of p
ϵ	small fraction to relax $\circledast_p^q$
$\theta_\circledast$	central angle of the pruning sector
$r_\circledast$	radius of the pruning sector
p_i	a point in S
p_{nn}	the nearest neighbor point of q
B_i	an index node in R-tree
$mindist$	minimum distance measure [112]
$rank$	measure that combines $mindist$ and sim_θ
PQ	priority queue
cts	current timestamp
$\mathrm{ts}[Node]$	timestamp of a node in PQ

Definition 2.3.2. λ-diverse k-nearest neighbor search. *Given a set of points S, a query point q, and a diversity ratio λ, the* λ-diverse k-nearest neighbor search *on q retrieves a set of k resulting points $R = \{p_1, \ldots, p_k\}$, such that $R =$*

$$\underset{\substack{R \subseteq S \\ |R| = k}}{\operatorname{argmin}} \left[\alpha \sum_{i=1}^{k} sim(q, p_i) + \frac{2\beta}{k(k-1)} \sum_{i=1}^{k} \sum_{j=i+1}^{k} sim_\theta(p_i, q, p_j) \right] \tag{2.2}$$

where $\alpha = 1 - \lambda$ and $\beta = \lambda$. This function minimizes pairwise angular similarity (depending on λ) and maximizes relevancy (depending on $1 - \lambda$) of the results.

Note that the relative importance of diversity versus relevance is adjusted with the λ parameter. When $\lambda = 0$, the method reduces to k-NN search. For $0.5 \leq \lambda \leq 0.9$, the results are expected to be diverse enough without sacrificing relevancy. As a result, Definition 2.3.2 provides a more flexible and user-tunable setting for diversification of k-NN queries.

The rest of the work includes proposed geometric and index-based methods for efficiently solving λ-diverse k-nearest neighbor search problem. The notation is given in Table 2.1.

2.4 Geometric Diverse Browsing

In the spatial domain, diverse k-nearest neighbor search is conceptually similar to the idea of natural neighbors, which is calculated with Voronoi diagrams (VD) and Delaunay triangulation (DT) [8, 83]. In this section, we first present an analogy of diversity with natural neighbors. Based on the analogy, we propose a natural neighbor-based method along with the techniques that are used to efficiently retrieve natural neighbors of a query point. Although the discussions are mostly on the 2D space, the method can be extended to work on the 3D space as well. Observing the limitations of this method in higher dimensional spaces, we present another geometric approach, namely the Gabriel graph-based diverse browsing method.

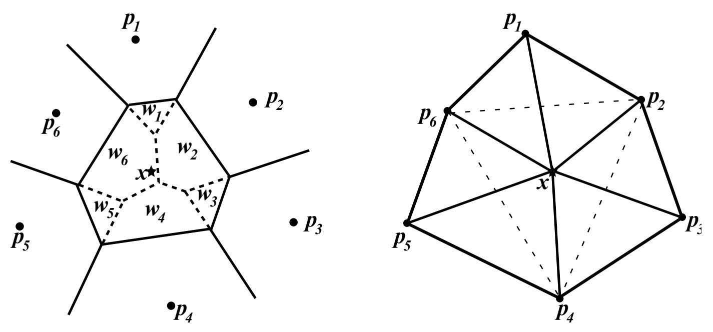


Figure 2.3: Natural neighbor weights w_i of x in 2D (left), DT with and without x (right).

2.4.1 Analogy with natural neighbors

The *natural neighbors* (NatN) of a point $p \in S$ are the points in S sharing an edge with p in DT(S). They also correspond to Voronoi cells that are neighbors of V_p. In case of a point $x \notin S$, its natural neighbors are the points in S whose Voronoi cells would be modified if x is inserted in VD(S). The insertion of x creates a new Voronoi cell V_x^+ that steals volume from the Voronoi cells of its potential natural neighbors (see Fig. 2.3).

To capture the influence of each NatN, we use natural neighbor weights in natural neighbor interpolation [115]. Let D be the VD(S), and $D^+ = D \cup \{x\}$. The Voronoi cell of a point p in D is defined by V_p, and V_p^+ is its cell in D^+. The natural neighbor weight of x with respect to a point p_i is

$$w_i(x) = \frac{\text{Vol}(V_{p_i} \cap V_x^+)}{\text{Vol}(V_x^+)}, \tag{2.3}$$

where Vol(V_{p_i}) represents the volume of V_{p_i}, and $0 \le w_i(x) \le 1$. The natural neighbor weights are affected by both the distance from x to p_i and the spatial distribution of p_i around x.

2.4.2 Natural neighbor-based method

Based on the property of the natural neighbor concept which captures both the distance to a query point and also the spatial distribution around it, we claim that the natural neighbors of a query point q give a diverse set of similarity search results, if the natural neighbor weights W are used as ranking measures. The method works as follows: (1) simulate the insertion of q into DT(S), (2) find the natural neighbors of q: $\{p_1, \ldots, p_{k'}\}$ along with the weights $W = \{w_1, \ldots, w_{k'}\}$, and (3) report results according to the weights in descending order. Details are provided in the following sections;

Offline Generation of Delaunay Triangulation In the preprocessing stage, DT(S) is calculated for all the data points in S. Although the weights are calculated with the overlapping areas of these cells, and natural neighbors are defined (and also easy to understand) in terms of Voronoi cells, performing operations on DT is computationally more efficient.

There are I/O- and memory-efficient methods for building Delaunay triangulation in 2D and 3D [2, 58]. These methods can generate DTs for billions of points. There are also publicly available implementations for 2D [114] and for higher dimensions [10]. We use Qhull implementation [10] to generate DT(S).

Step 1 - Flip-based Incremental Insertion We simulate insertion of q into DT(S) with a flip-based insertion algorithm. It is easy to determine the simplex of DT(S) containing q in linear time by inspecting all triangles.

Let τ be the simplex in DT(S) containing q. All vertices of τ automatically become natural neighbors of q once it is inserted to DT(S). Then, the necessary edge flips are carried out until no further edge needs to be flipped, revealing even more natural neighbors of q.

The number of flips needed to insert q is proportional to the degree of q (the number of incident edges) after its insertion. The average degree of a vertex in a 2D DT is six. This number is proportional to the number of dimensions.

Step 2 - Find NatNs and weights Vertices $\{p_1, \ldots, p_{k'}\}$ adjacent to q in DT($S \cup q$) are the natural neighbors of q. The volume of a d-dimensional Voronoi cell is computed by decomposing it into d-simplices and summing their volumes. The volume of a d-simplex τ with vertices $(v_0, \ldots, v_n)$ [83] is computed as:

$$\mathrm{Vol}(\tau) = \frac{1}{d!} \left| \det \begin{pmatrix} v_1 - v_0 & \cdots & v_n - v_0 \end{pmatrix} \right|, \tag{2.4}$$

where each column of the $n \times n$ determinant is the difference between the vectors representing two vertices. Weights W are then calculated with Eq. 2.3.

Step 3 - Report for Diverse *k*NN NatN-based method naturally returns k' results as an answer to query point q. The result set R is ranked according to the weights of each neighbor. If $k' \geq k$, we report the top-k ranked results. Otherwise, k' points are returned.

Overview of the method is given in Algorithm 1. Note that if the number of natural neighbors is greater than k, the points with smaller weights are eliminated. Otherwise, the method may return less than k results.

2.4.3 Limitations

The drawback of using natural neighbors in diverse k-nearest neighbor search is that there is always a fixed number of natural neighbors of a point, and the number

Algorithm 1: Algorithm NatNDiversitySearch

```
procedure NatNDiversitySearch($q, k$,DT)
DT$'$ $\leftarrow$ Insert(DT, $q$)
$W \leftarrow \{\}$
for each point $p_i$ in $adj[q]$ do
    $w_i \leftarrow$ CalculateWeight(DT$'$, $p_i, q$)
    $W \leftarrow W \cup \{w_i\}$
$W' \leftarrow$ Sort($W$)
if $adj[q] > k$ then
    $W' \leftarrow W'[1:k]$
return $W'.i$
```

is proportional to the number of dimensions. This can even be seen as an advantage since parameter k is not specified by the user; it is inherently captured by the process. For browsing purposes, one cannot restrict the search with only natural neighbor results as the user may demand more search results. The search needs to continue incrementally through the neighbors of neighbors with Voronoi cells. Without any assumptions on the distribution of the data, the average degree of a vertex in a 2D DT is 6 [83]. In this case, diverse k-nearest neighbor search with the NatN-based method for 2D space may not return a result set with k items. As a result the method is forced to investigate the neighbors of neighbors with Voronoi cells which were not modified with the insertion of q.

For higher dimensional spaces, the average degree of a point in DT grows quickly with d (approximately d^d) [38]. The problem of selecting a subset of elements in this set to obtain a diverse set of k items cannot be trivially solved with a NatN-based approach. Because of the disadvantages, NatN-based method is more appropriate for low-dimensional data and small k values.

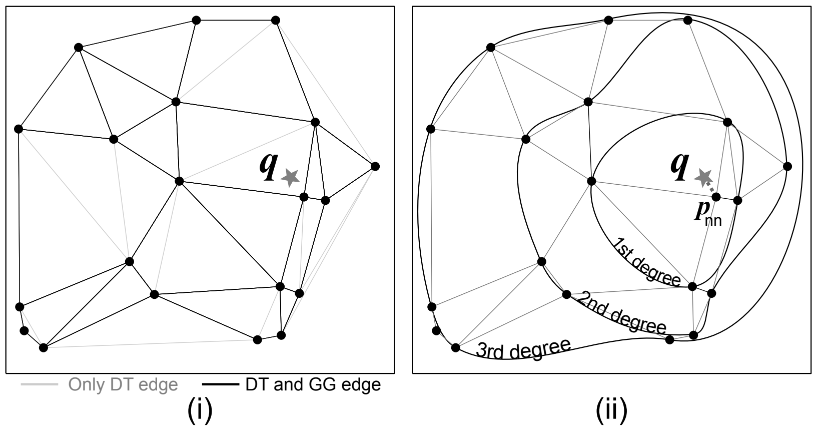


Figure 2.4: Incremental browsing of a Gabriel graph. (i) Delaunay triangulation of the points with Gabriel edges highlighted. (ii) For a query point q, diverse results are gathered layer-by-layer starting with p_{nn}.

2.4.4 Gabriel neighbor-based method

In high dimensions, DT is intractable in terms of both construction complexity $O(n^{\lceil d/2 \rceil})$ and browsing efficiency. For better scalability and browsing capability in high dimensional spaces, we propose using Gabriel graphs (GG) instead of DT.

The Gabriel graph [41] is the set of edges e_{ij} that is a subset of $\text{DT}(S)$, for which the circle with diameter $[p_i p_j]$ contains no other points from S.

$$\text{GG}(S) = \{e_{ij} \subseteq \text{DT}(S) | \, \forall p_k \in S, |p_k p_i|^2 + |p_k p_j|^2 \geq |p_i p_j|^2\} \tag{2.5}$$

Gabriel graph contains those edges of DT that intersect their Voronoi faces [99]. Hence, GG can be constructed in $O(n \log n)$ time by first constructing DT and VD, and then adding each edge in DT to GG if it intersects its Voronoi face. Without DT and VD, GG can always be constructed by brute-force in $O(n^3)$ time.

The advantage of working with GG is that both nearest neighbor graph (NNG) and minimum spanning tree (MST) are subgraphs of it; therefore, GG still captures proximity relationships among data points. Furthermore, GG is reasonably sparse and simple: for planar graphs $|GG(S)| \leq 3n-8$ [99]. As a result, the Gabriel graph is particularly popular in constructing power-efficient topology for wireless and sensor networks [91].

Our solution for diverse k-nearest neighbor search problem is to browse GG layer-by-layer, starting from the nearest point p_{nn} to the query point q. Fig. 2.4 shows an example of GG layers connected with B-spline. For efficiency, the query point is not inserted into GG(S), but rather the spatial location of q is imitated with its nearest neighbor p_{nn}.

After finding p_{nn}, the algorithm iteratively searches the n-degree neighbors of p_{nn} in GG(S), starting with $n=1$. GGDiversitySearch stops when k or more points are included in R. Note that the resulting points are added layer-by-layer; therefore, there is a ranking among layers. However, they are not sorted within layers, since there is no concept similar to natural neighbor weights in Gabriel graphs. In addition, $|R| \geq k$, meaning that the algorithm may return more than k results. The method is given in Algorithm 2.

It is possible to return exactly k results by examining the last layer of points added into R. One method is to choose a subset of points from the last layer, which optimizes the overall diversity of R. We are using a similar approach in the experiments, explained in the next section.

Algorithm 2: Algorithm GGDiversitySearch

```
procedure GGDiversitySearch($q$, $k$,GG, $S$)
$p_{\text{nn}} \leftarrow$ NearestNeighbor(S, $q$)
$R \leftarrow \{\}$
$R' \leftarrow \{p_{\text{nn}}\}$
while $|R| < k$ do
    $R \leftarrow R \cup R'$
    $R'' \leftarrow \{\}$
    for each point $p$ in $R'$ do
        $R'' \leftarrow R' \cup \text{adj}[p]$
    $R' \leftarrow (R'' \setminus R)$
return $R$
```

2.4.5 Optimization of Gabriel graph layers

GG-based method browse GG layer-by-layer, and return $|R| \geq k$ results. When $|R| > k$, it is possible to investigate the elements of the last layer and select a subset of them to make sure that exactly k results are returned. The problem can be defined as follows:

Let q be a query point, and k be the number of results for a diverse k-nearest neighbor search. Suppose R is the Gabriel neighbors of q, inserted layer-by-layer up to the layer $l_{GG}(k)-1$, satisfying $|R| < k$. The problem is to select a set of $k-|R|$ objects from the layer $l_{GG}(k)$ so that the diversity of R' will be maximum. L refers to the last layer to be investigated. Algorithm 3 finds a local maximum for diversity metric of the results.

2.5 Index-based Diverse Browsing

Spatial databases mostly come with an index structure, such as the widely used R-tree [112]. A popular method called *distance browsing* [56] tries to find the k-nearest

Algorithm 3: Algorithm GGOptimizeLastLayer

```
procedure GGOptimizeLastLayer($q, k, R, L$)
$R' \leftarrow R$
$L' \leftarrow L$
while $|R'| < k$ do
    $o \leftarrow \text{argmin}_{obj \in L'} \text{DIV}(R' \cup obj)$
    $R' \leftarrow R' \cup \{o\}$
    $L' \leftarrow L' \setminus \{o\}$
return $R'$
```

neighbors (k-NN) of a point in a spatial database that uses the R-tree index. We introduce *diverse browsing* for the diverse k-nearest neighbor search over an R-tree index.

The principal idea of diverse browsing is to use the distance browsing method with a pruning mechanism that omits non-diverse data points and minimum bounding rectangles (MBRs). A priority queue is maintained with respect to a *rank*, which is a combination of the *mindist* [56] and the angular similarity (Eq. 2.1) for the object *obj* (either a data point or an R-tree index node). In each iteration, the closest object is investigated (see Fig. 2.5).

In the following sections, ranking and pruning mechanisms (2.5.1), correctness of the algorithm (2.5.2), and their integration into our incremental diverse browsing method (2.5.3) are explained. Note that the algorithm is given in two parts (Algorithms 4 and 5).

2.5.1 Pruning and ranking

When a point p is added to the result set R, we draw an imaginary sector $\circledast_p^q$ from the query point q in the direction of p with $\theta_\circledast = 2 \times \theta_s$ angle and $r_\circledast = r_s \times |\overrightarrow{qp}|$. Every

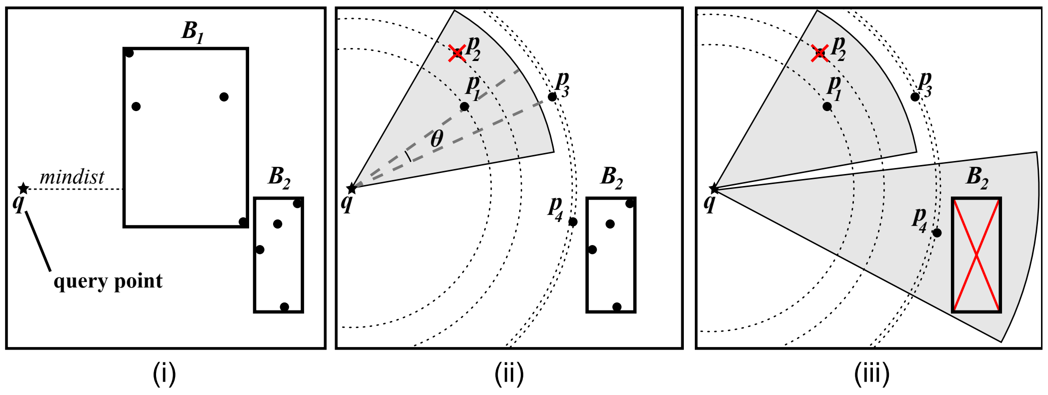


Figure 2.5: Example for diverse browsing. (i) Suppose B_1 and B_2 are two internal nodes of the R-tree index. (ii) When the closest MBR is investigated and the closest point p_1 is added to the result set, p_2 is pruned because of high angular and distance similarity to p_1. The rank of p_3 is also increased here due to its high angular similarity to p_1. (iii) Next, p_4 is added to the result set and causes B_2 to be pruned since none of the items in B_2 can be diverse.

point in this sector will eventually be pruned. By default, $r_s = 1 + \lambda$ and $\theta_s = \frac{2\pi}{k+\epsilon}$, unless specified otherwise.

We use the term *rank* as an alternative to *mindist* in distance browsing. Ranks of points and MBRs in the priority queue are calculated according to the angular similarity and distance with respect to the elements in R (see Algorithm 4). Note that the points with ranks closer to 0 are more likely to be included in the result set.

Points without enough angular diversity and distance from another point in R are pruned. Similarly, the algorithm also prunes MBRs only if none of the corners of the object are diverse enough to be in the result set. The advantage here is that all the pruned data can be displayed as *similar results* of each resulting point with a small modification since we have the information why a point is pruned.

Algorithm 4: Algorithm GetRank

```
procedure GetRank($(q, obj, R, \lambda)$)
Inputs: q: query point, obj: point or rectangle, R: current state of the result set
Inputs: $\lambda$: importance of diversity vs. relevance, $r_s$: pruning radius ratio
$\theta_s \leftarrow \frac{2\pi}{k+\epsilon}$
$r_s \leftarrow 1 + \lambda$
$\delta \leftarrow mindist(q, obj)$
if $obj$ is a point then
    for each point $p$ in $R$ do
        $s[p] \leftarrow \mathrm{sim}_\theta(obj, q, p)$
        if $s[p] > 0$ and $\delta < |\overrightarrow{qp}| \times r_s$ then
            return Prune($(obj)$)
    $rank \leftarrow \lambda \times \max(s) + (1 - \lambda) \times \delta$
else if $obj$ is a rectangle then
    for each point $p$ in $R$ do
        $s \leftarrow \min_{y \in obj.corners}(\mathrm{sim}_\theta(y, q, p))$ if $\forall y \in obj.corners$ in $\circledast_p^q$ then
            return Prune($(obj)$)
        else if $\exists y \in obj.corners$ in $\circledast_p^q$ then
            $\delta \leftarrow min(|\overrightarrow{qp}| \times r_s, |\overrightarrow{qy}|)$
        $rrank[p] \leftarrow \lambda \times s + (1 - \lambda) \times \delta$
    $rank \leftarrow \max(rrank)$
return $rank$
```

2.5.2 Correctness of the algorithm

The efficiency of the proposed method comes from the *diverse browsing* of the R-tree structure. As in incremental nearest neighbor search algorithms and distance browsing [56], a min-priority queue (PQ) is maintained after each operation. However, instead of *mindist* metric we use the result of the GetRank function as the value of each object in PQ. GetRank gives a non-negative value as the rank of a point or an MBR depending on its angular diversity and distance to the query point depending on both R and λ. In each iteration, the object with the lowest rank (top of PQ) is investigated.

As the rank of each object in PQ depends on the current state of R, some of the ranks will be obsolete after another point is inserted into R. But, instead of updating all the objects in PQ (which would be inefficient), we argue to update only the top of PQ with a *timestamp-based* approach until an up-to-date object is acquired. Current timestamp (cts) is incremented every time a point is included in the result set. The proposed method based on timestamp-based update of ranks in PQ is proved by Lemma 2.5.1 and Theorem 2.5.2.

Lemma 2.5.1. *Update operation on an object, which is on top of PQ and has an earlier timestamp, can either increase the rank of the object or does not affect it at all.*

Proof. An object obj is updated only when $\mathrm{ts}[obj] < cts$. Since cts increases when a new item is added to R, there are exactly (cts-$\mathrm{ts}[obj]$) new items in the result set R^+ compared to the time when $rank[obj]$ was calculated.

Suppose that the new rank of obj at the current timestamp is $rank'[obj]$. If obj is a point, three possible outcomes of the update are:

1. $\exists p \in R^+,\ obj$ resides in $\circledast_p^q\ \Rightarrow \textsc{Prune}(obj)$

2. $\exists p \in R^+,\ \mathrm{sim}_\theta(obj, q, p) > \max_{r\in R}(\mathrm{sim}_\theta(obj, q, r)) \Rightarrow rank'[obj] > rank[obj]$

3. $\forall p \in R^+,\ \mathrm{sim}_\theta(obj, q, p) \le \max_{r\in R}(\mathrm{sim}_\theta(obj, q, r)) \Rightarrow rank'[obj] = rank[obj]$

On the other hand, if obj is a leaf or internal node, the rank depends on the corners of the MBR:

1. $\exists p \in R^+,\ \forall y \in obj.corners,\ y$ resides in $\circledast_p^q \Rightarrow\ \textsc{Prune}(obj)$

2. $\exists p \in R^+,\ \exists y \in obj.corners,\ \mathrm{sim}_\theta(y, q, p) > \max_{r \in R}(\mathrm{sim}_\theta(obj, q, r))$

   $\Rightarrow rank'[obj] > rank[obj]$

3. $\forall p \in R^+,\ \exists y \in obj.corners,\ y$ resides in $\circledast_p^q,\ \mathrm{sim}_\theta(y, q, p) \leq \max_{r \in R}(\mathrm{sim}_\theta(obj, q, r))$

   $\Rightarrow rank'[obj] = rank[obj]$

We have shown that some updated objects are pruned. If not, its rank either increases or stays the same. □

Theorem 2.5.2. *An object on top of PQ with the current timestamp provides a lower-bound for the ranks of all objects in PQ, even if there are other objects in PQ with earlier timestamps.*

Proof. Suppose obj is on top of PQ with the current timestamp, and let obj' be another object in PQ with an older timestamp ($ts[obj'] < cts$). Following Lemma 2.5.1, even if the rank of obj' is updated, it is either pruned or $rank^+[obj'] \geq rank[obj']$. Since obj' is not on top of PQ, $rank[obj'] \geq rank[obj]$. Hence $rank^+[obj'] \geq rank[obj]$. Therefore, $rank[obj]$ is still a lower-bound for the ranks of the objects in PQ. □

2.5.3 Incremental diverse browsing

After extending the distance browsing feature of R-trees with diverse choices, incremental browsing of an R-tree gives diverse results depending on λ. Details of the method are given in Algorithm 5, excluding certain boundary conditions, i.e., when $\lambda = 1$, or PQ becomes empty.

The proposed algorithm has the following properties:

Property 5.3: Diverse k-nearest neighbor results obtained by the diverse browsing method always contain p_{nn}, the nearest neighbor of the query point q.

Algorithm 5: Algorithm DiverseKNNSearch

```
procedure DiverseKNNSearch($q, k, \lambda, R\text{-}tree$)
R ← {}
cts ← 0
PQ ← MinPriorityQueue()
Enqueue(PQ, ⟨R-tree.root, cts, 0⟩)
while not IsEmpty(PQ) and |R| < k do
    while ts[Top(PQ)] < cts do
        e ← Dequeue(PQ)
        Enqueue(PQ, ⟨e, cts, GetRank(e)⟩)
    e ← Dequeue(PQ)
    if e is a point then
        R ← R ∪ {e}
        cts ← cts + 1
    else
        for each obj in node e do
            Enqueue(PQ, ⟨obj, cts, GetRank(obj)⟩)
return R
```

Proof. Initially $R = \emptyset$. Therefore the rank of every object $obj \in$ PQ is calculated solely depending on the *mindist* to the query point (see Algorithm 4). The algorithm's behavior is similar to that of the distance browsing method at this stage. When the first point p is dequeued from PQ, it is included to R. p is also the point with the minimum distance to q. Hence, $p_{\text{nn}} \in R$. □

Property 5.4: Diverse browsing can capture the set P_c which comprises k points uniformly distributed around q with the same distances as p_{nn}.

Proof. The method selects the points in P_c without pruning any of them. We guarantee that the points in P_c are retrieved without any assumptions about the order. In addition, $\min \widehat{p_i q p_j} = 2\pi/k$ where $p_i, p_j \in P_c$. Therefore $[2\pi/k] \geq [2\pi/(k+\epsilon)] = \theta_s$. Hence, no point in P_c is pruned. □

Dissimilarity-based diversification methods (e.g., [59]) do not support this property since they are likely to prune some points in P_c, especially when $k > 6$.

2.6 Experiments

We define the evaluation measures in Section 2.6.1. Real and synthetic datasets used in the experiments are summarized in Section 2.6.2. Evaluation and discussion of the methods for spatial and high-dimensional datasets are given in Sections 2.6.3 and 2.6.4.

2.6.1 Evaluation measures

In order to measure how well the methods capture the relevancy and the spatial distribution around the query point, we use the evaluation measures given in Definitions 2.6.1, 2.6.2, and 2.6.3.

Definition 2.6.1. Angular diversity. *Given a query point q and a set of results R, angular diversity measures the spatial diversity around the query point:*

$$DIV(q, R) = 1 - \frac{\left\| \sum_{p_i \in R} \frac{\overrightarrow{qp_i}}{\|\overrightarrow{qp_i}\|} \right\|}{|R|}. \tag{2.6}$$

The intuition behind this measure is that each of the points in R tries to influence the overall result in the direction of the point itself. If the result set is fully diverse, sum of these "forces" will be closer to the center; therefore, the average influence on the query point gives an idea of how diverse the result set is. This measure can easily be applied to higher dimensions since it consists of simple vector additions and normalization. See Fig. 2.6 for the angular diversity of the points in Fig. 2.3.

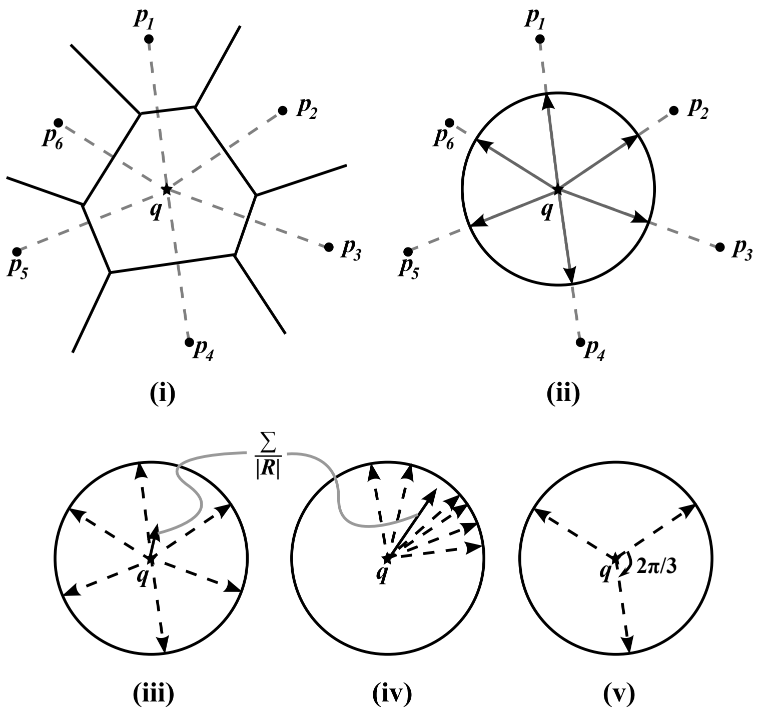


Figure 2.6: Angular diversity measure. (i) Suppose diverse 6-nearest neighbor search for q retrieves $\{p_1, \ldots, p_6\}$. (ii) Angular diversity of the result set is calculated by the sum of vectors $q\vec{p}_i$ on a unit circle/sphere. (iii) When the result set is diverse, the average of these vectors will be close to the center; otherwise, (iv) the average will be close to the circle. (v) For example, maximum angular diversity (DIV=1) in 2D for $k = 3$ can be achieved with points which have an angle of $2\pi/3$ pairwise.

However, the angular diversity measure is not adequate to evaluate the diversity of a result set since an algorithm can always return a better set of items more distant than the nearest neighbors if the distance factor is omitted.

Definition 2.6.2. Relevance. *Given a query point q, its k-nearest neighbors K, and a set of results R, relevance measure calculates the normalized average distance of the points in R with respect to the k-nearest neighbors:*

$$REL(q, K, R) = \frac{\sum_{p_j \in K} \|\overrightarrow{qp_j}\|}{\sum_{p_i \in R} \|\overrightarrow{qp_i}\|}. \tag{2.7}$$

The result of REL is in the interval [0,1]. Magnitude of each vector is calculated in the Euclidean space although any metric distance measure can be applied to the function, as long as the measure is consistent with the one used in calculation of k-NN.

Definition 2.6.3. Diverse-relevance. *Given a query point q, its k-nearest neighbors K, a set of results R, and a parameter λ, diverse-relevance measures both relevancy and angular diversity of the results:*

$$DIVREL(q, K, R) = \lambda \times DIV(q, R) + (1 - \lambda) \times REL(q, R). \tag{2.8}$$

DIVREL is based on the Maximal Marginal Relevance (MMR) method [18]. When $\lambda = 0$, the measure evaluates the relevance of the results excluding the diversity. The aim of our methods is to maximize the diverse-relevance of a result set depending on the value of λ.

2.6.2 Datasets

We conduct our experiments on both real datasets of POIs in 2D and synthetic high-dimensional datasets to evaluate the efficiency and effectiveness of the proposed methods. The properties of our datasets are summarized in Table 2.2.

Real datasets. We use four real-life datasets in our experiments. `ROAD` is the latitudes and longitudes of road crossings in Montgomery County MD. 10% of the `ROAD` dataset is randomly selected as query points. The North East `NE` dataset contains postal addresses from three metropolitan areas (New York, Philadelphia and Boston).[3] The `CAL` dataset consists of points of interest in California.[4] To avoid

[3]http://www.rtreeportal.org

[4]http://www.cs.fsu.edu/∼lifeifei/SpatialDataset.htm

Table 2.2: Properties of datasets

Dataset	d	**Card.**	**Description**
ROAD	2	64K	road crossings, Montgomery ct
NE	2	124K	postal addresses, northeast of US
CAL	2	105K	points of interest in California
USPOI	2	5,8M	POIs in the US, Factual data
NORM	6	500K	normal distribution ($\mu = 0, \sigma^2 = 1$)
UNI	6	500K	uniform distribution
SKEW	6	500K	skew normal dist. (μ=0,σ^2=1,α=1)
HDIM	10	1M	uniform distribution

querying outside of the data region, 500 points from both datasets are randomly selected as queries.

Finally, the `USPOI` dataset is extracted from more than 13M points-of-interest in the US, gathered by Factual.[5] Among those, 8.8M have the location information and 5.8M are unique. We have randomly chosen 1000 POIs as queries.

Synthetic datasets. We generate four synthetic high-dimensional datasets. `NORM` is a 6D dataset generated with normal distribution ($\mu = 0, \sigma^2 = 1$). `UNI` and `HDIM` are 6D and 10D datasets generated with uniform distribution. `SKEW` is generated with skew normal distribution ($\mu = 0, \sigma^2 = 1, \alpha = 1$). For each of these synthetic datasets, 200 query points are produced with the original distribution and parameters.

2.6.3 Results

Real datasets. We compare the results of geometric approaches (NatN-based and GG-based) with diverse browsing of R-tree and KNDN-IG and KNDN-BG [59]

[5]http://www.factual.com

on `ROAD`, `NE`, `CAL`, and `USPOI` datasets. In order to be consistent, the results of the NatN-based method is obtained first. Depending on the number of natural neighbors of each query k', we run other algorithms with $k = k'$. R-trees are built with a page size of 512 bytes (which holds 64 data points) and a fill factor of 0.5. Immediate greedy (IG) and buffered greedy (BG) approaches of the KNDN method are adopted for the spatial domain: the threshold parameter *MinDiv* is set to 0.1, which is also modified according to the value of λ. Note that when $\lambda = 0$, both KNDN and diverse browsing on R-tree methods reduce to k-NN.

Similar results for all real datasets (see Fig. 2.7) indicates that geometric methods naturally produce diverse results in terms of the DIV measure. Since the natural and Gabriel neighbors of a point are fixed in a dataset, these methods are the most efficient ones, only if (1) the purpose of the search is angular diversity, and (2) the spatial database is stable. The GG-based method has an advantage over the NatN-based method while it enables for incremental diverse browsing.

However, life itself is always changing. In most cases a spatial index is used to represent certain points-of-interests, and those databases are updated constantly. Roads are built, new restaurants are opened, old buildings are replaced by new ones. Because geometric methods require a preprocessing time for building the entire DT or GG, they are not appropriate for dynamic databases. In addition, users may want to adjust how diverse vs. relevant the search results should be. Index-based methods provide such flexibility. We will discuss the advantages and disadvantages of each method in Section 2.6.4.

If we focus on index-based search methods, both diverse browsing and KNDN start with DIVREL ≈ 1 when $\lambda = 0$, and they try to adjust their results as the user

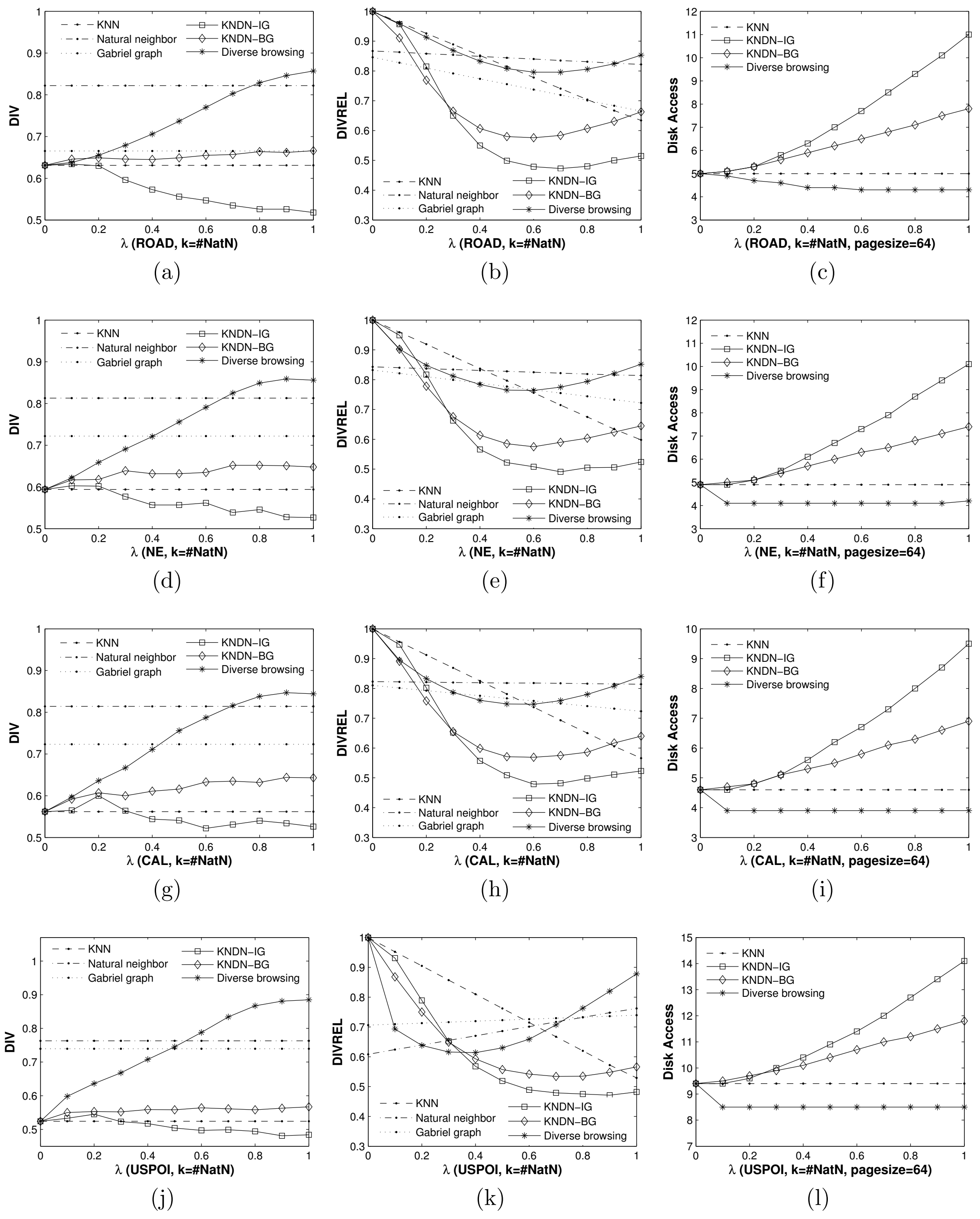


Figure 2.7: Comparison of the algorithms on `ROAD` (a,b,c), `NE` (d,e,f), `CAL` (g,h,i), and `USPOI` (j,k,l) datasets. Aim of the methods is to maximize the diverse-relevance (DIVREL) of the results. Angular diversity (DIV) of the geometric approaches are stable because the natural and Gabriel neighbors of a point are fixed.

asks for more diversity. It is seen that diverse browsing performs better in producing a more diverse result set for the spatial domain compared to KNDN-IG and KNDN-BG (about 20% improvement for $\lambda = 1$, 10% improvement overall). The diverse browsing method also gives a high diverse-relevant set of results as the user seeks diversity in the results (about 15% to 25% improvement).

Synthetic datasets. The purpose of experimenting in high-dimensional space is to show the efficiency of each algorithm and the diverse-relevance of the results. Fig. 2.8 shows the comparison of diverse browsing, GG-based and KNDN methods on synthetic 6D datasets. The NatN-based method is omitted, because it is not scalable to high dimensions due to its high average degree (see Section 2.4.3). In order to measure the efficiency of each index-based method, we spot the page accesses when $\lambda = 1$, for which the algorithms investigate the highest number of internal nodes.

For the queries where relevance is preferred over diversity (i.e., $\lambda < 0.5$), diverse browsing and KNDN-BG perform better than the GG-based method since they are both based on the distance browsing feature of R-trees. On the other hand, the Gabriel graph-based method is extremely powerful for diversity-dominant queries (i.e., $\lambda \geq 0.5$) in terms of both computational efficiency and the diverse-relevance of the results. After retrieving the nearest neighbor p_{nn} in the database, it only takes page accesses equal to the number of layers $l_{GG}(k)$ required to obtain k Gabriel neighbors. From our observations, $l_{GG}(k) \leq 2$ for $k = O(d^2)$. The GG-based method also improves the diverse-relevance of the results up to 25% when $\lambda \geq 0.5$.

The most challenging dataset we experiment on is the `HDIM` dataset. Because generating the GG efficiently in high-dimensions is not the concern of this work, we decided to extract only the necessary Gabriel-edges for this experiment. Fig. 2.8-(j,k)

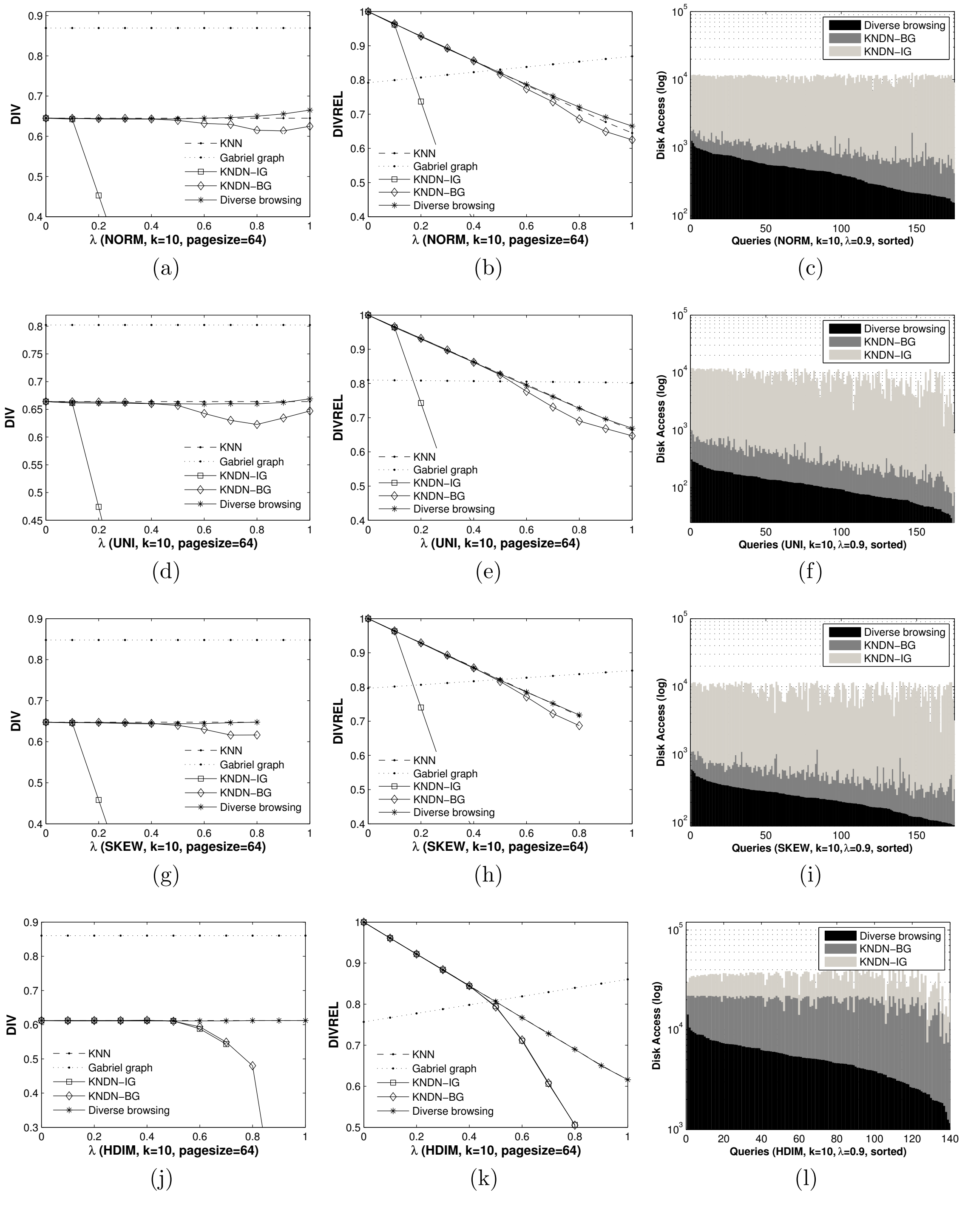


Figure 2.8: Comparison of the algorithms on `NORM` (a,b,c), `UNI` (d,e,f), `SKEW` (g,h,i), `HDIM` (j,k,l) datasets.

suggest that the GG-based method is highly effective for diversity-intended queries, where index-based methods return similar results for different λ values. This is obviously because the Euclidean distance in higher dimensions may not accurately measure the similarity. But still, the diverse browsing method is able to produce the same results for $\lambda \leq 0.5$ as KNDN-IG and KNDN-BG with 36% less page accesses. For $\lambda > 0.5$, diverse browsing is more effective and efficient.

Diverse browsing makes less disk accesses as it successfully prunes out the index nodes that are not diverse with respect to the results found. Hence, it does not make any unnecessary disk accesses. However, both KNDN methods iteratively investigate nearest neighbors to find the next diverse element.

2.6.4 Discussions

Proposed geometric and an index-based diverse browsing methods have their own advantages in terms of preprocessing, querying, flexibility, and scalability. A summary of the proposed methods are given in Table 2.3.

Table 2.3: Comparison of the methods

Method	**Results**	**Ordered**	**Prep.**	**Incremental**
NatN	k'	by w_i	build DT	NO
GG	$\geq k$	by layer	build GG	YES
Diverse Browsing	k	by *rank*	NO	YES

Preprocessing. The advantage of the index-based diverse browsing method is that it does not require any preprocessing and is ready to execute on any spatial

Table 2.4: Diversity (DIV), diverse-relevance (DIVREL) and the number of disk accesses (DA) for the `USPOI` dataset for $\lambda = \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

	KNN			Natural Neigh.		Gabriel Graph	
λ	**DIV**	**DIVREL**	**DA**	**DIV**	**DIVREL**	**DIV**	**DIVREL**
0.5	0.524	0.762	9.4	0.763	0.686	0.739	0.723
0.6	0.524	0.715	9.4	0.763	0.701	0.739	0.726
0.7	0.524	0.667	9.4	0.763	0.717	0.739	0.730
0.8	0.524	0.620	9.4	0.763	0.732	0.739	0.733
0.9	0.524	0.572	9.4	0.763	0.748	0.739	0.736
1.0	0.524	0.529	9.4	0.763	0.762	0.739	0.739
AVG	0.524	0.644	9.4	0.763	0.724	0.739	0.731

	KNDN-IG			KNDN-BG			Diverse Browsing		
λ	**DIV**	**DIVREL**	**DA**	**DIV**	**DIVREL**	**DA**	**DIV**	**DIVREL**	**DA**
0.5	0.504	0.519	10.9	0.558	0.557	10.4	0.745	0.630	8.5
0.6	0.497	0.489	11.5	0.564	0.542	10.7	0.788	0.659	8.5
0.7	0.499	0.479	12.2	0.561	0.534	11.0	0.834	0.707	8.5
0.8	0.494	0.475	12.9	0.558	0.535	11.3	0.867	0.763	8.5
0.9	0.481	0.471	13.6	0.563	0.548	11.6	0.881	0.820	8.5
1.0	0.484	0.482	14.3	0.567	0.566	11.9	0.885	0.878	8.5
AVG	0.493	0.486	12.6	0.562	0.547	11.1	0.833	0.742	8.5

database that use data partitioning, such as R-tree, R*-tree. On the other hand, geometric methods require to build DT or GG, which can be very complex depending on dimensionality and cardinality. As a result, we suggest index-based diverse browsing for a dynamic database, which is more likely to be based an index that handles insert, delete and update operations efficiently; whereas geometric methods for static databases, which would not cause DT and GG to be frequently calculated.

Querying. As mentioned before, the NatN-based method naturally returns a result set with a fixed number of points. If the user does not specify k and the purpose is to find a perfectly balanced diverse and relevant set of results (see DIVREL graphs at $\lambda \approx 0.5$), the NatN-based method is appropriate. However, diverse browsing is

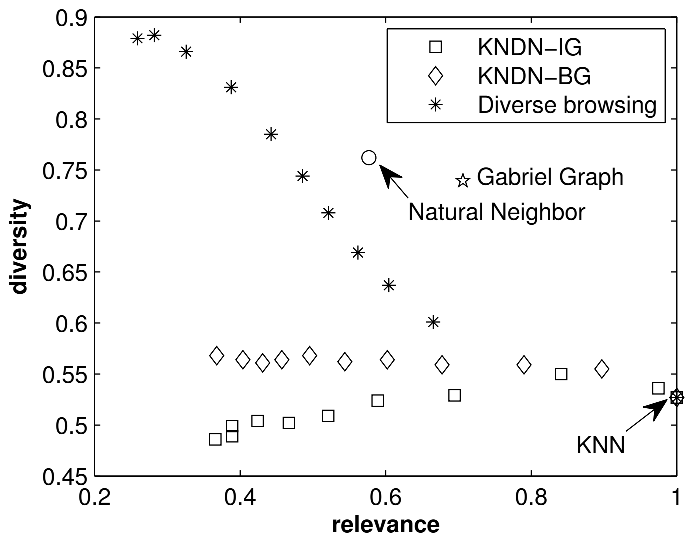


Figure 2.9: Relevance vs. diversity for `USPOI` dataset. Each point corresponds to the average values of DIV and REL metrics for a run with a different λ ($\lim_{\lambda \to 0} REL \to 1$ for all index-based methods).

more suitable for diverse k-NN search, which requires exactly k results to be returned. If the query asks for at least k results, the GG-based method can be used as well.

Flexibility. We can investigate this property in two different ways. The first is the flexibility of setting the importance of diversity over relevance. Only diverse browsing method adjusts itself for various λ values, since the natural and Gabriel neighbors are fixed in a graph. Also note that among the index-based methods, only diverse browsing can provide diverse results when the user is willing to sacrifice relevancy (see Fig. 2.9). The second is the flexibility of incremental diverse browsing, where the user demands more search results. Both index-based diverse browsing and GG-based methods enable the retrieval of additional diverse results.

Scalability. For high dimensional spaces, the NatN-based method is intractable (see Section 2.4.3). Since data partitioning methods are shown to be inefficient for high dimensional data, the GG-based method can be preferred over index-based diverse browsing. Experiments (see Fig. 2.8-(j,k)) show that the GG-based method is in fact very efficient ($\sim$10K vs. $l_{GG}(k)$ page accesses) and effective (0.6 vs. 0.85 DIVREL for $\lambda = 1$) in high dimensional datasets.

2.7 Summary

In this work, we investigate the diversification problem in multi-dimensional nearest neighbor search. Because diverse k-nearest neighbor search is conceptually similar to the idea of natural neighbors, we give a definition of *diversity* by making an analogy with the concept of natural neighbors and propose a natural neighbor-based method. Observing the limitations of NatN-based method in higher dimensional spaces, we present a Gabriel graph-based method that scales well with dimensionality. We also introduce an index-based diverse browsing method, which maintains a priority queue with the ranks of the objects depending on both relevancy and diversity, and efficiently prunes non-diverse items and nodes in order to efficiently get the diverse nearest neighbors. To evaluate the diversity of a given result set to a query point, a measure that captures both the relevancy and angular diversity is presented. We experiment on spatial and multi-dimensional, real and synthetic datasets to observe the efficiency and effectiveness of proposed methods, and compare with index-based techniques found in the literature.

Results suggest that geometric approaches are suitable for static data, and index-based diverse browsing is for dynamic databases. Our index-based diverse browsing method performed more efficient than k-NN search with distance browsing on R-tree (in terms of the number of disk accesses) and more effective than other methods found in the literature (in terms of MMR). In addition, Gabriel graph-based method performed well in high dimensions, which can be investigated more and applied to other research fields where search in high dimensional space is required. Since there are numerous application areas of diverse k-nearest neighbor search, we plan to extend our method to work with different types of data and distance metrics.

Chapter 3: Sentiment Analysis and Opinion Diversification

Increasing popularity of personal web content via blogs, tweets, and other types of social media has boosted the generation of public **opinions** on certain famous people (politicians, singers, etc.), locations (cities, countries, etc.) and brands (products, companies, organizations, etc.). Search and retrieval of opinions on a subject is extremely useful for a number of application areas, including reputation management. However, due to the fact that there might be enormous number of opinions on a subject, or the opinions are skewed towards a viewpoint in some cases, the task of sampling and ranking of the opinions by representing all the viewpoints is a complex task.

In this chapter, we start our analysis by looking at direct correlations between sentiments of opinions and the demographics of people (e.g., gender, age, education level, etc.) that generate those opinions. The details of a large-scale sentiment analysis on Yahoo! Answers data is given in Section 3.1. Since the availability of a topical classification makes it possible to differentiate sentiments attached to a particular entity according to the context, we investigate this property to obtain a more faceted representation of the opinions about an entity. Based on the hypothesis of *network diversity is positively associated with receiving more diverse and less redundant information*, we argue that **opinion diversity** can also be achieved by diversifying the sources of opinions, which is basically finding a set of opinions from various opinion holders with diverse demographics. We discuss the outlines of such an opinion diversification framework in Section 3.2, and leave the practical application as a future work for those who have access to user profiles of social networks.

3.1 Sentiment Analysis on a Large Scale Q&A Website

Sentiment extraction from online web documents has recently been an active research topic due to its potential use in commercial applications. By sentiment analysis, we refer to the problem of assigning a quantitative positive/negative mood to a short bit of text. Most studies in this area are limited to the identification of sentiments and do not investigate the interplay between sentiments and other factors. In this work, we use a sentiment extraction tool to investigate the influence of factors such as gender, age, education level, the topic at hand, or even the time of the day on sentiments in the context of a large online question answering site. We start our analysis by looking at direct correlations, e.g., we observe more positive sentiments on weekends, very neutral ones in the Science & Mathematics topic, a trend for younger people to express stronger sentiments, or people in military bases to ask the most neutral questions. We then extend this basic analysis by investigating how properties of the (asker, answerer) pair affect the sentiment present in the answer. Among other things, we observe a dependence on the pairing of some inferred attributes estimated by a user's ZIP code. We also show that the best answers differ in their sentiments from other answers, e.g., in the Business & Finance topic, best answers tend to have a more neutral sentiment than other answers. Finally, we report results for the task of predicting the attitude that a question will provoke in answers. We believe that understanding factors influencing the mood of users is not only interesting from a sociological point of view, but also has applications in advertising, recommendation, and search.

3.1.1 Introduction

The advance of Web 2.0 boosted the creation of personal web content involving sentiments, e.g., blogs, tweets, and other types of social media. Extraction and analysis of sentiments in this type of content do not only give an emotional snapshot of the online world but also have potential applications in electronic commerce, where the marketing strategy of a product might depend on the mood of the customer. Given both the sociological and financial motivations to understand sentiments, a large body of research has recently investigated the issues involved in sentiment analysis [105].

Despite the intense interest in sentiment analysis, however, relatively little has been done to understand the interplay between sentiments and other factors. In this work, we take the first step in this direction. In particular, we use a state-of-the-art sentiment extraction tool [123] to extract sentiments from a very large sample of questions and answers found in Yahoo! Answers.[6] Our sample provides a rich source for sentiments and also has rich meta-data, including demographic details of users and their degree of experience in the system. To facilitate our analysis, we introduce the metrics of attitude and sentimentality, which enable us to quantify the direction (i.e., positive or negative) and strength of sentiments, respectively.

[6]http://answers.yahoo.com

Our analysis starts with a large-scale study on the correlation of various features with the observed attitude and sentimentality. We investigate textual, topical, demographical, spatial, and temporal features. We then take the analysis one step beyond and answer questions of the following kind:

- Topical context: Do sentiments depend on the context? For example, is there a difference in the attitude of answers related to Tiger Woods in the context of news and events versus the context of sports?
- Class interaction: Who answers how to whom? How do age groups differ in their answers to each other?
- Experience level: How does one's experience in Yahoo! Answers affect the expressed sentiments? Does high experience imply more positive attitude?
- Emotional congruence: How do one's own sentiments correlate with prior sentiments of others? Does a question with a strong sentimentality attract answers with a similar degree of sentimentality?
- Answer preference: How do sentiments within an answer influence an asker's preference? Do people tend to select positive or neutral answers as best answers?

In addition to seeking answers to such questions, we briefly elaborate on the predictability of sentiments. In particular, we build a machine learning model to predict the attitude that will be generated in response to a given question.

Some selected findings of our work are the following:

- There is a strong dependency on the topic. Topics such as Beauty & Style attract strong and generally positive sentiments, whereas Science & Mathematics attracts answers of low sentimentality.

- Demographic factors suggest a strong influence in our data, with women generally expressing stronger, more positive sentiments than men, young people being more positive than older people, and people from predominantly black neighborhoods expressing relatively more neutral sentiments. We also observe a trend for more educated people to give less sentimental answers.
- Sentiments show temporal variation. At a monthly level, the most positive sentiments are observed both during the summer and December. At a daily level, the most positive sentiments are expressed on Saturday and Sunday. At an hourly level, the attitude is at its lowest at around five in the morning.
- People have stronger tendency to give neutral answers as they gain more experience in the online world.
- Best answers differ significantly from other answers in terms of expressed sentiments with more neutral answers being preferred in Business & Finance and more positive ones in News & Events.

The rest of this work is organized as follows. In Section 3.1.2, we provide some information about the Yahoo! Answers data used in our study. Section 3.1.3 summarizes the framework adopted for sentiment analysis. Potential caveats of our study are discussed in Section 3.1.4. We investigate the correlation between the sentiments and features extracted from the data in Section 3.1.5. In Section 3.1.6, we conduct various analyses involving sentiments. Section 3.1.7, as a representative prediction task, explores the predictability of the attitude a question will provoke in its answers. We survey the related work in Section 3.1.8. Finally, Section **??** gives the conclusions with a brief discussion of potential future work.

3.1.2 Yahoo! Answers

Background

Yahoo! Answers is the largest collaborative question answering site in the Web. People ask questions on different topics and share their knowledge, opinions, and personal experiences as answers to these questions. Questions are manually classified by askers into topics so that answerers can easily find them. Answerers can find questions by searching or browsing through a fixed hierarchy of categories.

Every question goes through a best answer selection process. A question remains open during four days for others to answer. The duration of the process can be shortened or extended by the asker of the question. The asker has the option to select a best answer, starting from one hour after the first answer is received, or he can leave the decision to the community vote. If he does neither and when there is only a single answer, then the system automatically selects the best answer after a certain time. Answerers whose answers are selected as the best answer gain experience points, which provide a motivation for answering others' questions.

Dataset Characteristics

We use a sample set of questions and answers posted in Yahoo! Answers during a 12-month period, from October 2009 to September 2010. The sample is restricted to posts originating from the US and contains 34M questions, 132M answers, and 412M sentences. In this sample, about 2.4M users have participated as either an asker or an answerer. We were able to obtain self-provided demographic information for about 1.5M users, of which 54.5% are females and 45.5% are males. The users in our sample are mainly young people, whose ages vary between 15 and 30 (Fig. 3.1 (left)).

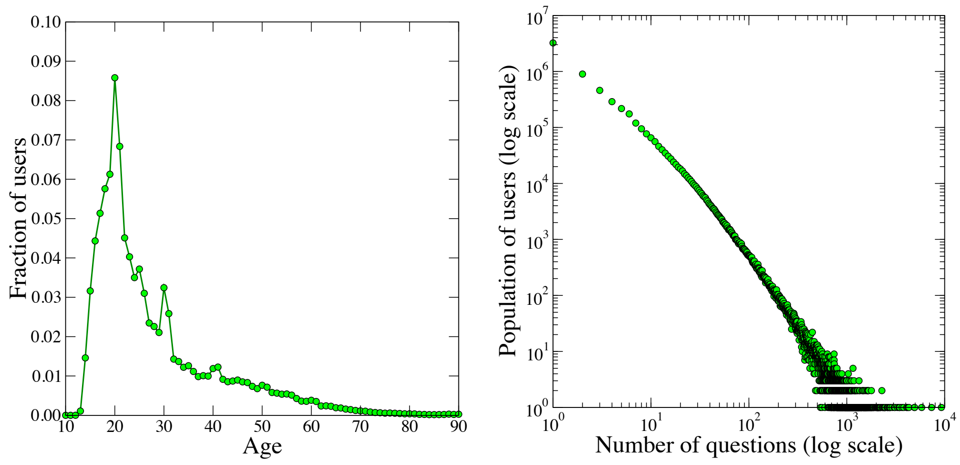


Figure 3.1: Fraction of user population with a certain age (left), population of users who posted a certain number of questions (right).

In our data, a user posts 5.5 questions and 32.7 answers, on average.[7] Populations of users who posted a specific number of questions or answers both follow a power law distribution, as shown in Figs. 3.1 (right) and 3.2 (left), respectively. In Fig. 3.2 (left), it is interesting to note the sudden jump at 20 answers, due to many users aiming to post at least 20 answers since a new level is gained at this point. As seen in Fig. 3.2 (right), the distribution of the number of answers a question receives is also highly skewed (on average, 13.3 answers, excluding questions without any answer).

In Yahoo! Answers, there are 1676 editorially defined categories, 26 of which are top-level categories (e.g., Computers & Internet, Politics & Government). The rest are either second-level or third-level subcategories. A question is labeled with the lowest-level category selected by its asker.

[7]A user can provide only one answer to the same question.

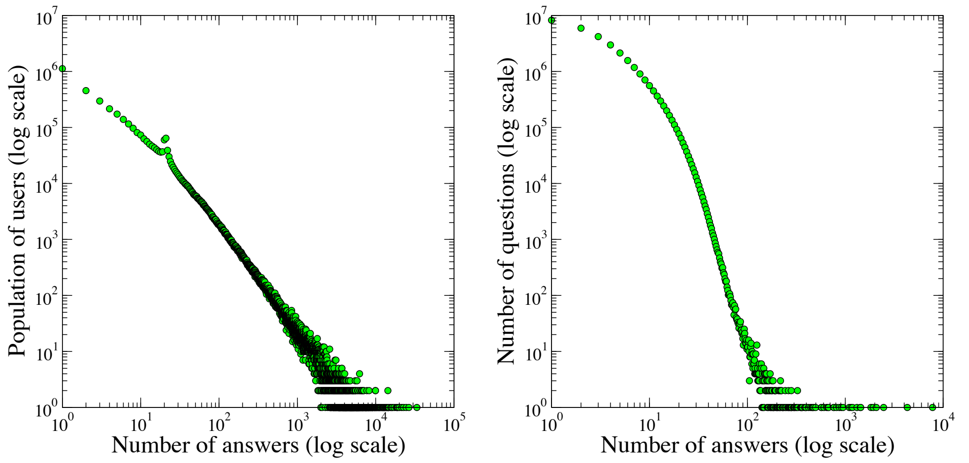


Figure 3.2: Population of users who posted a certain number of answers (left), number of questions receiving a certain number of answers (right).

3.1.3 Analysis framework

Our analysis framework involves the following steps. We first process our sample data to extract questions posted in Yahoo! Answers and their corresponding answers. We then obtain the demographics information of users who posted these questions and answers. Within these two steps, we also extract a number of features that facilitate our analysis. Next, we compute the sentiment scores for individual sentences in the posts, using a state-of-the-art sentiment analysis software as a black box [123]. Based on the sentiment scores obtained at the sentence level, we compute two metrics, referred to as attitude and sentimentality, for different granularities of text. The details of these steps are provided in the rest of this section.

Question and Answer Extraction

In our framework, a question is represented by the sentences in the title section of the question,[8] i.e., the sentences in the abstract section, which provides more details about the question, are ignored. Together with the question text, we also extract some features related to the question (e.g., question length, category, time/date, asker's current experience, and ZIP code). Since the adopted tools cannot handle non-English text, we omit questions which are submitted to a frontend whose language is not English.

We represent an answer by the set of sentences it contains. To split answers into sentences, we use the Stanford parser.[9] Very short (less than 5 characters) or long sentences (more than 400 characters) are ignored. Since the language of the frontend is not always present in our data, we require that either the language is present and set to English or that the location is present and set to the US. Only answers given in response to questions submitted to an English frontend are considered. During the answer extraction process, we also extract some features related to answers (e.g., time/date, answerer's current experience, and ZIP code).

Demographics Extraction

To post either a question or an answer on Yahoo! Answers, a user must have an account and be logged in. Although initially Yahoo!-Answers-only accounts existed, for several years, a general Yahoo! account has been required. For these accounts, we obtain self-provided registration information, which includes birth year, gender, ZIP code, and country.

[8]Typically, the title contains a single question sentence.

[9]`http://nlp.stanford.edu/software/lex-parser.shtml`

Table 3.1: An example question and the first two sentences of the answer

Label	Text	Scores
Q1.1	Is Scotland a good place to live and start a small business?	+2/−1
A1.1	Yes, Edinburgh is always in the top 5 places to live in the UK and usually at the top.	+1/−1
A1.2	One problem regarding setting up a business is the competition is fierce and Edinburgh people are unusually highly qualified.	+1/−2

For users with an existing US ZIP code, we obtain demographic estimates of their income (as quantified by the annual per-capita income), their education level (as quantified by the fraction of the population holding a bachelor's degree or higher), and even their race via the 2000 Governmental Census Data,[10] using the same approach as in [134]. We classify ZIP codes and their corresponding users under the White, Black, and Asian classes[11] if the fraction of the corresponding race in that ZIP code surpasses 50%. In all plots and tables, we always report the results for the largest suitable user population. For example, if a user does not have a valid ZIP code, we do not involve the user in experiments about income, but the user contributes to statistics about the age distribution.

[10] http://factfinder.census.gov/

[11] We use the terminology used in the US census.

Sentiment Score Computation

To assign numerical scores to sentiments of an individual sentence, we use the SentiStrength[12] tool developed by Thelwall et al. [123]. This tool simultaneously assigns both a positive and a negative score to bits of English text, the idea being that users can express both types of sentiments at the same time, such as in "I love you, but I also hate you". Positive sentiment strength scores range from $+1$ (not positive) to $+5$ (extremely positive). Similarly, negative sentiment strength scores range from -1 to -5. The tool works by assigning scores to tokens in a dictionary which includes common emoticons. For example, "love" is mapped to $+3/-1$ and "stink" is mapped to $+1/-3$. Modifier words or symbols can boost the score such that "really love" is mapped to $+4/-1$ (the same for "love!!" or "looove"). The final positive sentiment strength for a bit of text is then computed by taking the maximum score among all individual positive scores. The negative sentiment strength is similarly calculated. Table 3.1 gives an example of a question, its answer, and the corresponding sentiment scores.

Fig. 3.13 (left) shows the distribution of sentiment scores given to sentences in answers. The vast majority of sentences are assigned a neutral $+1/-1$ sentiment score (58.26%). Slightly negative ($+1/-2$) and slightly positive ($+2/-1$) scores are also common (6.04% and 15.87%, respectively). Sentences with very strong sentiments, having either a positive score of $+4$ or higher or a negative score of -4 or lower, make up merely 4.18% of the total sentence volume.

[12]http://sentistrength.wlv.ac.uk/

Table 3.2: The formulas used for computing the attitude and sentimentality metrics

		Metrics	
Input type	Symbol	Attitude	Sentimentality
Question	q_j	$\phi^q(q_j) = \varphi^+(q_j) + \varphi^-(q_j)$	$\psi^q(q_j) = \varphi^+(q_j) - \varphi^-(q_j) - 2$
Asker	K_i	$\Phi^K(K_i) = \frac{1}{\|\mathcal{Q}^i\|} \sum_{q_j \in \mathcal{Q}^i} \phi^q(q_j)$	$\Psi^K(K_i) = \frac{1}{\|\mathcal{Q}^i\|} \sum_{q_j \in \mathcal{Q}^i} \psi^q(q_j)$
Sentence in an answer	s_ℓ	$\phi^s(s_\ell) = \varphi^+(s_\ell) + \varphi^-(s_\ell)$	$\psi^s(s_\ell) = \varphi^+(s_\ell) - \varphi^-(s_\ell) - 2$
Answer to a question	A_k	$\Phi^A(A_k) = \frac{1}{\|\mathcal{S}^k\|} \sum_{s_\ell \in \mathcal{S}^k} \phi^s(s_\ell)$	$\Psi^A(A_k) = \frac{1}{\|\mathcal{S}^k\|} \sum_{s_\ell \in \mathcal{S}^k} \psi^s(s_\ell)$
Answerer	R_i	$\Phi^R(R_i) = \frac{1}{\|\mathcal{P}^i\|} \sum_{A_k \in \mathcal{P}^i} \Phi^A(A_k)$	$\Psi^R(R_i) = \frac{1}{\|\mathcal{P}^i\|} \sum_{A_k \in \mathcal{P}^i} \Psi^A(A_k)$
Answer set of a question	$\mathcal{A}^j$	$\Phi^{\mathcal{A}}(\mathcal{A}^j) = \frac{1}{\|\mathcal{A}^j\|} \sum_{A_k \in \mathcal{A}^j} \Phi^A(A_k)$	$\Psi^{\mathcal{A}}(\mathcal{A}^j) = \frac{1}{\|\mathcal{A}^j\|} \sum_{A_k \in \mathcal{A}^j} \Psi^A(A_k)$

Metrics: Attitude and Sentimentality

Before introducing the metrics of attitude and sentimentality, we introduce some notation. We use q_j, s_ℓ, A_k, $\mathcal{A}^j$, K_i, and R_i to represent the basic types in our data: a question, a particular sentence within an answer, an answer given to a question, the set of answers given to question q_j, an asker, and an answerer, respectively. We also use notation $\mathcal{S}^k$, $\mathcal{Q}^i$, and $\mathcal{P}^i$ to denote the set of sentences in A_k, the set of questions posted by asker K_i, and the set of answers posted by answerer R_i, respectively. Positive and negative sentiment scores generated by the sentiment analysis software for a given question q_j are denoted by $\varphi^+(q_j)$ and $\varphi^-(q_j)$. Similarly, $\varphi^+(s_\ell)$ and $\varphi^-(s_\ell)$ denote the positive and negative sentiment scores for a given sentence s_ℓ.

Based on this notation, we now define the attitude and sentimentality metrics. The attitude metric computes the inclination towards positive or negative sentiments. The sentimentality metric computes the amount of sentiments. In a sense, the former metric indicates the sign of sentiments while the latter indicates their magnitude. These metrics are computed by the formulas shown in Table 3.2, for different fields of the data: question, asker, sentence in an answer, answer, answerer, and set of answers

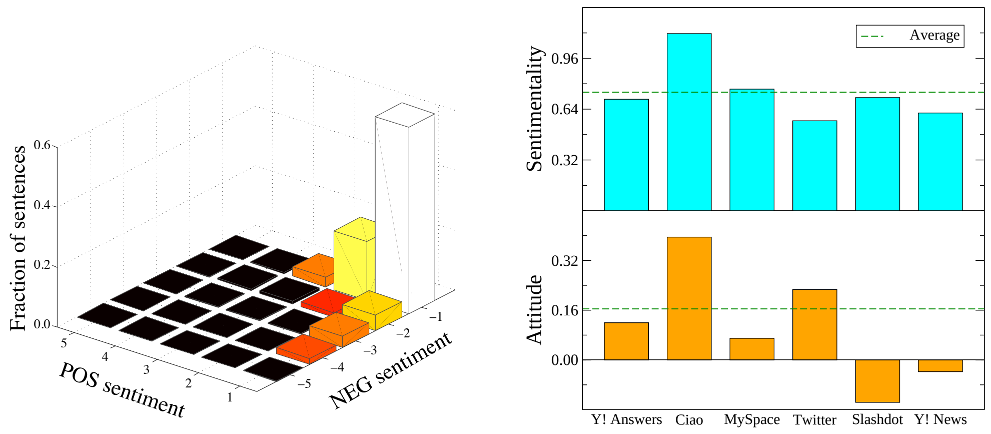


Figure 3.3: Distribution of sentiment scores for sentences in answers (left), average attitude and sentimentality for different web datasets (right).

to a question. All results in the rest of the work are reported by averaging the metrics shown in Table 3.2. At the beginning of each section, we indicate the specific formula we used in the experiments related to that section.

3.1.4 Caveats

Inferring Demographics

Certain online user attributes (e.g., race) can be inferred by aggregating real-life data, obtained from online Web resources. In our work, we aggregate the online data of people living in a region with a specific ZIP code to infer certain attributes of Yahoo! Answers users who provided the same ZIP code. Obviously, this kind of an inference can be quite noisy, especially if the demographical classes within a ZIP code are uniformly distributed (e.g., almost equal number of males and females in every ZIP code) or if the distribution of classes across the ZIP codes is highly skewed (e.g., very few regions where the Asians are the dominant race). We believe that the education level and income attributes, which we infer based on ZIP codes, are relatively less error-prone in this respect as the distribution of these attributes are typically neither uniform within a ZIP code nor very skewed across different ZIP codes.

Sentiment Analysis

We use sentiment analysis as the main technique to quantify the attitude and sentimentality. Obviously, there is no perfect sentiment analysis tool to date. The tool we use indeed performs a simple syntactical analysis over sentences rather than a sophisticated semantic analysis. However, we still hope the erroneous cases to be minimal as the accuracy of the tool has been shown to be good enough in another domain [123]. Moreover, the vast amount of data we use helps supporting the significance of the reported results.

User Bias

We emphasize that our results cannot be generalized to the entire population of the world as the user sample we have in our data may have a biased distribution. In particular, certain demographic classes may be under represented in the Internet. Even if they are represented in the Internet with similar likelihood, their distribution in Yahoo! Answers may be skewed. Finally, although they might be equally well presented in Yahoo! Answers, the rate at which they contribute to the posts may differ. Consequently, our findings are limited to the Yahoo! Answers users who actively participate in the questions and answers.

Site Bias

The generalizability of our results to other web sites may be questioned. To reveal any potential bias, we compute the attitude and sentimentality metrics over six web datasets,[13] each with different characteristics. The selected datasets include questions and answers from Yahoo! Answers, movie reviews from Ciao, forum discussions from MySpace, short messages from Twitter, comments on news-related posts from Slashdot, and news in English from Yahoo! News. For this experiment, we randomly sample 40K sentences from each dataset and compute the attitude and sentimentality metrics by averaging the respective scores over all sentences.

According to Fig. 3.13 (right), the Ciao dataset has the most positive attitude and a relatively higher sentimentality. The most negative attitude is observed in the Slashdot dataset. The Yahoo! Answers dataset stands very close to the average over all datasets, in terms of both the attitude and sentimentality metrics. This observation further justifies our use of Yahoo! Answers as a representative data source.

3.1.5 Feature analysis

We group the features used in our analysis under five headings: textual, topical, demographical, spatial, and temporal features (Table 3.3). All features are extracted from both questions and answers, except for the textual features, which are only extracted from questions. The rest of the section investigates the correlation of the extracted features with the previously defined attitude and sentimentality metrics. Due to space constraints, a detailed analysis of certain features is omitted. We note

[13]http://caw2.barcelonamedia.org/node/7

Table 3.3: Summary of extracted features

Type	Feature	Range	Applicable to Question	Applicable to Answer
Textual	Question length	$\mathbb{N}$	Yes	No
	# of ? symbols	$\mathbb{N}$		
	# of ! symbols	$\mathbb{N}$		
	First word	String		
Topical	Top category	1–26	Yes	Yes
	Leaf category	1–1676		
Demog.	Gender	$\{M, F\}$	Yes	Yes
	Age	$\mathbb{N}$		
	Race	$\{A, B, W\}$		
	Income	$\mathbb{R}^+$		
	Educ. degree	0-100%		
	Experience	$\mathbb{N}$		
Spatial	ZIP code	00000–99999	Yes	Yes
Temporal	Month	1–12	Yes	Yes
	Day of month	1–31		
	Day of week	1–7		
	Hour of day	1–24		

that, in all results presented in this work, we report only the findings that are statistically significant according to a two-tailed t-test for equality of means at the 95% confidence level. Similarly, we only display results where all of the three pairs (maximum, median), (maximum, minimum), and (median, minimum) differ at the 95% confidence level or above. This ensures that the reported results are "meaningful" and not merely noise.

Textual Features

The results reported in this section are obtained by averaging the ϕ^q and ψ^q values over all questions.

Question length. The sentimentality correlates positively with the question length in words (Fig. 3.4 (left)). This is somewhat expected as the probability of

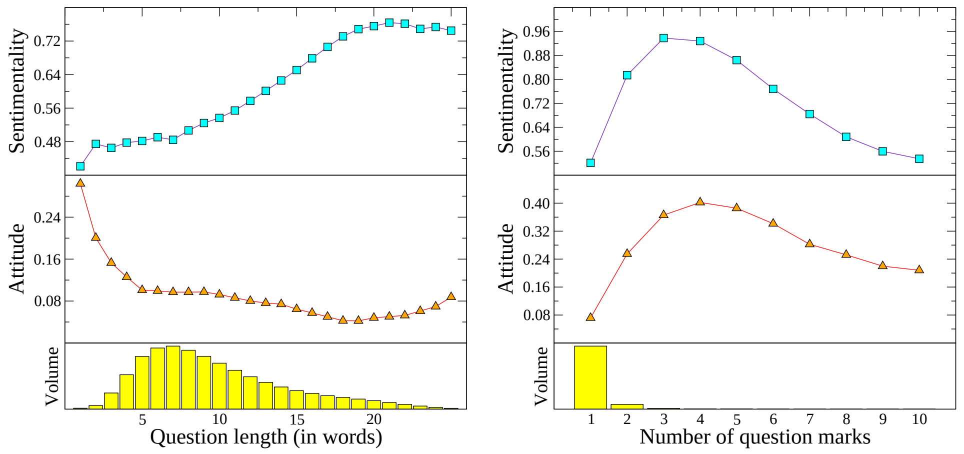


Figure 3.4: Attitude and sentimentality as the question's length (left) and the number of question marks it contains (right) increases.

having sentimental words increases as new words are added to the question.[14] Interestingly, however, the attitude demonstrates a decreasing trend. This might suggest that shorter questions are more likely to contain positive sentiments and negative sentiments are more common in longer questions.

Number of question marks. We observe relatively higher sentimentality and attitude in questions that contain multiple question marks (Fig. 3.4 (right)). Both metrics reach a maximum value when there are three or four question marks in the question. This increase is simply because of question titles that contain multiple question sentences (remember that we do not split questions into sentences). Interestingly, however, the further increase in the number of question marks results in a decrease in the metrics.

[14]Recall that the sentiment scores of a sentence are cumulated by taking the maximum over the words in the sentence. Hence, scores are not normalized by the sentence length.

Table 3.4: Attitude and sentimentality for the most popular question starting words

		Questions		Answers	
Word	Volume (%)	ϕ^q	ψ^q	$\Phi^{\mathcal{A}}$	$\Psi^{\mathcal{A}}$
What	13.2	0.16	0.45	0.21	0.68
How	10.8	0.04	0.36	0.16	0.60
I	5.6	0.07	0.69	0.17	0.70
Is	4.9	0.05	0.62	0.14	0.69
Why	4.4	-0.17	0.66	-0.01	0.75
Can	3.4	0.10	0.38	0.14	0.59
Do	3.1	0.15	0.67	0.18	0.74
Does	2.2	0.11	0.55	0.18	0.69
Where	2.1	0.13	0.28	0.31	0.55
My	2.0	-0.09	0.70	0.07	0.70

Starting words. Table 3.4 shows the attitude and sentimentality of questions with the most popular starting words, listed in decreasing order of popularity. According to the table, the only words with negative attitude are "Why" and "My" while the rest imply mostly positive attitude. In terms of sentimentality, questions starting with words "What", "How", "Can", and "Where" have the lowest sentimentality. In general, we observe that questions seeking information have lower sentimentality than those asking for opinions.

Topical Features

The results reported in this section are obtained by averaging the ϕ^q and ψ^q values of questions and also by averaging the ϕ^A and ψ^A values of answers under specific categories.

Category. Fig. 3.5 shows the distribution of questions posted in the 16 most popular top-level and leaf-level categories. Table 3.5 shows the categories with the

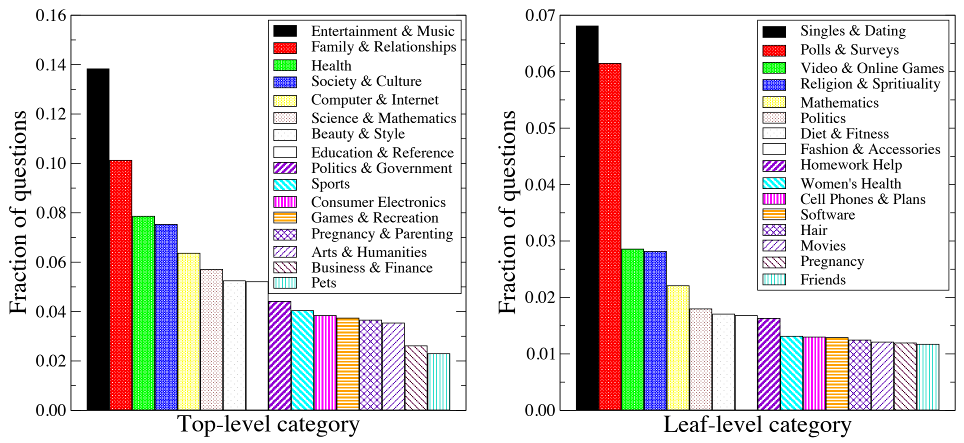


Figure 3.5: Most popular top-level (left) and leaf-level (right) categories.

Table 3.5: Highest and lowest attitude values for top-level and leaf-level categories

	Top-level category				Leaf-level category			
	Questions		Answers		Questions		Answers	
	Category	ϕ^q	Category	$\Phi^{\mathcal{A}}$	Category	ϕ^q	Category	$\Phi^{\mathcal{A}}$
Highest	Beauty & Style	0.28	Beauty & Style	0.43	Baby Names	0.46	Baby Names	0.61
	Dining Out	0.24	Business & Finance	0.30	Beauty & Style	0.44	Valentine's Day	0.54
	Food & Drink	0.21	Dining Out	0.24	Valentine's Day	0.42	Makeup	0.51
	Arts & Humanities	0.20	Pregnancy & Parenting	0.24	Rock and Pop	0.36	Christmas	0.48
	Travel	0.18	Food & Drink	0.22	R&B & Soul	0.36	Fashion & Access.	0.47
Lowest	Society & Culture	0.02	Health	0.05	Psychology	-0.38	Law Enforcement	-0.10
	Politics & Government	-0.13	Environment	0.04	Injuries	-0.40	Spam & Bulk Mail	-0.11
	News & Events	-0.14	Social Science	-0.03	Heart Diseases	-0.52	Pain Management	-0.11
	Social Science	-0.21	Politics & Government	-0.07	Mental Health	-0.71	Current Events	-0.14
	Health	-0.22	News & Events	-0.08	Pain Management	-1.35	Boxing	-0.19

highest and lowest attitude, both for questions and answers. Similarly, Table 3.6 shows the categories with the highest and lowest sentimentality, again both for questions and answers. These results indicate that both the attitude and sentimentality of questions and answers are highly influenced by the category in which they are posted.

Since the dependence of the attitude and sentimentality on the topic is so pronounced, we test whether the topical differences induce all other differences. That

Table 3.6: Highest and lowest sentimentality values for top-level and leaf-level categories

	Top-level category				Leaf-level category			
	Questions		Answers		Questions		Answers	
	Category	ψ^q	Category	$\Psi^{\mathcal{A}}$	Category	ψ^q	Category	$\Psi^{\mathcal{A}}$
Highest	Family & R.ships	0.91	Family & R.ships	0.91	Pain Management	1.74	Poetry	1.02
	Social Science	0.81	Social Science	0.83	Mental Health	1.25	Mental Health	0.97
	Health	0.68	Beauty & Style	0.78	Psychology	1.01	Singles & Dating	0.94
	Society & Culture	0.62	Pregnancy & Parenting	0.78	Friends	0.95	Friends	0.92
	News & Events	0.62	Business & Finance	0.76	Family & R.ships	0.95	Family & R.ships	0.91
Lowest	Home & Garden	0.32	Cars & Transportation	0.48	Yahoo! Mail	0.19	Astronomy&Space	0.36
	Business & Finance	0.32	Home & Garden	0.47	Embassies & Consul.	0.18	Biology	0.35
	Consumer Electronics	0.32	Science & Mathematics	0.46	Packing & Prep.	0.17	Accounts & Pass.	0.35
	Cars & Transportation	0.30	Local Businesses	0.46	Addr. Book & Cal.	0.17	Geography	0.32
	Local Businesses	0.27	Comp. & Internet	0.45	External Mail	0.17	Yahoo! Mail	0.32

is, we test if the differences between classes can be fully explained by the fact that class members have different topical interests. To test this hypothesis, we look at demographic differences for the attitude, averaged over all sentences in the set of answers, on a per-topic basis. Table 3.7 shows the difference of the average attitudes for a selection of topics. In the table, differences that are significant at the 1% level for a two-sided t-test for equality of means are indicated in bold font. The attitude differences correspond to the numerical difference for the attitude of each of the following demographic group pairs: female versus male genders, $[10, 20]$ versus $[70, 80]$ age intervals, Black versus White races, and $[0\%, 5\%)$ versus $[50\%, 55\%)$ of population having a bachelor's degree or higher. The main finding of this experiment is that topical differences are not sufficient to explain the differences between demographic groups and that, even for the same topic, different groups express different sentiments.

Table 3.7: Differences in demographical features on a per-topic basis

Topic	Gender	Age	Race	Ed. level
Beauty & Style	**0.111**	**0.164**	**−0.049**	**0.028**
Business & Fin.	**−0.025**	**0.119**	0.010	**−0.047**
Arts & Human.	**0.180**	**0.233**	**0.015**	**0.103**
Pregn. & Parent.	**0.154**	**0.172**	**−0.020**	**−0.081**
News & Events	**0.015**	−0.018	**0.036**	−0.024
Games & Recr.	**0.090**	0.015	−0.008	−0.003
Science & Math.	−0.001	**0.103**	**0.018**	**0.034**

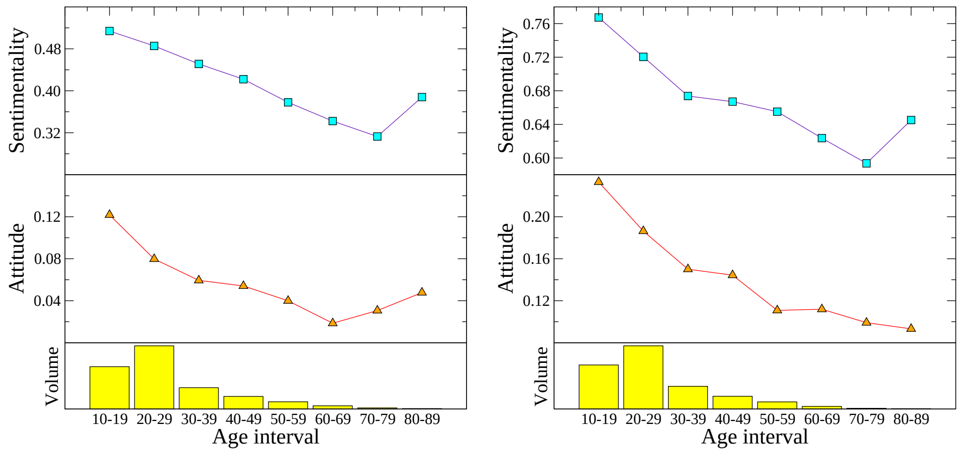


Figure 3.6: Attitude and sentimentality based on askers' (left) and answerers' (right) age.

Demographical Features

The results of this section are obtained by averaging the Φ^K and Ψ^K values of askers and also by averaging the Φ^R and Ψ^R values of answerers in a certain demographic group.

Gender. Our analysis suggest that women are more sentimental when answering a question than men (on average, $\Psi^R = 0.76$ and $\Psi^R = 0.66$ for women and men, respectively). We also observe that, in terms of attitude, men are more neutral, whereas women have more positive attitude in their answers (on average, $\Phi^R = 0.23$ and $\Phi^R = 0.13$ for women and men, respectively). We observe a similar behavior in the questions they post.

Age. Fig. 3.6 shows the attitude and sentimentality values for askers and answerers of varying age. We observe that, in general, the sentimentality decreases with increasing age. We also observe a trend towards negative attitude as the reported age of the user increases.

Race. We observe that users from predominantly Black ZIP codes have more neutral attitude values (on average, $\Phi^R = 0.19$, $\Phi^R = 0.17$, and $\Phi^R = 0.19$, for Asian, Black, and White, respectively). Users from predominantly Asian ZIP code are less sentimental in both asking and answering among these race groups (on average, $\Phi^R = 0.67$, $\Phi^R = 0.71$, and $\Phi^R = 0.72$, for Asian, Black, and White, respectively).

Education level. As the education level increases, the data suggest that the sentimentality of askers tend to increase (Fig. 3.7). However, answerers become more neutral as education level increases. On the other hand, the attitude shows a similar increasing trend for both askers and answerers with increasing education level. Irregularities in the 55%–60% interval are due to the relatively high volume of users with the self-provided ZIP code 90210 (Beverly Hills).

Spatial Features

The results of this section are obtained by averaging the Φ^K and Ψ^K values of askers and by averaging the Φ^R and Ψ^R values of answerers who reported a specific ZIP code.

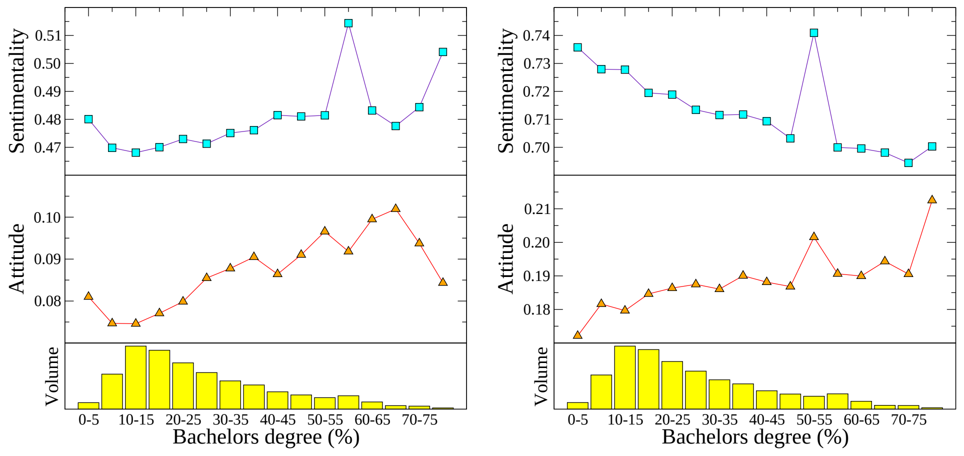


Figure 3.7: Attitude and sentimentality based on askers' (left) and answerers' (right) education level.

ZIP code. We do not observe a clear effect of the location on sentimentality. Hence, we only display the two-digit ZIP code prefixes and corresponding states where the highest and lowest sentimentality values are observed (Fig. 3.8).[15] The attitude distribution is not very conclusive either, but it gives some hints about the user profiles of certain states.

Temporal Features

The results in this section are obtained by averaging the Φ^A and Ψ^A values of all answers posted in a specific time interval. In all cases, the Eastern Time Zone[16] is used as the timestamps in our data are in this timezone.

[15]Note that ZIP code prefixes typically correspond to geographically adjacent regions. See http://en.wikipedia.org/wiki/List_of_ZIP_code_prefixes.

[16]http://en.wikipedia.org/wiki/Eastern_Time_Zone

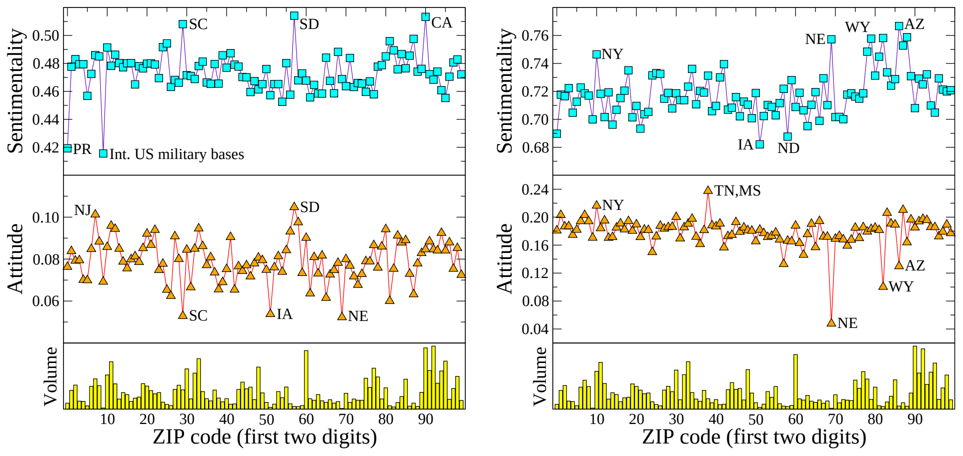


Figure 3.8: Attitude and sentimentality based on askers' (left) and answerers' (right) location.

Month. The attitude and sentimentality show significant variation across the months. Answers posted during the summer and in the holiday season (i.e., December) have higher sentimentality and more positive attitude. Both the lowest sentimentality and attitude are observed in March.

Day of month. The variance over days of the month is very minor. Hence, we omit a discussion on this feature.

Day of week. Interesting behaviors are observed for day of the week. Answers posted during the weekends and on Friday are more sentimental than those posted during the weekdays (Fig. 3.9 (left)). The attitude moves from positive to neutral as the days go from Sunday to Thursday. Then, it changes its trend on Friday and Saturday.

Hour of day. Sentimentality shows a slightly increasing trend during the day (Fig. 3.9 (right)). Especially, a sharper increase is observed between the 18:00 and 23:00 hours, followed by a decreasing trend throughout the night. The attitude

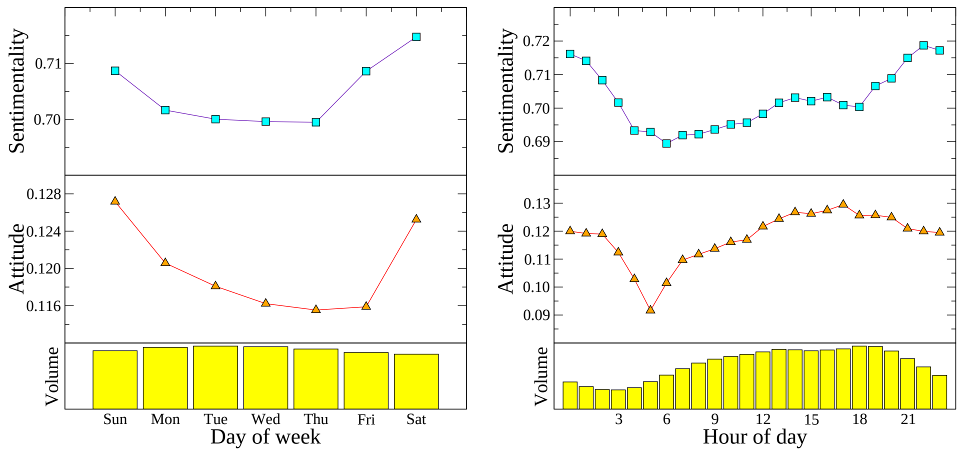


Figure 3.9: Answers' sentimentality and attitude versus day of week (left) and hour of day (right).

reaches its lowest value around 5:00 in the morning, which might correspond to an average of 3:00–4:00 on a national level when correcting for the fact that we use the Eastern Time for all users.

3.1.6 Further analyses

We determine five different concepts, for which we deepen our analysis on the attitude and sentimentality: topical context, class interaction, experience level, emotional congruence, and answer preference.

Topical Context

As mentioned in Section 3.1.5, the topical context plays an important role on the attitude and sentimentality of answers (Tables 3.5 and 3.6), i.e., the interpretation of a concept may differ according to the context. The same concept may be mentioned in a very positive sense in one category while the attitude can be quite negative in another. As a representative example, in our sample, the opinions about "Tiger

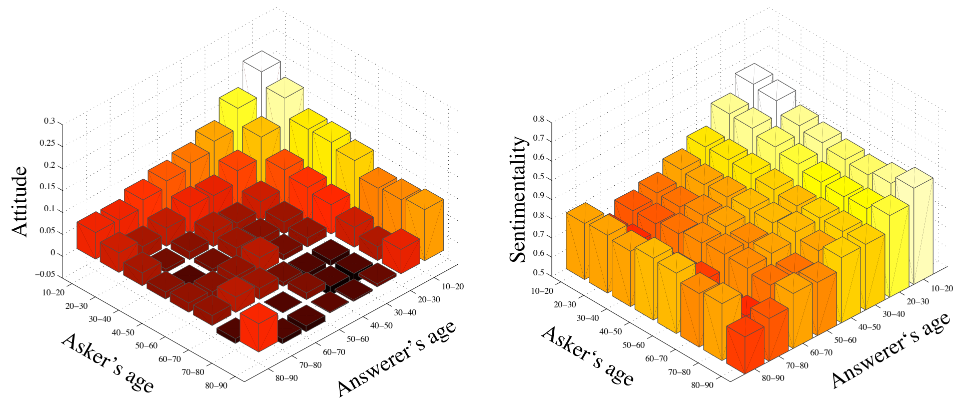


Figure 3.10: Class interaction: Attitude (left) and sentimentality (right) of answers versus ages of askers and answerers.

Woods" show high variation based on the context of the discussion.[17] Questions about him receive answers with negative attitude in the News & Events ($\phi^s = -0.10$) category, whereas the answers are quite positive in the Sports ($\phi^s = 0.18$) and Games & Recreation ($\phi^s = 0.35$) categories. We observe a neutral attitude in Social Science ($\phi^s = -0.02$) and Family & Relationships ($\phi^s = 0.04$) categories. We further quantify the influence of the topical context in Section 3.1.7.

Class Interaction

Gender. We observe that the askers are more likely to receive answers from users with the same gender. Especially, female askers receive most answers from other females (%63.5). Female answerers have more positive attitude when the asker is a female ($\Phi^R = 0.17$). According to our analysis, the lowest attitude is observed when both the asker and answerer are males ($\Phi^R = 0.03$).

[17]In our case, opinions refer to sentences that include a named entity extracted with the Stanford CoreNLP tool (http://nlp.stanford.edu/software/corenlp.shtml).

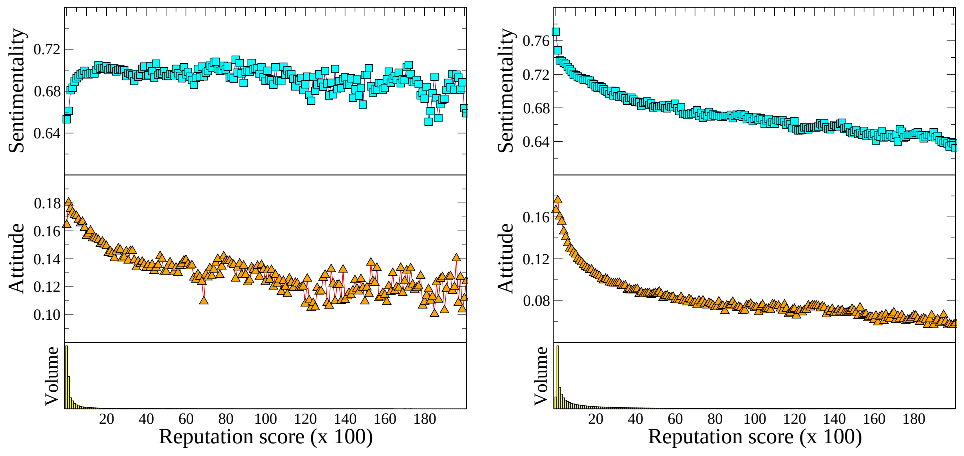


Figure 3.11: Experience level: Attitude and sentimentality for questions (left) and answers (right) based on users' experience.

Age. Younger people post and receive answers with a more positive attitude (Fig. 3.10). The attitude reaches the maximum when both the asker and answerer are less than 20 years old. In particular, people are likely to respond to people of the same age in a more positive manner. On the other hand, we do not observe a sentimentality change in answers given to different age groups.

Income. When both the asker and answerer are within the 30K–50K income range, the attitude of answers is slightly more positive. Otherwise, we do not observe any strong trend with the income level.

Experience Level

As discussed in Section 3.1.2, Yahoo! Answers awards points to its users for answering questions, with additional points being awarded for best answers. The total number of accumulated points can hence be seen as an indicator of the experience of

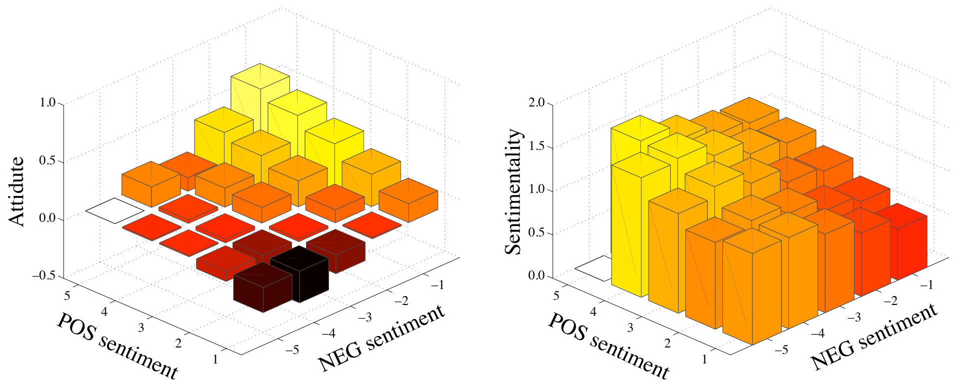


Figure 3.12: Emotional congruence: Questions' positive and negative sentiments versus the attitude (left) and sentimentality (right) of received answers.

the user in the system. Herein, we try to understand whether more experienced users differ in the sentiments they express in their questions and their answers.

As far as the questions are concerned, we observe that users with very little or no experience tend to issue the most positive questions, with the attitude of the question decreasing with an increase in experience (Fig. 3.11 (left)). On the other hand, as the experience increases, answerers become less sentimental in their answers (Fig. 3.11 (right)). This may indicate a correlation between the experience and the objectivity of answerers. This is also supported by the observation that the attitude becomes less positive as the experience increases.

Emotional Congruence

Fig. 3.12 shows the attitude and sentimentality of answers that are posted in response to questions with different positive and negative sentiment scores.[18] In general, there is a strong positive correlation between the sentimentality of initial

[18]The data point $(-5, 5)$ is omitted from the two plots due to the very low volume of sentences.

Table 3.8: Answer preference: Answers selected as the best answer differ from other answers in terms of both attitude and sentimentality

		Answers		
	Category	Best	Other	Diff.
Attitude	Business & Finance	0.16	0.32	-0.16
	News & Events	-0.07	-0.14	0.08
	Environment	0.05	-0.01	0.07
	Entertainment & Music	0.21	0.15	0.06
	Health	-0.01	0.05	-0.06
Sentiment.	Business & Finance	0.47	0.77	-0.30
	News & Events	0.68	0.76	-0.08
	Travel	0.53	0.61	-0.08
	Local Business	0.44	0.51	-0.07
	Entertainment & Music	0.74	0.67	0.06

questions and received answers. That is, highly sentimental questions are more likely to receive very sentimental answers and vice versa. The way answers are created is also determined by the attitude of questions. More negative questions receive answers with negative attitude and vice versa, indicating a trend towards "emotional congruence" between the asker and the answerer.

Answer Preference

In general, the best answers, which are preferred over the other answers, have lower sentimentality ($\Psi^A = 0.67$ versus $\Psi^A = 0.71$, on average). At the same time, the best answers are more likely to be selected from answers with a more positive attitude ($\Phi^A = 0.16$ versus $\Phi^A = 0.14$, on average). For certain categories, the preference is even stronger (Table 3.8). As a striking example, in the Business & Finance category, the best answers are more inclined towards those with low sentimentality and neutral attitude. On the other hand, in the Entertainment & Music category, answers with higher sentimentality and more positive attitude seem to be preferred by the users.

3.1.7 Attitude prediction

Is it possible to predict the attitude or sentimentality of the answers a particular question will attract before the answers are posted? Accurate prediction of attitude and sentimentality can have different practical use cases. For example, questions that are predicted to generate high sentimentality can be boosted in visible areas (in the main page or as hot topics) to increase page views. As another example, questions that have the potential to lead to very negative attitude in answers may be sent to moderators, beforehand.

Herein, we only focus on the problem of predicting the attitude in future answer posts. We formulate this particular task as a machine learning problem. In general, this is a very challenging task as the only information comes from the question and the asker. Features about the answerer, which the previous sections found to be correlated with the attitude in the answer, are not used as they are not available until the answers are posted. In addition to the features in Table 3.3, we also extract and use positive and negative sentiment scores of questions as features, which play a relatively important role in the prediction.[19] In our task, the value we try to predict for each question instance is the average attitude of the answers given to a question, i.e., average $\Phi^{\mathcal{A}}$.

We test the performance of the model over a dataset containing eight million question instances, obtained after filtering out questions whose askers' demographics

[19]We place these two features under the textual features category as they are obtained through text processing.

Table 3.9: Feature importance

Feature	Importance
Leaf-level category	100.0
Negative sentiment score	63.0
Positive sentiment score	39.7
Starting word	19.7
Number of ! symbols	12.0
Month	9.1
Question length	7.2
Gender of asker	6.7
Age of asker	3.7
Experience of asker	2.2

information is missing. We train our machine learning model using gradient boosted decision trees [40, 137] and test on our data via 10-fold cross-validation.[20] The 10 most important features, as provided by the learning tool, are shown in Table 3.9.

We adopt the root mean square error (RMSE) to evaluate the performance. As the baseline technique, we use a simple yet effective predictor which always predicts the average attitude value observed in the training data. We build our classifier using different combinations of feature types to observe their individual contributions to the performance.

Table 3.10 shows the improvement, relative to the baseline, achieved by different classifiers trained with different combinations of feature types. When only the topical features are used, the prediction performance can be improved by 3.14%. Textual features also significantly improve the performance. The contributions of the asker's

[20]In training, we set the number of trees to 40, the number of leaf nodes per tree to 40, and the learning rate to 0.5.

Table 3.10: Prediction performance

Classifier	RMSE	Improv.
Baseline	0.5261	–
Topical	0.5096	3.14
Topical + Textual	0.4964	5.65
Topical + Demographical	0.5091	3.23
Topical + Spatial + Temporal	0.5091	3.23
Topical + Textual + Demographical	0.4962	5.68
Topical + Textual + Spatial + Temporal	0.4960	5.72
Topical + Demographical + Spatial + Temporal	0.5086	3.35
Topical + Textual + Demographical + Spatial + Temporal	0.4939	6.12

demographics and spatio-temporal features are relatively low. When all features are used the classifier can improve the baseline by 6.12%.

3.1.8 Related work

A large body of work so far have dealt with different aspects of sentiment analysis, mainly sentiment extraction [1, 9, 32, 138], classification [33, 106, 129], retrieval [42, 43, 145], summarization [11, 85], and presentation [50]. The core application areas are finance [13, 32, 34], reviews [33, 106, 129], politics [66, 125], and news [45]. Herein, we omit a discussion of these works and focus only on works that are directly relevant to our work. Interested reader may refer to [105], for a detailed survey on sentiment analysis.

A large number of studies apply sentiment analysis to Twitter [12, 122]. These works differ from ours as they omit demographic factors and Twitter messages are explicitly written in a non-anonymous manner for mass consumption. The "We feel fine" project[21] [61] is probably the most closely related work. In that project, the

[21] http://www.wefeelfine.org/

authors use a large number of blog posts and, wherever possible, annotate the posts with demographic information provided in public user profiles as well as weather data for profiles from which also the location could be extracted. The focus of their work is on interface design and on offering experimental data visualization. However, they also observe temporal patterns over the course of a day (lowest fraction of “joy” sentiments at night) and over the course of a week (high fraction of “relaxed” during the weekend). Similar to our finding, they observe a trend for older people to be less negative. As data mining is not their main focus, however, their quantitative findings are less comprehensive than ours and do not include features such as educational level or race in their analysis.

Apart from demographically annotated blogs, Facebook data is often used as it offers rich per-user profile information. Typically, sentiment analysis is applied to status messages of users. It has been observed, for example, that users with a relationship status “in a relationship” or “married” are more likely to have a positive status message [17].

We note that neither Facebook status messages nor blog posts have the question-response interaction available in our data, which allows investigation of the effect of the original sentiment on the induced sentiments in answers. A tendency towards “emotion homophily” for comments left by friends on blogs is observed in [121]. The gender differences that we observed are independently observed in a small-scale study using MySpace blog posts [124]. Given that both blog posts and status messages on Facebook are not anonymous and written for explicit consumption by an ideally large group of people, it is surprising to see the general trends of sentiments to be re-confirmed in a question answering site.

3.2 Demographics-Based Opinion Diversification

Increasing popularity of personal web content via blogs, tweets, and other types of social media has boosted the generation of public opinions on certain famous people (politicians, singers, etc.), locations (cities, countries, etc.) and brands (products, companies, organizations, etc.). Search and retrieval of opinions on a subject is extremely useful for a number of application areas, including reputation management. However, due to the fact that there might be enormous number of opinions on a subject, or the opinions are skewed towards a viewpoint in some cases, the task of sampling and ranking of the opinions by representing all the viewpoints is a complex task.

Because of the information overload, traditional web search and text retrieval methods fail to identify and select a set of diverse but representative opinions; therefore, they commonly return unsatisfactory results. For example, opinions written from the same location about a mobile network operator can be all positive; although, people in another area can have difficulties connecting to that network. Another example is, the opinions about a singer can be dominated by positive ones if they are mostly written by a specific age group. Traditional similarity-based techniques are likely to miss uncommon opinions in those scenarios, while diversification algorithms are proposed to solve the problems.

Diversity became a popular topic in the last decade, and studied for spatial data [59, 69], text retrieval [25], web search results [4] and product recommendation [140, 148]. In opinion search, however, understanding the semantics of a sentence and diversifying opinions accordingly still remains as an unsolved problem. Based on the hypothesis of "network diversity is positively associated with receiving more diverse and less redundant information" [7], we argue that opinion diversity can also be achieved by considering the diversity of the sources of opinions, which is basically the metadata including demographics of the opinion holders and other relevant information.

In this study, we first formalize the diversification problem on opinion-user space, then reduce it to a well-known problem called Maximal Marginal Relevance [18] and attack the problem with an heuristics approach. In order to test the proposed idea on a real-world problem, we propose a framework to extract high-quality opinionated sentences from a large Q&A website. We also present metrics for sentiment analysis in order to distinguish factual sentences from opinions. Based on the metadata extracted along with the opinions and similarity metrics defined, we select a diverse set opinions for a given entity (product, person, location, or organization) using a state-of-the-art diversification algorithm.

As a motivating example, we show how the mean and standard deviation of the metrics for sentiment analysis on diverse opinions change for given entity queries. Since it is possible to diversify the opinions based on a single or rather a combination of the attributes (age, gender, date, age and date, etc.), we compare those sets of attributes that are used for diversification on person, location and organization entity queries. We diversify opinions based on different demographic attributes, such as age, gender, income, education level, etc. Besides, our framework can also diversify opinions based on location, sentiment, and in time. We propose metrics to evaluate our results. We compare the diverse opinions with top-k results.

3.2.1 Application areas

It's been discussed that there are different dimensions of diversity: diversity of resources, topics, viewpoints, demographics, language, location, and time [53]. Therefore, it is possible to diversify opinions regarding the attributes of the writer and the opinion itself. Examples of diversifying such attributes and the related application areas are given below:

Demographic (age, gender, race, etc.) diversity: People from different age categories (teenagers, middle-aged, old, etc.) may have different opinions about popular singers, politicians, or even products or companies. Those variations may also occur when genders are compared.

Sentiment-based diversity: Online shopping websites allow its users to rate products along with their reviews, and the most helpful positive and negative comments are often displayed next to each other. Those ratings are generally good indications of the polarity of the reviews.

Spatial diversity: Opinions on a politician may differ based on the region, city, state, or even country.

Temporal diversity: For the reputation management, the change of people's opinions along time is typically important. For example, the opinions about Obama reaches its peak before and after the election time, but the volume reduces over time.

Categorical diversity: Opinions on "Tiger Woods" show high variation based on the context of the discussion, such as in Sports, Video games, and News & Events categories.

3.2.2 Preliminaries

Before formulating the problem, let us define some basic components [95]:

opinion holder: person with some demographic information; generally a website user.

entity: an object, i.e., product, person, organization, location, etc., about which people express opinions.

opinion: a view, attitude, or appraisal towards a certain entity.

opinionated sentence: a sentence that explicitly or implicitly expresses positive or negative opinions.

3.2.3 Opinions vs. facts

Identifying whether a sentence is an opinion or fact is a well-known and unsolved problem in NLP. Based on the definition of opinionated sentences (see Section 3.2.2), we hypothesize that the opinionated sentences are more likely to have non-zero sentimentality ($\varphi^s(s_\ell) > 0$) compared to the factual sentences. We test this hypothesis in two step:

First, average attitude and sentimentality metrics are computed from 40K randomly-sampled sentences over seven web datasets.[22] Assuming the Wikipedia abstracts are generally factual sentences, we show that abstracts composed of factual sentences have the least sentimentality among some representative datasets on the web (Figure 3.13). Therefore, eliminating sentences with non-zero sentimentality ($\varphi^s(s_\ell) > 0$) will possibly leave us opinionated sentences.

[22]http://caw2.barcelonamedia.org/node/7

Source	Data type
Y! Answers	Q&As
Ciao	Movie reviews
Myspace	Forum discussions
Twitter	Short messages
Slashdot	Comments
Y! News	News
Wikipedia	Abstracts

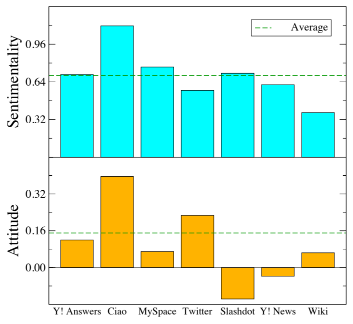


Figure 3.13: Web datasets with different characteristics (left), and average attitude and sentimentality of sampled sentences (right).

Next, we test whether this assumption works for the sentences retrieved from Y! Answers. We manually labeled 1000 sentences as either factual or opinionated, and check if we lose opinions with $\varphi^s(s_\ell) = 0$, or keep factual sentences with $\varphi^s(s_\ell) > 0$. Our experiments show that the assumption "opinionated sentences have sentimentality > 0" is often satisfactory to build an opinion database for our opinion diversification task.

3.2.4 Opinion dataset generation

Collecting opinions for this study requires using a large-scale dataset with rich information regarding sentiments. Although blog and social media (e.g., Twitter) datasets seem very suitable, they turn out to be inadequate in our context due to the unavailability of personal information about the creators of opinions. Hence, in this work, we prefer to use a large question-answer dataset obtained from Yahoo! Answers. This dataset contains demographics and experience information about users. Moreover, it includes a fairly large amount of content with sentiments and is also suitable to explore different views on a subject. An analysis of Yahoo! Answers is given in [67].

To generate an opinion dataset with high-quality sentences from Yahoo! Answers data, a large-scale and multi-step processing is required. We will not go into the details due to the page limitations; however, we try to give a complete view of the overall extraction process below. An overview of the steps are given in Figure 3.14.

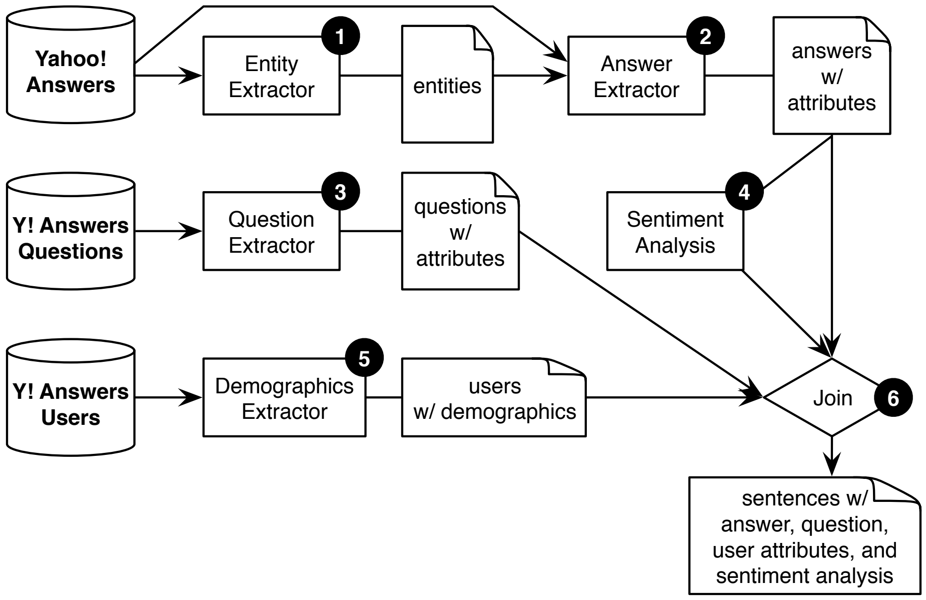


Figure 3.14: Overview of the sentence and attribute extraction process on Yahoo! Answers.

First, all named entities (i.e., persons, organizations, locations) are detected from answers with Stanford Named Entity Recognizer (NER) [39]. Based on the empirical results, entities with maximum of five words ($w_e \leq 5$) and at least 50 occurrences ($f_e \geq 50$) are selected for the database.

Using the set of selected entities, answers in English are split into sentences, and the ones containing at least one entity are kept for further processing. If the sentence begins with a conjunction (e.g., but, P.S., frankly, etc.) or a pattern similar to "I believe [that]", this part is removed. After improving the quality, if the length of the sentence is appropriate (i.e., $5 \leq |w_s| \leq 70$ and $|ch_s| \leq 400$) we keep the sentence along with some textual attributes including the matched entity, its position in the sentence, number of occurrences, etc. Note that only opinionated sentences with a positive sentimentality ($\psi^s(s_\ell) > 0$) are included in the opinion database (see Section 3.2.3).

For the selected sentences, the metadata of the answer that the sentence belong to, and of the question that the answer belong to, are also extracted. The metadata includes date and time when the answer is written, answerer's user ID, and the category of the corresponding question. With the corresponding Yahoo! accounts, we obtain self-provided registration information, such as birth year, gender, ZIP code, and country. For users with an existing US ZIP code, we obtain demographic estimates of their income (as quantified by the annual per-capita income), their education level (as quantified by the fraction of the population holding a bachelor's degree or higher), and even their race via the 2000 Governmental Census Data[23] using the same approach as in [134]. Details of demographic information extraction are discussed in [67].

The final opinion database contains opinionated sentences with rich metadata of demographic (age, gender, estimated income, estimated education level, and race),

[23]http://factfinder.census.gov/

spatial (ZIP code), temporal (date and time), textual (entity, position, number of occurrences), and categorical information.

3.2.5 Proposed diversification framework

For a given entity (object), top-k search finds the opinions based on similarity defined between an opinion and the entity (e.g., number of occurrences, most recent ones, most viewed, etc.). However, the results are generally unsatisfactory since they do not represent the whole opinion space or summarize well.

Demographics-based opinion diversification let users or companies summarize the opinions on an entity (generally product or person) based on geographic location, demographic information, and/or along time. The problem is formulated in Definition 3.2.1.

Definition 3.2.1. **(α, β, γ)-diverse top-k search.** *Given a set of opinions X, each written by a single opinion holder in Y, and an entity as query q,* (α, β, γ)-diverse top-k search *finds a multiset of k opinions and its holders, $R^* = \{(x_i, y_i) : x_i \in X, y_i \in Y, 1 \leq i \leq k\}$, such that*

$$\begin{aligned} R^* = \underset{\substack{R \subseteq (X \times Y) \\ |R| = k}}{\operatorname{argmax}} [\alpha \sum_{i=1}^{k} s(q, x_i) \\ + \frac{2\beta}{k(k-1)} \sum_{i=1}^{k} \sum_{j=i+1}^{k} (1 - s'(x_i, x_j)) \\ + \frac{2\gamma}{k(k-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} (1 - s''(y_i, y_j))] \end{aligned} \tag{3.1}$$

where n is the number of opinions holders who wrote k-selected items. α, β, γ are relevance, item-diversity, and source-diversity ratios, respectively. s, s', s'' are entity-to-opinion, opinion-to-opinion, and user-to-user similarity functions, respectively.

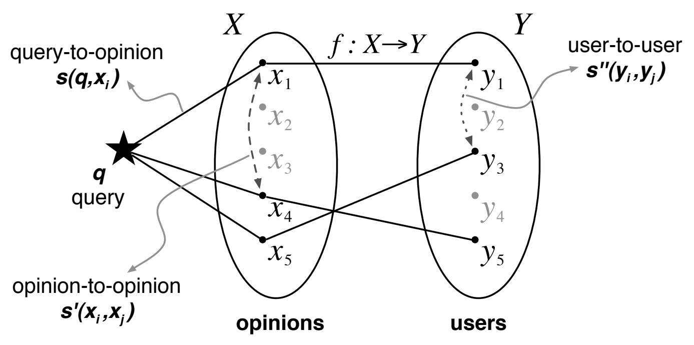


Figure 3.15: Similarity functions and relationships between opinions, opinion holders, and the query.

The objective is to find k opinions in $(X \times Y)$ which are relevant to q (depending on α), but diverse with respect to the opinions (depending on β), and their holders (depending on γ). Obviously, α is a dependent coefficient, and can be obtained with $\alpha = 1 - (\beta + \gamma)$.

General concept of all user-generated opinions/reviews can be represented with the following graph relation: $f : X \to Y$ is injective, assuming that each opinion $x_i \in X$ has a unique id, and is generated by only one user $y_i \in Y$, even if two opinions have the exact same content. This one-to-one property lets us use the attributes of the users (demographics) as a property of the generated content itself. Under this assumption, (α, β, γ)- diverse top-k search problem reduces to Maximal Marginal Relevance (MMR) problem [18], which can be solved with:

$$R^* = \underset{\substack{R \subseteq (X \times Y) \\ |R| = k}}{\operatorname{argmax}} [(1-\lambda) \sum_{i=1}^{k} s(q, x_i) + \frac{2\lambda}{k(k-1)} \sum_{i=1}^{k} \sum_{j=i+1}^{k} (1 - \overline{s}((x_i, y_i), (x_j, y_j)))] \quad (3.2)$$

where $\lambda = 1 - \alpha = \beta + \gamma$, and $\overline{s}$ is a linear combination of s' and s'' with respect to β and γ.

The problem of maximizing diversity of a result set is proved to be NP-hard [20]; however, there are heuristic diversification approaches in product recommendation [140] and web search [4]. We may use the index-based diverse browsing method presented in [69].

Chapter 4: Direction Awareness and Diversity for Citation Recommendation

The academic community has published millions of research papers to date, and the number of new papers has been increasing with time. For example, based on DBLP[24], computer scientists published 3 times more papers in 2010 than in 2000 (see Figure 4.1-left). With more than one hundred thousand new papers each year, performing a complete literature search became a herculean task. A paper cites 20 other papers on average (see Figure 4.1-right for the distribution of citations in our data), which means that there might be more than a thousand papers that cite or are cited by the papers referenced in a research article. Researchers typically rely on manual methods to discover new research such as keyword-based search via search engines, reading proceedings of conferences, browsing publication list of known experts or checking the reference list of the paper they are interested. These techniques are time-consuming and only allow to reach a limited set of documents in a reasonable time. Developing tools that help researchers to find unknown and relevant papers will certainly increase the productivity of the scientific community.

Some of the existing approaches and tools for the literature search cannot compete with the size of today's literature. Keyword-based approaches suffer from the

[24]statistics based on data acquired from DBLP in Dec'11

confusion induced by different names of identical concepts in different fields. (For instance, *partially ordered set* or *poset* are also often called *directed acyclic graph* or *DAG*). Conversely, two different concepts may have the same name in different fields (for instance, *hybrid* is commonly used to specify software hybridization, hardware hybridization, or algorithmic hybridization). These two problems may drastically increase the number of suggested but unrelated papers.

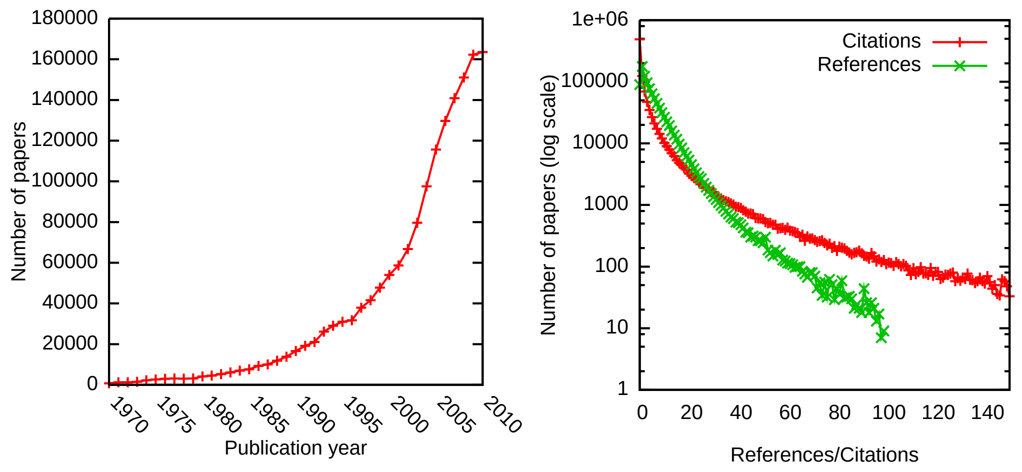


Figure 4.1: Number of new papers published each year based on DBLP (left), and number of papers with given citation and reference count (right).

To alleviate the above mentioned problems, we built a web service called the**advisor**[25]. It takes a bibliography file containing a set of papers, i.e., *seeds*, as an input to initiate the search. The user can specify that she is interested in classical papers or in recent papers. Then, the service returns a set of suggested papers ordered with respect to a ranking function. The service works using only the *citation graph* of known bibliography. In other words, it does not take the textual data into account because our

[25] http://theadvisor.osu.edu/

aim is finding all conceptually related and high quality documents even if they use a different terminology. It has been shown that text-based similarity is not sufficient for this task and that most of the relevant informations are contained within the citation graph [117]. Besides, it is plausible that there is already a correlation between citation similarities and text similarities of the papers [113].

RefSeer[26] is another webservice that shares our goals but uses a very different approach. It aims at providing relevant references of an existing text by discovering its main topics and suggests the most famous works within each topic of interest. Therefore, it tends to suggest only very well-cited documents. We believe a citation based approach will be more local and will provide the opportunity of finding papers that are not popular but still highly relevant.

There are various citation-analysis-based paper recommendation methods depending on a pairwise similarity measure between two papers. Bibliographic coupling, which is one of the earliest works, considers papers having similar citations as related [63]. Another early work, Cocitation, considers papers which are cited by the same papers as related [116]. A similar cites/cited approach by using collaboration filtering is proposed by McNee et al. [100]. CCIDF also considers only common citations, but by weighting them with respect to their inverse frequencies [82].

More recent works define different measures such as Katz which is proposed by Liben-Nowell and Kleinberg for a study on the link prediction problem on social networks [93] and used later for information retrieval purposes including citation recommendation by Strohman et al. [117]. For two papers in the citation network, the Katz measure counts the number of paths by favoring the shorter ones. Lu et al.

[26] http://refseer.ist.psu.edu/

stated that both bibliographic coupling and Cocitation methods are only suitable for special cases due to their very local nature [97]. They proposed a method which computes the similarity of two papers by using a vector based representation of their neighborhoods in the citation network. Liang et al. argued that most of the methods stated above considers only direct references and citations alone [92]. Even Katz and the vector based method of [97] consider the links in the citation network as simple links. Instead, Liang et al. added a weight attribute to each link and proposed the method Global Relation Strength which computes the similarity of two papers by using a Katz-like approach.

Many works use random walk with restarts (RWR) for citation analysis [48, 81, 88, 98]. RWR is a well known and efficient technique used for different tasks including computing the relevance of two vertices in a graph [104]. It is very similar to the well known PageRank algorithm [15] which is used by both Li and Willett [88] (ArticleRank) and Ma et al. [98] to evaluate the importance of the academic papers. Gori and Pucci [48] proposed an algorithm, called PaperRank, for RWR-based paper recommendation which can also be seen as a Personalized PageRank computation [60] on the citation graph. Lao and Cohen [81] also used RWR for paper recommendation in citation networks and proposed a learnable proximity measure for weighting the edges by using machine learning techniques. As far as we know, none of these works study the recent/traditional paper recommendation problem. The closest work is Claper [132] which is an automatic system that measure how much a paper is classical, allowing to rank a list of paper to highlight the most classical ones.

In Section 4.1, we evaluate the existing algorithms and present new algorithms that power the**advisor**. We introduce a class of parametric algorithms, said to be *direction aware*, which allow to give more importance to either the citation of papers or their references. They make the citation suggestion process easily tunable for finding either recent or traditional relevant papers. In particular we extend two eigenvector based methods into direction-aware algorithms, namely DaRWR and DaKatz. These algorithms are compared to state-of-the-art citation-based algorithms for bibliographic recommendation and their adequation to the problem is studied.

Next, we present different implementations and ordering techniques for reducing the query processing time in Section 4.2. Finally, we enhance various result diversification techniques with direction-awareness property for paper recommendation, propose new algorithms based on vertex selection and query refinement, and compare in Section 4.3.

4.1 Citation Recommendation and Direction Awareness

4.1.1 Problem definition and solutions

Let $G = (V, E)$ be the *citation graph*, with n papers $V = \{v_1, \ldots, v_n\}$. In G, each directed edge $e = (v_i, v_j) \in E$ represents a *citation* from v_i to v_j. We use the phrases *"references of v"* and *"citations to v"* as to describe the graph around vertex v (see Figure 4.2). We use $deg^-(v)$ and $deg^+(v)$ to denote the number of references of and citations to v, respectively.

In this work, we target the problem of paper recommendation assuming that the researcher has already collected a set of papers in the manuscript preparation [117]. Therefore, the objective is to return papers that the given manuscript might cite:

Paper recommendation (PR): Given a set of m seed papers $\mathcal{Q} = \{p_1, \ldots, p_m\}$ s.t. $\mathcal{Q} \subseteq V$, and a parameter k, return top-k papers which are relevant to the ones in $\mathcal{Q}$.

Random walk with restart

RWR is widely used in many fields. In citation analysis, PaperRank [48] is a method based on random walks in the citation graph G. However, the current structure of G is not suitable for finding recent and relevant papers since such papers have only a few incoming edges. Moreover, since the graph is acyclic, all random walks will end up on old papers. To alleviate this, given a PR query with inputs $\mathcal{Q}$ and k, PaperRank constructs a directed graph $G' = (V', E')$ by slightly modifying the citation graph G as follows:

A source node s is added to the vertex set: $V' = V \cup \{s\}$. Back-reference edges (E_b), the edges from s to seed papers (E_f), and restart edges from V to s (E_r) are added to the graph: $E_b = \{(y, x) : (x, y) \in E\}$, $E_f = \{(s, v) : v \in \mathcal{Q}\}$, $E_r = \{(v, s) : v \in V\}$, and $E' = E \cup E_b \cup E_f \cup E_r$.

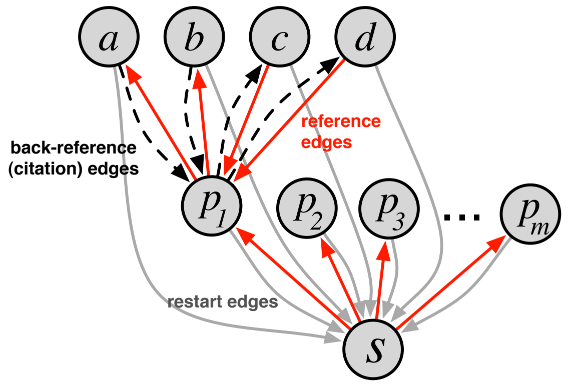


Figure 4.2: Citation graph with source node s and seed set $\mathcal{Q} = \{p_1, \ldots, p_m\}$. The papers a and b are cited by p_1, and c and d cites p_1. Note that there is a corresponding back-reference edge for each reference.

The new directed graph G' has *reference* (red), *back reference* (dashed), and *restart* (gray) edges (see Figure 4.2). In this model, the random walks are directed towards both references and citations of the papers. In addition, the restarts from the source vertex s will be distributed to only the seed papers in $\mathcal{Q}$. Hence, random jumps to any paper in the literature are prevented. We assume that a random walk ends in v continues with a neighbor with a damping factor $d \in (0, 1]$. And with probability $(1 - d)$, it restarts and goes to the source s. Let $R_{t-1}(v)$ be the probability of a random walk ends at vertex $v \neq s$ at iteration $t - 1$. Let $C_t(v)$ be the contribution of v to one of its neighbors at iteration t. In each iteration, d of $R_{t-1}(v)$ is distributed among its references and citations equally. Hence,

$$C_t(v) = d\frac{R_{t-1}(v)}{deg^+(v) + deg^-(v)}. \tag{4.1}$$

Initially, a probability score of 1 is given to the source node, meaning that a researcher expands the bibliography starting with the paper itself: $R_0(s) = 1$ and $R_0(v) = 0$ for all $v \in V$, where R_0 is the probability at $t = 0$. PaperRank algorithm computes the probability of a vertex u at iteration t as

$$R_t(u) = \begin{cases} (1-d)\sum_{v\in V} R_{t-1}(v), & \text{if } u = s \\ \sum_{(u,v)\in E} C_t(v) + \frac{R_{t-1}(s)}{|\mathcal{Q}|}, & \text{if } u \in \mathcal{Q} \\ \sum_{(u,v)\in E} C_t(v), & \text{otherwise.} \end{cases} \tag{4.2}$$

PaperRank converges when the probability of the papers are stable. Let Δ_t be the difference vector. We say that the process is in a *steady state* when the L2 norm of Δ_t is smaller than a given value ϵ.

Direction-aware random walk with restart

A random walk with restart is a good way to find relevance scores of the papers. However, the PaperRank algorithm treats the citations and references in the same way. This may not lead the researcher to recent and relevant papers if she is more interested with those. Old and well cited papers have an advantage with respect to the relevance scores since they usually have more edges in G'. Hence G' tends to have more and shorter paths from the seed papers to old papers. We define a *direction-awareness* parameter $\kappa \in [0, 1]$ to obtain more recent results in the top-k documents. We then define two types of contributions of each paper v to a neighbor paper in iteration t:

$$C_t^+(v) = d\kappa \frac{R_{t-1}(v)}{deg^+(v)}, \tag{4.3}$$

$$C_t^-(v) = d(1-\kappa) \frac{R_{t-1}(v)}{deg^-(v)}, \tag{4.4}$$

where $C_t^-(v)$ is the contribution of v to a paper in its reference list and $C_t^+(v)$ is the contribution of v to a paper which cites v. Hence, for a non-seed, non-source paper u,

$$R_t(u) = \sum_{(v,u) \in E_b} C_t^+(v) + \sum_{(v,u) \in E} C_t^-(v). \tag{4.5}$$

For a seed node u, $R_t(u)$ is computed similarly except that each seed node has an additional $\frac{R_{t-1}(s)}{|\mathcal{Q}|}$ in the equation. $R_t(s)$ is computed in the same way as (4.2). With this modification, the parameter κ can be used to give more importance either to traditional papers with $\kappa \in [0, 0.5]$ or recent papers with $\kappa \in [0.5, 1]$. We call this algorithm *direction-aware random walk with restart* (DaRWR).

Note that DaRWR (4.12) has the *probability leak* problem when a paper has no references or citations. If this is the case, some part of its score will be lost at each iteration. For such papers, we distribute the whole score from the previous iteration towards only its references or citations.

Katz and direction awareness

The direction awareness can be also adapted to other similarity measures such as the graph-based Katz distance measure [93] which was used before for the citation recommendation purposes [117]. With Katz measure, the similarity score between two papers $u, v \in V$ is computed as

$$Katz(u,v) = \sum_{i=1}^{L} \beta^i |paths_{u,v}^i|, \tag{4.6}$$

where $\beta \in [0,1]$ is the decay parameter, L is an integer parameter, and $|paths_{u,v}^i|$ is the number of paths with length i between u and v in the graph with paper and back-reference edges $G'' = (V, E \cup E_b)$. Notice that the path does not need to be elementary, i.e., the path $uvuv$ is a valid path of length 3. Therefore the Katz measure might not converge for all values of β when $L = \infty$. β needs to be chosen smaller than the larger eigenvalue of the adjacency matrix of G''. And in practice L is set to a fixed value (in our experiment $L = 10$). In our context with multiple seed papers, the relevance of a paper v is set to $R(v) = \sum_{u \in \mathcal{Q}} Katz(u,v)$.

We extend the Katz distance by using direction awareness to weight the contributions to references and citations differently with the κ parameter as in DaRWR:

$$DaKatz(u,v) = \sum_{i=1}^{L} \left[\kappa\beta^i |Rpaths_{u,v}^i| + (1-\kappa)\beta^i |Cpaths_{u,v}^i|\right],$$

where $|Rpaths_{u,v}^i|$ (respectively, $|Cpaths_{u,v}^i|$) is the number of paths in which the last edge in the path is a reference edge of E (respectively, a citation edge of E_b).

4.1.2 Experiments

Dataset collection

The retrieval of bibliographic information and citation graph generation is a difficult task since academic papers are generally copyrighted and they are accessible through publishers' digital libraries. Therefore, we limited our study to data with license that explicitly allow data mining.

We retrieved information on 1.9M computer science articles (as of March 2012) from DBLP[27] [86], 740K technical reports on physics, mathematics, and computer science from arXiv[28], and 40K publications from HAL-Inria[29] open access library. This data is well-formatted and disambiguated; however, it contains very few citation information (less than 470K edges). CiteSeer[30] is used to increase the number of paper-to-paper relations of computer science publications, but most of its data are automatically generated [44] and are often erroneous. We mapped each document in CiteSeer to at most one document in each dataset with the title information (using an inverted index on title words and Levenshtein distance) and publication years. Using the disjoint sets, we merged the papers and their corresponding metadata from four datasets. The papers without any references or incoming citations are discarded. The final citation graph has about 1M papers and 6M references, and is currently being used in our service.

[27] http://www.informatik.uni-trier.de/~ley/db/

[28] http://arxiv.org/

[29] http://hal.inria.fr/

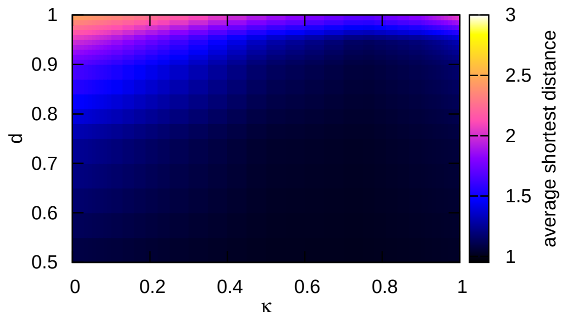
[30] http://citeseerx.ist.psu.edu/

Parameter tests

Before comparing the different methods presented in the paper, we study the impact of the damping factor d and the direction-awareness parameter κ on the papers recommended by the DaRWR algorithm. In particular, we want to verify that changing these parameters allows the user to obtain suggestions that are either closer to or farther away from the seed papers $\mathcal{Q}$, and to obtain suggestions that are either recent or more traditional. To verify these effects, a source paper published between 2005 and 2010 is randomly selected and its references are used as the seed papers. We use the top-10 results as the set of recommended papers $\mathcal{R}$. The test is repeated for 2500 distinct queries that satisfy the given constraints.

Figure 4.3 (top) shows the impacts of d and κ on the average year of $\mathcal{R}$ and average shortest distance in the citation graph between $\mathcal{R}$ and $\mathcal{Q}$. When d increases, the probability that the random research jumps back to the source node s is reduced. It allows reaching vertices distant from s to be reached more often. κ makes little difference in the average distance to the seed papers. However, setting a higher value of d should allow to find relevant papers whose relation to the seeds are not obvious.

Figure 4.17 (bottom) also shows that increasing d leads to earlier papers since they tend to accumulate more citations. But for a given κ, varying the damping factor do not allow to reach a large diversity of time frames. The direction-awareness parameter κ can be adjusted to reach papers from different years with a range from late 1980's to 2010 for almost all values of d. In our online service, the parameter κ can be set to a value of user's preference. It allows the user to obtain recent papers by setting κ close to 1 or finding older papers by setting κ close to 0.



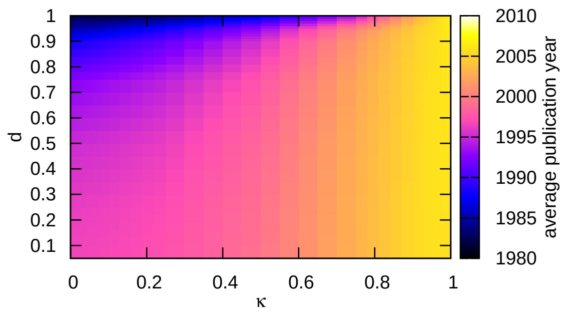


Figure 4.3: Average shortest distance from seed papers (left) and publication year (right) of top-10 recommendations by DaRWR based on d and κ.

Overall, first-level papers are often returned for $d < 0.8$; yet many papers at distance 2 and more appear. Also, it is possible to choose between traditional papers (by setting $\kappa < 0.4$) or recent papers (by setting $\kappa > 0.8$).

Experimental settings

We test the quality of the recommended citations by different methods in three different scenarios.

The **hide random** scenario represents the typical use-case where a researcher is writing a paper and trying to find some more references. To simulate that, a source paper s with enough references ($20 \leq deg^{+}(s) \leq 100$) is randomly selected from the papers published between 2005 and 2010. Then we remove s and all the papers published after s from the graph (i.e., $G_s = (V_s, E_s)$ where $V_s \subset V \setminus \{s\}$ and $\forall v \in V_s$, $year[v] \leq year[s]$) to simulate the time when s was being written. Out of $deg^{+}(s)$, 10% of the references are randomly put in the hidden set H, and the rest is used as the seed papers (i.e., $\mathcal{Q} = \{v \notin H : (s, v) \in E\}$). We compute the citation recommendations on $\mathcal{Q}$ and report the mean average precision (MAP) of finding hidden papers within the top-50 recommendations for 2500 independent queries.

The **hide recent** scenario represents another typical use-case where the author might be well aware of the literature of her field but might have missed some recent developments. Here, the hidden set H only contains the most recent references. Again, MAP of finding hidden papers within the top-50 recommendations is reported for each query. Similarly, we define **hide earlier** where the hidden papers are the oldest publications.

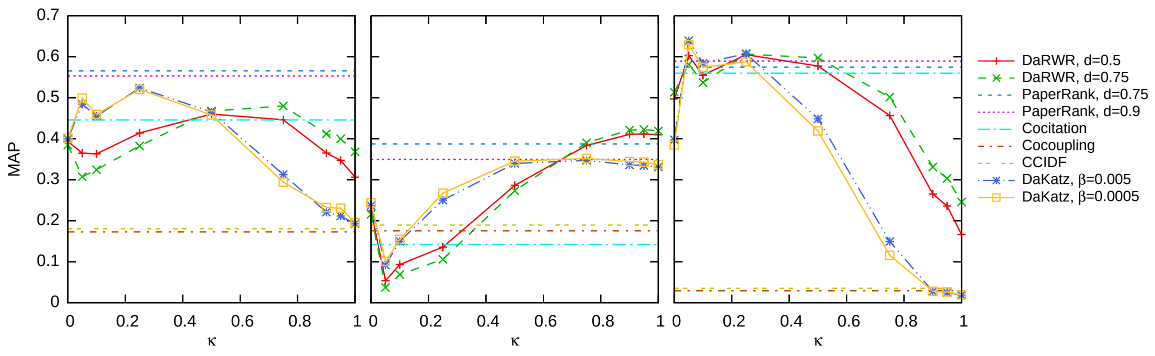
The methods we proposed are compared on the three scenarios against widely-used citation based approaches: bibliographic coupling [63], Cocitation [116], CCIDF [82], PaperRank [48] and the original Katz distance [93]. The algorithms and the parameters that lead to the best accuracy in different experiments are summarized in Table 4.1.

Table 4.1: Parameters used in the experiments.

Method	**Random**	**Recent**	**Earlier**
Katz$_\beta$	$\beta = 0.0005$		$\beta = 0.005$
DaKatz	β=0.005 $\kappa = 0.25$	$\beta = 0.0005$ $\kappa = 0.75$	$\beta = 0.005$ $\kappa = 0.05$
PaperRank	$d = 0.75$	$d = 0.75$	$d = 0.9$
DaRWR	$d = 0.75$ $\kappa = 0.75$	$d = 0.75$ $\kappa = 0.95$	$d = 0.75$ $\kappa = 0.25$

Results

Accuracy: Figure 4.4 presents a comparison of all the methods on there scenarios. Many algorithms are represented as horizontal lines since they are not direction aware.



(a) hide random (b) hide recent (c) hide earlier

Figure 4.4: Mean average precision of the algorithms on three experiments based on κ and other parameters. Note that Katz is equal to DaKatz at $\kappa = 0.5$.

The first remark is that Cocoupling and CCIDF perform poorly on all four scenarios. Cocitation performs the worst in the hide recent scenario and performs reasonably good but not the best in the other scenarios. These methods only consider counting and weighting distance-2 papers from the seeds, and they are outperformed by the eigenvector-based methods which take whole graph into account.

Notice that PaperRank performs well overall but for different values of the damping parameter d. The performance of DaKatz is significantly varying with the parameter set, but it is important to notice that the variations with the direction-awareness parameter are similar to the one observed on DaRWR. The results of Katz are not explicitly presented but can be read on DaKatz when $\kappa = 0.5$. Notice that DaKatz is always a better method that Katz. PaperRank achieves the best results when the query is generic (on the hide random scenarios); however direction-aware methods lead to higher accuracy when the query is targeted.

Table 4.2: Results of the experiments with mean average precision and 95% confidence intervals.

	hide random			**hide recent**			**hide earlier**		
	mean	interval		mean	interval		mean	interval	
DaRWR	48.00	46.80	49.20	42.22	**40.95**	**43.50**	60.64	59.48	61.80
P.R.	56.56	**55.31**	**57.80**	38.75	37.50	40.00	58.93	57.76	60.10
DaKatz	52.39	51.18	53.60	35.18	33.96	36.40	63.93	**62.76**	**65.10**
Katz$_\beta$	46.33	45.16	47.50	34.56	33.42	35.70	44.19	42.97	45.40
Cocit	44.60	43.39	45.80	14.22	13.25	15.20	55.97	54.64	57.30
Cocoup	17.28	16.36	18.20	17.56	16.61	18.50	2.93	2.57	3.30
CCIDF	18.05	17.11	19.00	18.97	17.94	20.00	3.55	3.10	4.00

The accuracy being close to each other, we report in Table 4.2 the 95% confidence interval for the best parameters of each method on the three scenarios. In each scenario, the confidence interval of the method that performs best does not intersect with any other interval. It indicates that their dominance is statistically significant.

Coverage: The previous experiments show a statistically significant but little difference in accuracy between the best method and the runner-up. We investigate whether the recommended documents are similar or different. Table 4.3 presents the intersection matrix of the different methods on the three scenarios for a limited number of queries. Each method's parameters are set to optimize the accuracy. The diagonal of the matrix shows the actual accuracy of the methods. Other values show the MAP of the intersection of the solutions of the two corresponding methods. DaKatz clearly dominates Katz. Cocitation and CCIDF recommend different documents (up to 8%). DaRWR and PaperRank can show significant differences (up to 5.6%) as well. In each scenario, the best algorithm can not be improved more than 7% using the solution of another algorithm.

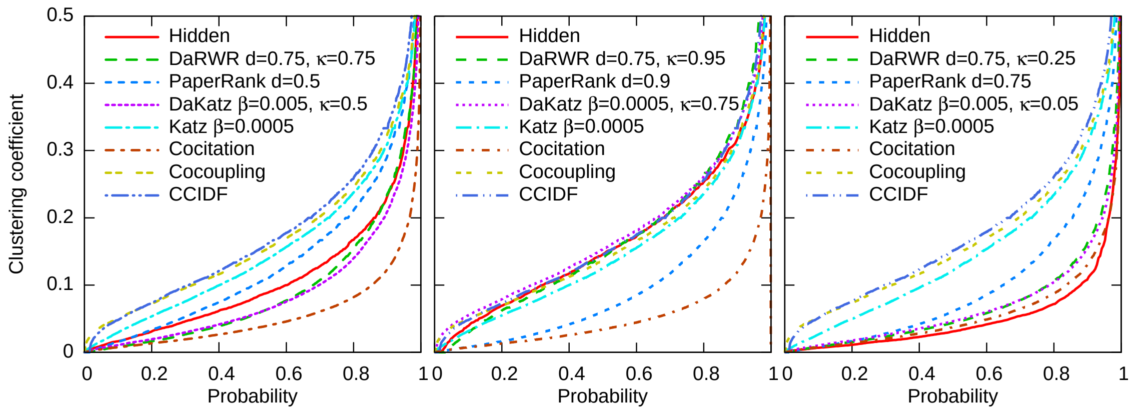
Table 4.3: Intersection matrix of the results for hide random (i), recent (ii), and earlier (iii) experiments.

(i)	DaRWR	P.R.	DaKatz	Katz$_\beta$	Cocit	Cocoup	CCIDF
DaRWR	**44.76**	41.54	40.54	34.13	31.96	11.95	12.61
P.R.		**51.97**	44.98	39.03	33.58	13.50	14.21
DaKatz			**51.89**	39.55	37.57	14.07	13.69
Katz$_\beta$				**42.73**	29.48	14.71	14.10
Cocit					**43.25**	10.46	8.95
Cocoup						**16.37**	11.64
CCIDF							**16.98**

(ii)	DaRWR	P.R.	DaKatz	Katz$_\beta$	Cocit	Cocoup	CCIDF
DaRWR	**40.14**	32.15	30.86	30.25	10.02	14.78	17.05
P.R.		**34.91**	27.34	27.75	11.31	14.17	16.30
DaKatz			**35.31**	33.23	9.05	15.95	16.79
Katz$_\beta$				**34.51**	9.54	16.05	16.72
Cocit					**13.50**	5.92	5.58
Cocoup						**17.39**	13.43
CCIDF							**19.22**

(iii)	DaRWR	P.R.	DaKatz	Katz$_\beta$	Cocit	Cocoup	CCIDF
DaRWR	**60.87**	52.39	56.56	40.10	47.31	2.17	2.306
P.R.		**57.99**	53.98	40.53	48.69	2.65	2.75
DaKatz			**63.84**	41.34	50.81	2.45	2.63
Katz$_\beta$				**42.09**	38.27	2.80	2.78
Cocit					**54.97**	2.43	2.16
Cocoup						**2.91**	2.04
CCIDF							**3.19**

Citation patterns: The large variation of the accuracy when the direction-awareness parameter varies indicates that searching for old papers is inherently different than searching for recent papers or arbitrary papers. We believe that traditional papers and recent papers cite and are being cited differently. To qualify this difference, we study the properties of the suggestions returned by the methods and compare them to the properties of the actual references within the papers. We argue that an appropriate method should suggest papers having patterns resembling the properties



(a) hide random (b) hide recent (c) hide earlier

Figure 4.5: Clustering coefficient of the suggested citations for the experiments.

of the papers it is designed to find. The *clustering coefficient* [133] C_v of a paper v can be used to qualify the citation patterns. It is computed as:

$$C_v = \frac{|\{(i,j) \in E \mid i,j \in N_v \cup \{v\}\}|}{|N_v| \times (|N_v| + 1)},$$

where N_v is the set of neighbor papers of v which either cite v or are cited by v. Intuitively, clustering coefficient indicates how close of being a clique a vertex and its neighbors are.

Figure 4.5 presents the cumulative density function of the clustering coefficients of the documents suggested by each algorithm and of the hidden papers in three scenarios. First of all, the trace of the hidden papers is different in the three scenarios. When the hidden papers are early papers, they are typically well cited and their clustering coefficients are low. (It is unlikely that a large neighborhood forms a clique since the outgoing degree is typically small.) Recent papers have a higher clustering coefficient and their neighborhoods are small. This confirms that clustering coefficient can be used to distinguish old and recent papers.

None of the methods matches the trace of the hidden paper in the **hide random** scenario. PaperRank matches its trace at the beginning of the curve, while DaRWR and DaKatz match it at the end. In the **hide recent** scenario, most algorithms have a trace similar to the one of the hidden papers beside PaperRank and Cocitation. Notice that CCIDF and Cocoupling exhibit a trace similar to the hidden papers despite suffering from a low accuracy: they find recent papers but not the relevant ones. In **hide earlier** scenario, DaRWR, DaKatz and Cocitation have patterns similar to hidden papers. The rest of the algorithms have patterns different from the hidden papers. In all cases, CCIDF has citation patterns similar to the one of Cocoupling.

This analysis shows that direction-aware algorithms can be tuned to reach a variety of citation patterns, allowing them to match the patterns of recent or old documents. However, having a similar trace is an important property but it is not enough to reach a high precision.

4.2 Service and Fast Recommendation

Sparse-matrix computations working exclusively on nonzero entries are usually not suitable for today's cache architectures if an ordinary ordering of the rows, columns, or nonzeros is used. The difficulty arises from the fact that the memory access pattern in such computations depends on the nonzero distribution in the matrix which usually does not have a well-defined regular structure. If the access pattern is random, the number of cache misses through the computation increases. Since there will be a penalty for each cache miss, reordering the nonzero accesses in the matrix is a good idea to reduce the number of cache misses and hence, the execution time.

One of the most widely used operation in network analysis is sparse matrix-dense vector multiplication (SpMxV). This operation is assumed to be the computational bottleneck for network analyses based a on random walk with restart (RWR) which is used in PageRank [103], impact factor computations [14], recommendation systems [65, 139] and finding/predicting genetic interactions [28]. The SpMxV kernel is also the core of other graph based metrics such as Katz which is proposed by Liben-Nowell and Kleinberg for a study on the link prediction problem on social networks [93] and used later for information retrieval purposes including citation recommendation by Strohman et al. [117].

In this work, we target a citation, venue, and expert recommendation problem in our publicly available web-service called the**advisor**. The service takes a bibliography file in various formats (bib, ris, xml) that contains a set of *query* papers to initiate the recommendation process. Then, it returns a set of papers ordered with respect to a ranking function. The user can guide the search or prune the list of suggested papers with positive or negative feedbacks by declaring some papers relevant or irrelevant. In this case, the service completely refines the set and shows the new results back to the user. In addition to papers, the**advisor** also suggests researchers or experts, and conferences or journals of interest. The service is designed to help researchers while performing several tasks, such as:

- literature search,
- improving the reference list of a manuscript being written,
- finding conferences and journals to attend, get subscribed, or submit papers,
- finding a set of researchers in a field of interest to follow their work,

- finding a list of potential reviewers, which is required by certain journals in the submission process.

The algorithm we employed in the**advisor** is based on RWR and is implemented by using the SpMxV operation. There exist several methods in the literature proposed to improve the cache locality for the SpMxV operations by ordering the rows and/or columns of the matrix by using graph/hypergraph partitioning [5, 126, 136, 142, 143] and other techniques [3, 108, 110, 120]. The recommendation algorithm used in the**advisor** is *direction aware*. That is, the user can specify that she is interested in classical papers or recent papers. This property brings a unique characteristic to the SpMxV operation used in the service which makes existing hypergraph partitioning based techniques [5, 142, 143] not directly applicable. We recently experimented on a direction-aware Katz-based algorithm and showed that it outperforms one without direction awareness when the objective is to find either traditional or recent papers [72].

In this work, our contribution is two-fold: First, we propose techniques to efficiently store the matrix used by direction-aware algorithms. We then propose efficient implementations of the algorithm and investigate several matrix ordering techniques based on a hypergraph partitioning model and ordering heuristics, such as the Approximate Minimum Degree (AMD) [6], Reverse Cuthill-McKee (RCM) [31], and SlashBurn [62]. State-of-the-art hypergraph partitioners are typically too slow to be used to optimize just a couple of SpMxV operations. However, considering the**advisor**'s purpose, the algorithm will be executed many times whereas the ordering is required only once. The current version of our service is already using the implementation and ordering described here.

We give a thorough evaluation of the proposed approach and algorithms, and measure the efficiency of the implementation and matrix storing/ordering techniques used in the**advisor**. The combination of all the techniques improved the response time of our service by 67% (3x). We believe that the techniques proposed here can also be useful for SpMxV-related sparse-matrix problems in social network analysis.

A preliminary version of this work was published in [70]. We extend in this work the discussion to a wider class of algorithms and exemplify the discussion by using Katz-based metrics. We also investigate the SlashBurn ordering. We discuss the impact of the convergence of the method on the choice of representation and ordering of the matrix.

4.2.1 Background

Citation Analysis: Random Walk with Restart and Katz

Citation analysis-based paper recommendation has been a popular problem since the 60's. There are methods that only take local neighbors (i.e., citations and references) into account, e.g., bibliographic coupling [63], cocitation [116], and CCIDF [82]. Recent studies, however, employ graph-based algorithms, such as Katz [93], random walk with restart [104], or well-known PageRank algorithm to investigate the whole citation network. PaperRank [48], ArticleRank [88], and Katz distance-based methods [93] are typical examples.

We target the problem of paper recommendation assuming that the researcher has already collected a list of papers of interest. Let $G = (V, E)$ be the directed citation graph with a vertex set $V = \{1, 2, \ldots, n\}$ and an edge set E, which contains (i, j) if paper i cites paper j. We define the problem as follows: Given a set of query papers $\mathcal{Q} \subseteq V$, and a parameter k, return top-k papers which are relevant to the ones in $\mathcal{Q}$.

Let E' be the undirected version of E, i.e.,

$$E' = \{\{i,j\} : (i,j) \in E\}.$$

Let $G' = (V, E')$ be the undirected citation graph and $\delta(i)$ denote the degree of a vertex $i \in V$ in G'. Random walk with restart is a widely used method in many fields. In citation analysis, RWR directs the random walks towards both references and citations of the papers. In addition, the restarts are directed only to the query papers in $\mathcal{Q}$. Hence, random jumps to any paper in the literature are prevented. Starting from the query papers, we assume that a random walk ends in paper i continues with a neighbor with a damping factor $d \in (0,1]$. And with probability $(1-d)$, it restarts and goes back to the query papers. Let $\mathbf{p}_t(i)$ be the probability that a random walk ends at vertex $i \in V$ at iteration t. Hence, $\mathbf{p}_0(i) = \frac{1}{|\mathcal{Q}|}$ for a query paper i, and 0 for other papers. Let $c_t(i)$ be the contribution of i to one of its neighbors at iteration t. In each iteration, d of $\mathbf{p}_{t-1}(i)$ is distributed among i's references and citations equally, hence, $c_t(i) = d\frac{\mathbf{p}_{t-1}(i)}{\delta(i)}$.

The Katz distance [93] is another measure which has been used for citation recommendation purposes [117]. The Katz distance between two papers $i, j \in V$ is computed as:

$$Katz(i,j) = \sum_{\ell=1}^{\infty} \beta^{\ell} |paths_{i,j}^{\ell}|, \tag{4.7}$$

where $\beta \in [0,1]$ is the decay parameter, and $|paths_{i,j}^{\ell}|$ is the number of paths of length ℓ between i and j in the undirected citation graph G'. Such a path does not need to be elementary, i.e., the path i,j,i,j is a valid path of length 3. Therefore, the Katz measure might not converge for all values of β; it needs to be chosen smaller than the reciprocal of the largest eigenvalue of the adjacency matrix of G'[31]. In citation recommendation with multiple query papers, the relevance of a paper j is computed as $\pi(j) = \sum_{i \in \mathcal{Q}} Katz(i,j)$.

Modeling sparse matrices with hypergraphs

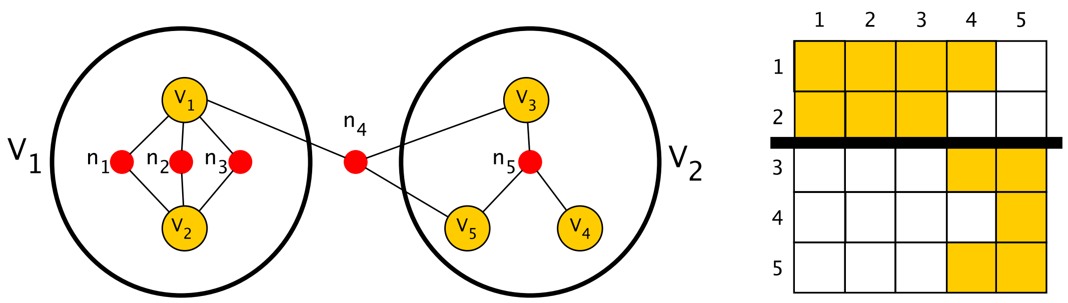
A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is defined as a set of vertices $\mathcal{V}$ and a set of nets (hyperedges) $\mathcal{N}$. A net $\eta \in \mathcal{N}$ is a subset of vertex set $\mathcal{V}$, and the vertices in η are called its *pins*. The *size* of a net is the number of its pins, and the *degree* of a vertex is equal to the number of nets that contain it. Figure 4.6.(a) shows a simple hypergraph with five vertices and five nets. A graph is a special instance of hypergraph such that each net has size two. Vertices can be associated with weights, denoted with $\mathsf{w}[\cdot]$, and nets can be associated with costs, denoted with $\mathsf{c}[\cdot]$.

A *K-way partition* of a hypergraph $\mathcal{H}$ is denoted as $\Pi = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ where parts are pairwise disjoint, each part $\mathcal{V}_k$ is a nonempty subset of $\mathcal{V}$, and union of the K parts is equal to $\mathcal{V}$.

In a partition Π, a net that has at least one pin (vertex) in a part is said to *connect* that part. The number of parts connected by a net η, i.e., *connectivity*, is denoted as λ_η. A net η is said to be *uncut* (*internal*) if it connects exactly one part, and *cut* (*external*), otherwise (i.e., $\lambda_\eta > 1$). In Figure 4.6.(a) the toy hypergraph with four internal nets and an external net is partitioned into two .

Let W_k denote the total vertex weight in $\mathcal{V}_k$ and W_{avg} denote the weight of each part when the total vertex weight is equally distributed. If each part $\mathcal{V}_k \in \Pi$ satisfies

[31]The Katz centrality of a node i can be computed as $Katz(i) = \sum_{j=1}^{n} Katz(i,j) = \sum_{j=1}^{n} \sum_{\ell=1}^{\infty} \beta^\ell (A^\ell)_{ji}$ where A is the 0–1 adjacency matrix of the citation graph. When β is smaller than the reciprocal of the largest eigenvalue, the Katz centralities can be computed as $\left((I - \beta A^T)^{-1} - I\right) \vec{I}$ where I is the identity matrix and $\vec{I}$ is the identity vector.



(a) A toy hypergraph (b) Corresponding matrix

Figure 4.6: A toy hypergraph with five vertices and five nets partitioned into two parts (a). Net n_4 is a cut net since it is connected to two parts, hence, $\lambda_4 = 2$. The rest of the nets are internal. The corresponding matrix (w.r.t. column-net model) whose nonzero entries are colored and zeros are shown with white (b).

the *balance criterion*

$$W_k \leq W_{avg}(1+\varepsilon), \quad \text{for } \; k = 1, 2, \ldots, K \tag{4.8}$$

we say that Π is *balanced* where ε represents the maximum allowed imbalance ratio.

The set of external nets of a partition Π is denoted as $\mathcal{N}_E$. Let $\chi(\Pi)$ denote the cost, i.e., *cutsize*, of a partition Π. There are various cutsize definitions [84]. In this work, we use

$$\chi_{conn}(\Pi) = \sum_{\eta \in \mathcal{N}} \mathsf{c}[\eta](\lambda_\eta - 1) \; . \tag{4.9}$$

as also used by Akbudak et al. for similar purposes [5]. The cutsize metric given in (4.9) will be referred to as the *connectivity*-1 metric. Given ε and an integer $K > 1$, the hypergraph partitioning problem can be defined as the task of finding a balanced partition Π with K parts such that $\chi(\Pi)$ is minimized. The hypergraph partitioning problem is NP-hard [84] with the above objective function. We used a state-of-the-art partitioning tool PaToH [22].

There are three well-known hypergraph models for sparse matrices. These are the column-net [21], row-net [21], and fine-grain models [23]. Here, we describe the column-net model we used for a sparse matrix $\mathbf{A}$ of size $n \times n$ with m nonzeros. In the *column-net model*, $\mathbf{A}$ is represented as a unit-cost hypergraph $\mathcal{H}_{\mathcal{R}} = (\mathcal{V}_{\mathcal{R}}, \mathcal{N}_{\mathcal{C}})$ with $|\mathcal{V}_R| = n$ vertices, $|\mathcal{N}_C| = n$ nets, and m pins. In $\mathcal{H}_{\mathcal{R}}$, there exists one vertex $v_i \in \mathcal{V}_{\mathcal{R}}$ for each row i. Weight $\mathsf{w}[v_i]$ of a vertex v_i is equal to the number of nonzeros in row i. There exists one unit-cost net $\eta_j \in \mathcal{N}_{\mathcal{C}}$ for each column j. Net η_j connects the vertices corresponding to the rows that have a nonzero in column j. That is, $v_i \in \eta_j$ if and only if $a_{ij} \neq 0$ (see Figure 4.6). The *row-net model* is the column-net model of the transpose of $\mathbf{A}$.

Matrix ordering techniques for improving cache locality in SpMxV

The SpMxV operation is defined as $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is an $n \times n$ sparse matrix with m nonzeros, $\mathbf{x}$ is the $n \times 1$ input vector, and $\mathbf{y}$ is the $n \times 1$ output vector. Let $\mathbf{P}$ and $\mathbf{Q}$ be two $n \times n$ permutation matrices. That is, $\mathbf{P}$ and $\mathbf{Q}$ have only a 1 in each of their rows and columns and the rest is 0. When the matrix $\mathbf{A}$ is ordered as $\mathbf{A}' = \mathbf{P}\mathbf{A}\mathbf{Q}$, the SpMxV operation can be written as $\mathbf{y}' \leftarrow \mathbf{A}'\mathbf{x}'$ where $\mathbf{y}' = \mathbf{P}\mathbf{y}$ and $\mathbf{x}' = \mathbf{Q}^T\mathbf{x}$. Some existing cache-locality optimization techniques use this fact and permute the rows and columns of $\mathbf{A}$ to improve cache locality.

To find good $\mathbf{P}$ and $\mathbf{Q}$, several approaches are proposed in the literature: Bandwidth reduction is proven to be promising for decreasing cache misses [120]. For this reason, the reverse Cuthill-McKee heuristic [31] has been frequently used as a tool and a benchmark by several researchers [109, 110, 126]. RCM has also been frequently used as a fill-in minimization heuristic for sparse LU factorization. Another successful fill-in minimization heuristic, the approximate minimum degree (AMD) [6], is also used for improving cache locality [109]. Another ordering heuristic SlashBurn has recently been proposed for graph compression and mining [62].

Graph and hypergraph partitioning models and techniques have been extensively studied for reducing cache misses [5, 126, 136, 142, 143]. Among those, the most similar ones to our work are [142, 143] and [5], which use hypergraph partitioning as the main tool to reduce the number of cache misses.

As should be evident, the sparse matrix storage format and the cache locality are related since the storage determines the order in which the nonzeros are processed. In this work, we use two of the most common formats. The coordinate format (COO) keeps an array of m triplets of the form $\langle a_{ij}, i, j\rangle$ for a sparse matrix $\mathbf{A}$ with m entries. Each triplet contains a nonzero entry a_{ij} and its row and column indices (i, j). The COO format is suitable for generating arbitrary orderings of the non-zero entries. The compressed row storage format (CRS) uses three arrays to store a $n \times n$ sparse matrix $\mathbf{A}$ with m nonzeros. One array of size m keeps the values of nonzeros where the nonzeros in a row are stored consecutively. Another array parallel to the first one keeps the column index of each nonzero. The third array keeps the starting index of the nonzeros at a given row where the ending index of the nonzeros at a row is one less than the starting index of the next row. A matrix represented in CRS is typically 30% smaller than the COO since the m entries representing i in COO are compressed in an array of size n in CRS.

Table 4.4: Statistics for the citation graph G.

$\lvert V\rvert$	$\lvert E\rvert$	avg δ	max δ^+	max δ^-
982,067	5,964,494	6.07	617	5418

4.2.2 Direction-aware methods for citation recommendation

As described previously, our recommendation service, the**advisor**, is designed to solve the following problem: Given the directed citation graph $G = (V, E)$, a set of query papers $\mathcal{Q} \subseteq V$, and a parameter k, return top-k papers which are relevant to the ones in $\mathcal{Q}$. We defined a *direction awareness* parameter $\kappa \in [0, 1]$ to obtain more recent or traditional results in the top-k documents [72]. Let $\delta^+(i)$ and $\delta^-(j)$ be the number of references of and citations to paper u, respectively. The citation graph we use in the**advisor** has been obtained by cross-referencing the data of four online database: DBLP, CiteSeer, HAL-Inria and arXiv. The properties of the citation graph are given in Table 4.4.

In this work, we discuss efficient ways to compute the result set using two *direction-aware* algorithms. The first one is based on the direction-aware random walk with restart (DaRWR) and the second one is based on the direction-aware Katz similarity (DaKatz). The following sections present both methods by addressing their similarities and differences when they are implemented with SpMxV operations.

Direction-aware Random Walk with Restart (DaRWR)

Given $G = (V, E)$, k, $\mathcal{Q}$, d, and the direction awareness parameter κ, our algorithm computes the steady-state probability vector $\mathbf{p}$. For an iterative DaRWR implementation, at iteration t, the two types of contributions of paper i to a neighbor paper are defined as:

$$c_t^+(i) = \mathbf{p}_{t-1}(i)\frac{d(1-\kappa)}{\delta^+(i)}, \tag{4.10}$$

$$c_t^-(i) = \mathbf{p}_{t-1}(i)\frac{d\kappa}{\delta^-(i)}, \tag{4.11}$$

where $c_t^+(i)$ is the contribution of paper i to a paper in its reference list and $c_t^-(i)$ is the contribution of paper i to a paper which cites i. The rank of paper i after iteration t is computed with,

$$\mathbf{p}_t(i) = \mathbf{r}(i) + \sum_{(i,j)\in E} c_t^-(j) + \sum_{(j,i)\in E} c_t^+(j), \tag{4.12}$$

where $\mathbf{r}$ is the restart probability vector due to jump backs to the papers in $\mathcal{Q}$, computed with,

$$\mathbf{r}(i) = \begin{cases} \frac{1-d}{|\mathcal{Q}|}, & \text{if } i \in \mathcal{Q} \\ 0, & \text{otherwise.} \end{cases} \tag{4.13}$$

Hence, each iteration of the algorithm can be defined with the following linear equation:

$$\mathbf{p}_t = \mathbf{r} + \mathbf{A}\mathbf{p}_{t-1}, \tag{4.14}$$

where $\mathbf{r}$ is an $n \times 1$ restart probability vector calculated with (4.13), and $\mathbf{A}$ is a structurally-symmetric $n \times n$ matrix of edge weights, such that

$$a_{ij} = \begin{cases} \frac{d(1-\kappa)}{\delta^+(i)}, & \text{if } (i,j) \in E \\ \frac{d\kappa}{\delta^-(i)}, & \text{if } (j,i) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{4.15}$$

The algorithm converges when the probability of the papers are stable, i.e., when the process is in a *steady state*. Let $\Delta_t = (\mathbf{p}_t(1)-\mathbf{p}_{t-1}(1),\ldots,\mathbf{p}_t(n)-\mathbf{p}_{t-1}(n))$ be the difference vector. We say that the process is in the steady state when the L2 norm of Δ_t is smaller than a given value ξ. That is,

$$\|\Delta_t\|_2 = \sqrt{\sum_{i \in V} (\mathbf{p}_t(i) - \mathbf{p}_{t-1}(i))^2} < \xi. \tag{4.16}$$

Direction-aware Katz (DaKatz)

The direction awareness can be also adapted to other similarity measures such as the graph-based Katz distance measure [93]. We extend the measure to weight the contributions to references and citations differently with the κ parameter as in DaRWR. Given $G = (V, E)$, $\mathcal{Q}$, κ, and an integer parameter L, the relevance score of paper j is computed as:

$$\mathbf{p}(j) = \sum_{i \in \mathcal{Q}} d_L(i, j), \tag{4.17}$$

where $d_L(i, j)$ is the direction aware Katz distance between a query paper i and paper j with the paths of length up to L, computed recursively as:

$$d_L(i, j) = \beta\kappa \sum_{(k,j) \in E} d_{L-1}(i, k) + \beta(1 - \kappa) \sum_{(j,k) \in E} d_{L-1}(i, k), \tag{4.18}$$

with the stopping case $d_0(i, i) = 1$ if $i \in \mathcal{Q}$, and 0 otherwise.

The structure of the DaKatz computation is very similar to the one of DaRWR, therefore, it can be efficiently implemented with SpMxV operations as follows: The scores are decayed and passed to the references and citations, rather than distributed among them, hence:

$$a_{ij} = \begin{cases} \beta(1 - \kappa), & \text{if } (i, j) \in E \\ \beta\kappa, & \text{if } (j, i) \in E \\ 0, & \text{otherwise,} \end{cases} \tag{4.19}$$

is used to build the structurally symmetric $n \times n$ transition matrix $\mathbf{A}$. There is no jumps to the query vertices; therefore, the linear equation in (4.30) is simplified to:

$$\mathbf{p}_t = \mathbf{A}\mathbf{p}_{t-1}, \tag{4.20}$$

where the Katz distance is initialized with $\mathbf{p}_0(i) = 1$ if $i \in \mathcal{Q}$, and 0 otherwise. The final relevance score $\mathbf{p}(i)$ of a vertex i aggregates all Katz distances with each path length up to L, i.e.,

$$\mathbf{p}(i) = \sum_{t=1}^{L} \mathbf{p}_t(i). \tag{4.21}$$

Even though the relevance scores $\mathbf{p}(i)$ monotonically increase with each iteration, the algorithm still converges because of the *decay* parameter β.

Implementations with Standard CRS (CRS-Full)

Assume that $\mathbf{A}$ is stored in CRS format and let A_{i*} be the ith row of $\mathbf{A}$. Algorithms 6 and 7 show the pseudocodes of DaRWR and DaKatz where (4.30) or (4.20) is computed at each iteration, respectively. Colored lines are used in order to distinguish the differences between the two computations.

Algorithm 6: DaRWR with CRS-Full

```
Input: $n \times n$ transition matrix $\mathbf{A}$ in CRS format, query paper set $\mathcal{Q}$
Output: relevance vector $\mathbf{p}$
$\mathbf{p}_t \leftarrow \mathbf{0}$
$\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow 1/|\mathcal{Q}|$
$e \leftarrow \|\mathbf{p}_t\|_2$
while $e > \xi$ do
    $\mathbf{p}_{t-1} \leftarrow \mathbf{p}_t$
    $\mathbf{p}_t \leftarrow \mathbf{0}$
    foreach paper $i = 1$ to $n$ do
        if $\mathbf{p}_{t-1}(i) > 0$ then
            foreach $a_{ij} \neq 0$ in $\mathbf{A}_{i*}$ do
                $\mathbf{p}_t(j) \leftarrow \mathbf{p}_t(j) + a_{ij}\mathbf{p}_{t-1}(i)$
    $\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + (1-d)/|\mathcal{Q}|$
    $e \leftarrow \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2$
return $\mathbf{p} \leftarrow \mathbf{p}_t$
```

Algorithm 7: DaKatz with CRS-Full

```
Input: $n \times n$ transition matrix $\mathbf{A}$ in CRS format, query paper set $\mathcal{Q}$
Output: relevance vector $\mathbf{p}$
$\mathbf{p}_t \leftarrow \mathbf{0}$   $\mathbf{p}_{total} \leftarrow \mathbf{0}$
$\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow 1$
$e \leftarrow \|\mathbf{p}_t\|_2$
while $e > \xi$ do
    $\mathbf{p}_{t-1} \leftarrow \mathbf{p}_t$
    $\mathbf{p}_t \leftarrow \mathbf{0}$
    foreach paper $i = 1$ to $n$ do
        if $\mathbf{p}_{t-1}(i) > 0$ then
            foreach $a_{ij} \neq 0$ in $\mathbf{A}_{i*}$ do
                $\mathbf{p}_t(j) \leftarrow \mathbf{p}_t(j) + a_{ij}\mathbf{p}_{t-1}(i)$
    $\mathbf{p}_{total} \leftarrow \mathbf{p}_{total} + \mathbf{p}_t$
    $e \leftarrow \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2$
return $\mathbf{p} \leftarrow \mathbf{p}_{total}$
```

To compute (4.30) or (4.20), one needs to read all of $\mathbf{A}$ at each iteration. Note that for each nonzero in $\mathbf{A}$, there is a possible update on $\mathbf{p}_t$. As described above, $\mathbf{A}$

contains $2|E|$ nonzeros which is approximately equal to 12×10^6. This size allows us to index rows and columns using 32-bit values. However, the probabilities and matrix entries are stored in 64-bit. Assuming it is stored in CRS format, the size of $\mathbf{A}$ in memory is roughly 147MB.

Implementations with Halved CRS (CRS-Half)

Here, we propose two modifications to reduce $\mathbf{A}$'s size and the number of multiplications required to update $\mathbf{p}_t$. The first modification is compressing the nonzeros in $\mathbf{A}$: we know that during an iteration, the contributions of paper i to the papers in its reference list are all equal to $c_t^-(i)$. Similarly, the contributions of i to the papers which cite i are equal to $c_t^+(i)$. Let $\mathbf{s}_R$ and $\mathbf{s}_C$ be the row and column scaling vectors defined for DaRWR and DaKatz as:

$$\text{(DaRWR)} \qquad\qquad\qquad\qquad \text{(DaKatz)}$$

$$\mathbf{s}_R(i) = \begin{cases} \frac{d(1-\kappa)}{\delta^+(i)} & \text{if } \delta^+(i) > 0 \\ 0 & \text{otherwise.} \end{cases} \qquad \mathbf{s}_R(i) = \begin{cases} \beta(1-\kappa) & \text{if } \delta^+(i) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.22}$$

$$\mathbf{s}_C(i) = \begin{cases} \frac{d\kappa}{\delta^-(i)} & \text{if } \delta^-(i) > 0 \\ 0 & \text{otherwise.} \end{cases} \qquad \mathbf{s}_C(i) = \begin{cases} \beta\kappa & \text{if } \delta^-(i) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.23}$$

Let $\mathbf{B}$ be the 0-1 adjacency matrix of G defined as

$$b_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{4.24}$$

Then (4.30) and (4.20) can be rewritten as

$$\mathbf{p}_t = \mathbf{r} + \mathbf{B}(\mathbf{s}_R * \mathbf{p}_{t-1}) + \mathbf{B}^T(\mathbf{s}_C * \mathbf{p}_{t-1}), \tag{4.25}$$

where $*$ denote the pointwise vector multiplication. In this form, the total size of $\mathbf{B}$, $\mathbf{B}^T$, $\mathbf{s}_R$, and $\mathbf{s}_C$ is roughly 71MB assuming we only store the indices of nonzeros in $\mathbf{B}$ and $\mathbf{B}^T$. This modification not only reduces the size of $\mathbf{A}$, but also decreases the number of multiplications required in each iteration. Here, we only need to do pointwise multiplications $\mathbf{s}_R * \mathbf{p}_{t-1}$ and $\mathbf{s}_C * \mathbf{p}_{t-1}$ before traversing the nonzero indices. Hence, we only need to do $2|V|$ multiplications per iteration. Assuming $\mathbf{p}_t(i) > 0$ for all $i \in V$, Algorithms 6 and 7 perform $2|E|$ multiplications. Hence this modification can lead up to 6 fold reduction on the number of multiplications on our dataset.

We can further reduce the memory usage by using the fact that $b_{ij} = 1$ if and only if $b^T_{ji} = 1$. We can only store $\mathbf{B}$, and when we read a nonzero b_{ij}, we can do the updates on $\mathbf{p}_t$ both for b_{ij} and b^T_{ji}. By not storing $\mathbf{B}^T$, we reduce the size roughly to 43MB. Furthermore, we actually read two nonzeros when we bring b_{ij} from the memory. However, we still need to do two different updates. A similar optimization has been proposed for some particular SpMxV operations [16]. Algorithms 8 and 9 show the pseudocodes of the DaRWR and DaKatz computation with the modifications described above.

Although the proposed modifications reduce the size of $\mathbf{A}$ and the number of multiplications, there is a drawback. In Algorithms 6 and 7, line 8 first checks if $\mathbf{p}_{t-1}(i) > 0$. If this is not the case there is no need to traverse any of the a_{ij}s. This shortcut is especially useful when $\mathbf{p}_{t-1}$ contains only a few positive values which is the case for the first few iterations. However, such a shortcut only works for nonzeros corresponding to the outgoing edges when the matrix is reduced. That is, if b_{ij} is nonzero, Algorithm 8 does the update $\mathbf{p}_t(j) \leftarrow \mathbf{p}_t(j) + \mathbf{sp}_C(i)$ even though $\mathbf{sp}_C(i)$ is zero. Hence, some updates with no effects are done in Algorithms 8 and 9, although they are skipped in Algorithms 6 and 7.

Algorithm 8: DaRWR with CRS-Half

```
Input: $n \times n$ adjacency matrix $\mathbf{B}$ in CRS format, query paper set $\mathcal{Q}$, row and column scaling vectors $\mathbf{s}_R$, $\mathbf{s}_C$
Output: relevance vector $\mathbf{p}$
$\mathbf{p}_t \leftarrow \mathbf{0}$
$\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow 1/|\mathcal{Q}|$
$e \leftarrow \|\mathbf{p}_t\|_2$
while $e > \xi$ do
    $\mathbf{sp}_R \leftarrow \mathbf{p}_t * \mathbf{s}_R$
    $\mathbf{sp}_C \leftarrow \mathbf{p}_t * \mathbf{s}_C$
    $\mathbf{p}_{t-1} \leftarrow \mathbf{p}_t$
    $\mathbf{p}_t \leftarrow \mathbf{0}$
    foreach paper $i = 1$ to $n$ do
        if $\mathbf{sp}_C(i) > 0$ then
            foreach $b_{ij} \neq 0$ in $\mathbf{B}_{i*}$ do
                $\mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + \mathbf{sp}_R(j)$
                $\mathbf{p}_t(j) \leftarrow \mathbf{p}_t(j) + \mathbf{sp}_C(i)$
        else
            foreach $b_{ij} \neq 0$ in $\mathbf{B}_{i*}$ do
                $\mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + \mathbf{sp}_R(j)$
    $\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + (1-d)/|\mathcal{Q}|$
    $e \leftarrow \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2$
return $\mathbf{p} \leftarrow \mathbf{p}_t$
```

Algorithm 9: DaKatz with CRS-Half

```
Input: $n \times n$ adjacency matrix $\mathbf{B}$ in CRS format, query paper set $\mathcal{Q}$, row and column scaling vectors $\mathbf{s}_R$, $\mathbf{s}_C$
Output: relevance vector $\mathbf{p}$
$\mathbf{p}_t \leftarrow \mathbf{0}$   $\mathbf{p}_{total} \leftarrow \mathbf{0}$
$\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow 1$
$e \leftarrow \|\mathbf{p}_t\|_2$
while $e > \xi$ do
    $\mathbf{sp}_R \leftarrow \mathbf{p}_t * \mathbf{s}_R$
    $\mathbf{sp}_C \leftarrow \mathbf{p}_t * \mathbf{s}_C$
    $\mathbf{p}_{t-1} \leftarrow \mathbf{p}_t$
    $\mathbf{p}_t \leftarrow \mathbf{0}$
    foreach paper $i = 1$ to $n$ do
        if $\mathbf{sp}_C(i) > 0$ then
            foreach $b_{ij} \neq 0$ in $\mathbf{B}_{i*}$ do
                $\mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + \mathbf{sp}_R(j)$
                $\mathbf{p}_t(j) \leftarrow \mathbf{p}_t(j) + \mathbf{sp}_C(i)$
        else
            foreach $b_{ij} \neq 0$ in $\mathbf{B}_{i*}$ do
                $\mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + \mathbf{sp}_R(j)$
    $\mathbf{p}_{total} \leftarrow \mathbf{p}_{total} + \mathbf{p}_t$
    $e \leftarrow \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2$
return $\mathbf{p} \leftarrow \mathbf{p}_{total}$
```

Implementations with Halved COO Storage (COO-Half)

If we apply the optimizations described for the halved CRS, one needs roughly 63MB in memory to store $\mathbf{B}$ in COO format. In this format, the nonzeros are read one by one. Hence, a shortcut for the updates with no effect is not practical. On the other hand, with COO, we have more flexibility for nonzero ordering, since we do not need to store the nonzeros in a row consecutively. Furthermore, techniques like blocking can be implemented with much less overhead. We give the COO-based pseudocodes of DaRWR and DaKatz in Algorithms 10 and 11.

Algorithm 10: DaRWR with COO-Half

```
Input: $n \times n$ adjacency matrix $\mathbf{B}$ in COO format, query paper set $\mathcal{Q}$, row and column scaling vectors $\mathbf{s}_R$, $\mathbf{s}_C$
Output: relevance vector $\mathbf{p}$
$\mathbf{p}_t \leftarrow \mathbf{0}$
$\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow 1/|\mathcal{Q}|$
$e \leftarrow \|\mathbf{p}_t\|_2$
while $e > \xi$ do
    $\mathbf{sp}_R \leftarrow \mathbf{p}_t * \mathbf{s}_R$
    $\mathbf{sp}_C \leftarrow \mathbf{p}_t * \mathbf{s}_C$
    $\mathbf{p}_{t-1} \leftarrow \mathbf{p}_t$
    $\mathbf{p}_t \leftarrow \mathbf{0}$
    foreach nonzero $b_{ij}$ of $\mathbf{B}$ do
        $\mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + \mathbf{sp}_R(j)$
        $\mathbf{p}_t(j) \leftarrow \mathbf{p}_t(j) + \mathbf{sp}_C(i)$
    $\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + (1-d)/|\mathcal{Q}|$
    $e \leftarrow \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2$
return $\mathbf{p} \leftarrow \mathbf{p}_t$
```

Algorithm 11: DaKatz with COO-Half

```
Input: $n \times n$ adjacency matrix $\mathbf{B}$ in COO format, query paper set $\mathcal{Q}$, row and column scaling vectors $\mathbf{s}_R$, $\mathbf{s}_C$
Output: relevance vector $\mathbf{p}$
$\mathbf{p}_t \leftarrow \mathbf{0}$   $\mathbf{p}_{total} \leftarrow \mathbf{0}$
$\forall i \in \mathcal{Q},\ \mathbf{p}_t(i) \leftarrow 1$
$e \leftarrow \|\mathbf{p}_t\|_2$
while $e > \xi$ do
    $\mathbf{sp}_R \leftarrow \mathbf{p}_t * \mathbf{s}_R$
    $\mathbf{sp}_C \leftarrow \mathbf{p}_t * \mathbf{s}_C$
    $\mathbf{p}_{t-1} \leftarrow \mathbf{p}_t$
    $\mathbf{p}_t \leftarrow \mathbf{0}$
    foreach nonzero $b_{ij}$ of $\mathbf{B}$ do
        $\mathbf{p}_t(i) \leftarrow \mathbf{p}_t(i) + \mathbf{sp}_R(j)$
        $\mathbf{p}_t(j) \leftarrow \mathbf{p}_t(j) + \mathbf{sp}_C(i)$
    $\mathbf{p}_{total} \leftarrow \mathbf{p}_{total} + \mathbf{p}_t$
    $e \leftarrow \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2$
return $\mathbf{p} \leftarrow \mathbf{p}_{total}$
```

4.2.3 Exploiting cache locality in reduced matrix operations

As explained in the previous section, one of the techniques we use for compressing the matrix is to store $\mathbf{B}$, but not $\mathbf{B}^T$. After this modification, when a nonzero b_{ij} is read, $\mathbf{p}_t(i)$ and $\mathbf{p}_t(j)$ are updated accordingly. Hence, when we order $\mathbf{B}$'s rows with a permutation matrix $\mathbf{P}$, we need to use the same $\mathbf{P}$ to order the columns if we want to find the nonzero indices in $\mathbf{B}^T$. Also, using the same permutation allow a simpler implementation of the iterative SpMxV operations. Note that although $\mathbf{s}_R$ and $\mathbf{s}_C$ can be permuted with different row and column permutations, we only have a single $\mathbf{p}_t$ array to process both b_{ij} and its transposed counterpart b_{ji} as shown in Figure 4.7. Due to these reasons, permuting the adjacency matrix as $\mathbf{B}' = \mathbf{PBP}^T$ is good practice for our problem. Note that the original SpMxV problem does not have such a restriction. Hence, existing hypergraph-partitioning-based approaches cannot

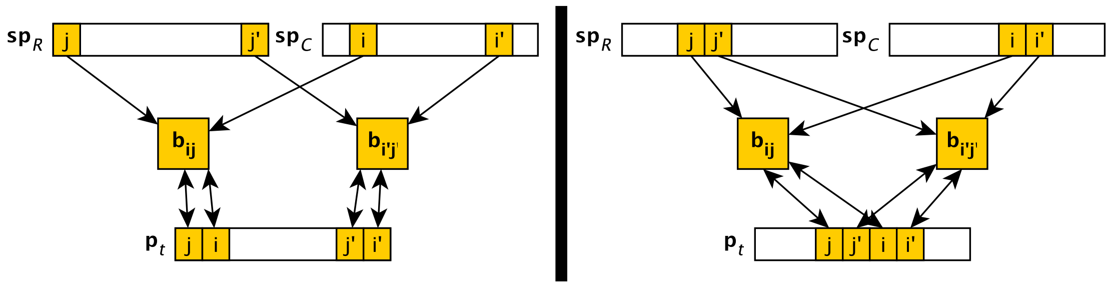


Figure 4.7: Two memory-access scenarios with different row/column permutations and nonzero orderings while processing b_{ij} and $b_{i'j'}$ consecutively. The left scenario has a poor cache locality, since the locations accessed in $\mathbf{s}_R$, $\mathbf{s}_C$, and $\mathbf{p}_t$ are far from each other. On the other hand, in the right scenario, the locations are close to each other. Thus, with high probability, the required values to process $b_{i'j'}$ will already be in the cache after processing b_{ij}.

be directly applied for our problem [5, 142, 143]. Note that we can still use symmetric permutations such as the ones obtained by RCM, AMD, and SlashBurn.

Similar to existing partitioning-based approaches, we use a two-phase permutation strategy which first partitions the rows of $\mathbf{B}$ into K parts and sorts them in the increasing order of their part numbers. The intra-part row ordering is decided later by using RCM, AMD, or SlashBurn, and the final permutation matrix $\mathbf{P}$ is obtained. Our column-net hypergraph $\mathcal{H}_{\mathcal{R}} = (\mathcal{V}_{\mathcal{R}}, \mathcal{N}_{\mathcal{C}})$ is created with n vertices and n nets corresponding to the rows and columns of $\mathbf{B}$, respectively, as described in Section 4.2.1. In $\mathcal{H}_{\mathcal{R}}$, two vertices v_i and $v_{i'}$ are connected via a net η_j if both b_{ij} and $b_{i'j}$ is equal to 1. To handle the above mentioned restriction of using the same permutation for rows and columns, we set $v_i \in \eta_i$ for all $i \in \{1, \ldots, n\}$. That is, we set all diagonal entries of $\mathbf{B}$, which originally has a zero diagonal, to 1 and partition it. With this modification, a net j can be internal if and only if the pins of j are in the same part with vertex j. Hence, when we permute the rows and columns with respect to the part numbers of the rows, the columns corresponding to the internal nets of a part will be accessed by the rows only in that part.

Since we store the matrix in CRS format, we know that $\mathbf{sp}_C$ is accessed sequentially (even for COO-Half, our nonzero ordering respects to row indices to some degree). Hence, accessing to $\mathbf{p}_t$ and $\mathbf{sp}_R$ with column indices will possibly be the main bottleneck. We use PaToH [22] to minimize *connectivity* $-$ 1 metric (4.9) and improve cache locality. Throughout the computation, the entry $\mathbf{sp}_R(j)$ will be put to cache at least once assuming the jth column has at least one nonzero in it. If column j is internal to part ℓ then $\mathbf{sp}_R(j)$ will be only accessed by the rows within part ℓ (e.g., nets n_1, n_2, n_3, and n_4, and the corresponding columns in Figure 4.6). Since the internal columns of each part are packed close in the permutation, when $\mathbf{sp}_R(j)$ is put to the cache, the other entries of $\mathbf{sp}_R$ which are part of the same cache line are likely to be internal columns of the same part. On the other hand, when an external column j is accessed by a part ℓ' which is not the part of j, the cache line containing $\mathbf{sp}_R(j)$ is unlikely to contain entries used by the rows in part ℓ' (n_4 and the fourth column in Figure 4.6). Minimizing the *connectivity* $-$ 1 metric equals to minimizing the number of such accesses. Note that the same is true for the access of $\mathbf{p}_t$ with column indices.

We find intra-part row/column orderings by using RCM, AMD, and SlashBurn where RCM and AMD have previously been used for fill-in minimization in sparse LU factorization. RCM is used to find a permutation σ which reduces the bandwidth of a symmetric matrix $\mathbf{A}$ where the bandwidth is defined as $b = \max(\{|\sigma(i) - \sigma(j)| : a_{ij} \neq 0\})$. When the bandwidth is small, the entries are close to the diagonal, and the cache locality will be high. The AMD heuristic also has the same motivation of minimizing the number of fill-ins, which usually densifies the nonzeros in different parts of the matrix. Since having nonzeros close to each other is good for cache locality, we used these heuristics to order rows and columns inside each part.

The last ordering heuristic, SlashBurn, has been proposed for matrix compression, i.e., to reduce the number of fixed-size tiles required to cover all the nonzeros in the matrix, which also implies a reduced number cache-misses. For several social and web graphs, SlashBurn is proven to be very effective [62]. However, its complexity is larger than that of RCM and AMD, and as a result, it is much slower in practice. Since the ordering will be executed only once as a preprocessing phase of the the**advisor**, for our application, we can ignore its complexity in the evaluation and concentrate on its benefits on the query response time.

For all the algorithms described in Section 4.2.2, we used the proposed ordering approach. For CRS-Full, we permuted **A**, and for CRS-Half and COO-Half, we permuted **B** as described above. For COO-Half, we also apply blocking after permuting **B**: we divide **B** into square blocks of size 1024×1024 and traverse the nonzeros with respect to their block ids (and row-wise within a block). The block size is tuned on the architecture the**advisor** is running on.

4.2.4 Experimental Results

The setup for the experiments can be summarized as follows:

Architectures: We used three different architectures to test the algorithms. The target architecture (**Arch1**) has a 2.4GHz *AMD Opteron* CPU and 4GB of main memory. The CPU has 64KB L1 and 1MB L2 caches. Our service, the**advisor**, is currently running on a cluster with 50 nodes each having the above mentioned architecture. For each query, the service opens a socket to a running process, submits the query, and returns the results to the user. For completeness, we also test the algorithms on two more recent architectures. The second architecture (**Arch2**) has a 2.4GHz quad-core *AMD Opteron* (Shanghai) CPU and 32GB of main memory. Each core has 64KB L1 and 512KB L2 cache and each socket has a 6MB L3 cache. The third architecture (**Arch3**) has a 2.27GHz quad-core *Intel Xeon* (Bloomfield) CPU and 48GB of main memory. Each core has 32KB L1 and 256KB L2 caches and each socket has an 8MB L3 cache.

Implementation: All of the algorithms are implemented in C++. The compiler `gcc` and the `-O2` optimization flag is used. For the experiments, we use only one core from each processor.

Queries: We generated 286 queries where each query is a set $\mathcal{Q}$ of paper ids obtained from the bibliography files submitted by the users of the service who agreed to donating their queries for research purposes. The number of query papers, $|\mathcal{Q}|$, vary between 1 and 449, with an average of 24.7.

Parameters: For DaRWR, we use $d = 0.8$ and $\kappa = 0.75$ which are the default values in the**advisor**. For DaKatz, we use $\beta = 0.005$. While generating the partitions, we set the imbalance ratio of PaToH to 0.4.

Convergence: We did not use a threshold ξ for convergence. We observed that DaRWR in our citation graph takes about 20, and DaKatz takes about 10 iterations to converge (see Fig. 4.8). Computing the error between iterations takes some time, and since we want to be consistent in the experiments, we let the algorithms iterate a fixed number of times.

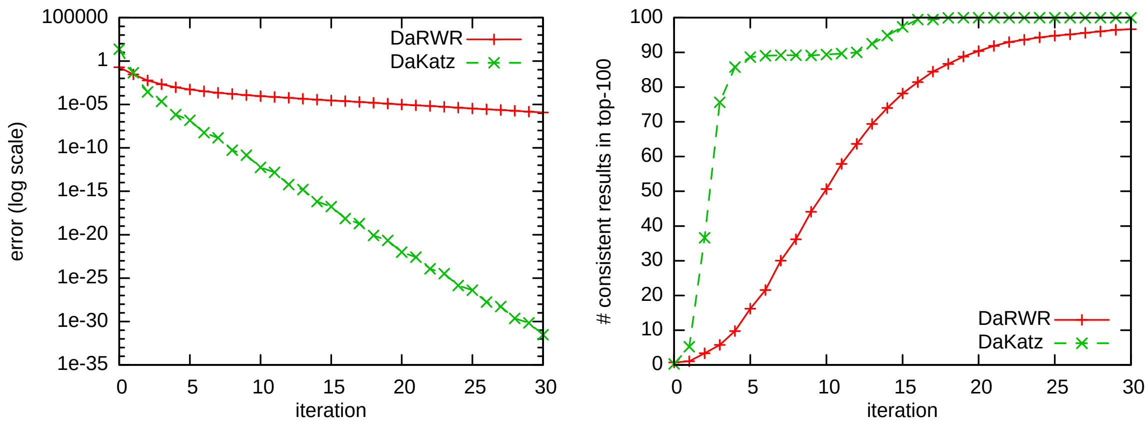


Figure 4.8: Errors and number of consistent results within top-100 when DaRWR and DaKatz is run with the given number of iterations.

Effects of the storage schemes on the number of updates

As mentioned in Section 4.2.2, the algorithms CRS-Full and CRS-Half can avoid some updates but COO-Half cannot, even they have no effect on $\mathbf{p}_t$. In our query set, the average number of papers is 24.7. In the first iteration, $\mathbf{p}_{t-1}$ has only a few positive values on average, and CRS-Full updates $\mathbf{p}_t$ only for the corresponding papers in $\mathcal{Q}$. Since $n \gg 24.7$, CRS-Full avoids roughly 12 million nonzeros/updates in the first iteration. This number is roughly 6 million for CRS-Half. COO-Half traverses all 12 million nonzeros and does the corresponding updates even if most of them have no effect for the first couple of iterations. However, the number of positive values in $\mathbf{p}_{t-1}$ increases exponentially. As Fig. 4.9 shows, the shortcuts in CRS-based algorithms are not useful after the 8th iteration. The figure also implies that the citation graph is highly connected since DaRWR and DaKatz seem to traverse almost all the nonzeros in $\mathbf{A}$. That is, random walks and paths can reach to

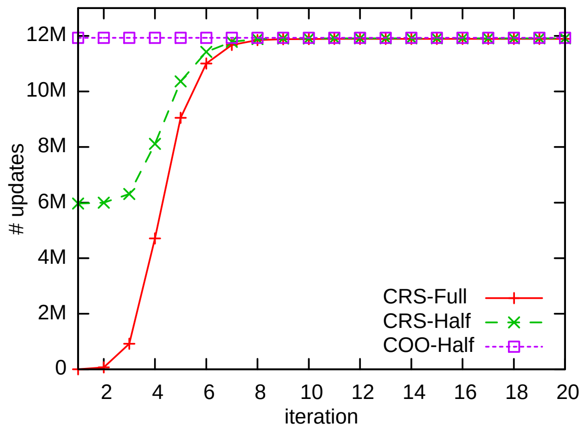
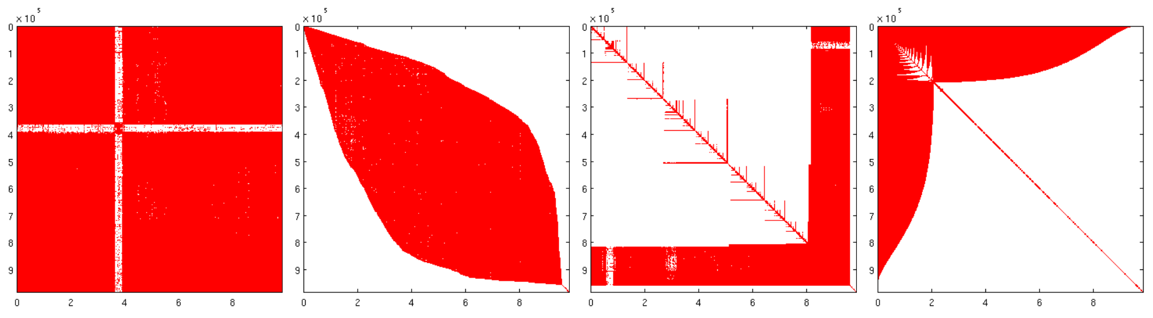


Figure 4.9: Number of updates per iteration of the algorithms.

almost all vertices in the graph. We observed that 97% of the vertices of the citation graph G are in a single connected component.

Effects of partitioning and ordering on nonzero patterns

The nonzero pattern of the adjacency matrix $\mathbf{B}$ is given in Fig. 4.10(a). As the figure shows, the nonzeros are distributed in all the matrix. In our experiments, the papers are originally numbered with respect to the order we parse their metadata. When $\mathbf{B}$ is ordered by using the RCM heuristic, the nonzero pattern (Fig. 4.10(b)) is densified near the diagonal as expected. The bandwidths of the original and RCM ordered $\mathbf{B}$ matrices are $981,287$ and $460,288$, respectively. Although the bandwidth is reduced more than half, it is still large. Figure 4.10(c) shows the nonzero pattern of $\mathbf{B}$ when ordered with the AMD heuristic. The nonzeros are densified inside one horizontal and one vertical block. We observed that 80% of the nonzeros are inside this region. As the figure shows, the remaining nonzeros are located in smaller horizontal and vertical regions which also may be helpful to reduce the number of cache misses.



(a) Original (b) RCM (c) AMD (d) SlashBurn

Figure 4.10: The nonzero pattern of $\mathbf{B}$ (a) when ordered with RCM (b), AMD (c), and SlashBurn (d). Nonzeros are colored with red and white areas show empty regions.

The pattern of SlashBurn also possesses similar characteristics: Figure 4.10(d) shows the arrow-shaped pattern obtained after ordering $\mathbf{B}$ with SlashBurn. All the nonzeros are densified inside the pattern, and the number of cache misses is expected to be much less.

As described in Section 4.2.3, we first partition $\mathbf{B}$ in the column-net model to reorder it. To do that, we use $K = \{2, 4, 8, 16, 32, 64\}$ and create 6 different partitions. For each partition, we create a permutation matrix $\mathbf{P}$ and reorder $\mathbf{B}$ as $\mathbf{B}' = \mathbf{P}\mathbf{B}\mathbf{P}^T$. The left-most images in Figs. 4.11(a)–4(c) show the structure of the nonzero pattern of $\mathbf{B}'$ for $K = 2$, 4, and 8, respectively. In the figures, the horizontal (vertical) lines separate the rows (columns) of the matrix w.r.t. their part numbers. The diagonal blocks in the figure contain the nonzeros b_{ij}s where the ith and jth row of $\mathbf{B}$ are assigned to the same part. We permute the rows and columns of these blocks by using the ordering heuristics RCM, AMD, and SlashBurn. Figure 4.11 also shows the nonzero patterns of these further permuted matrices for each partition with $K = 2$, 4, and 8.

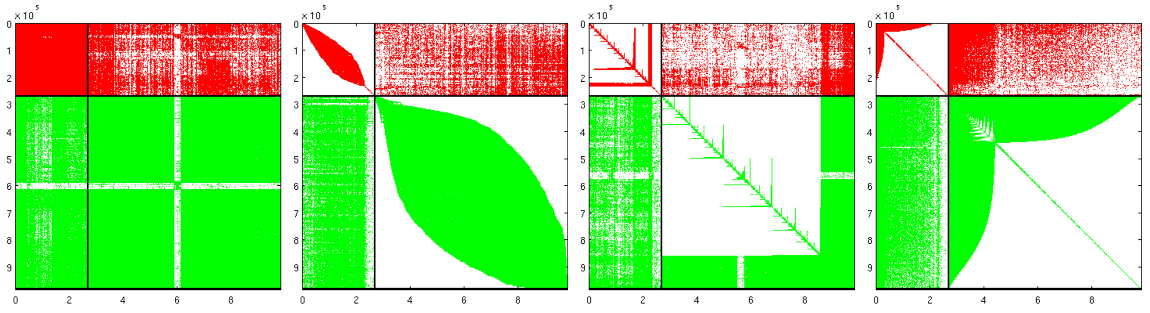
Performance analysis

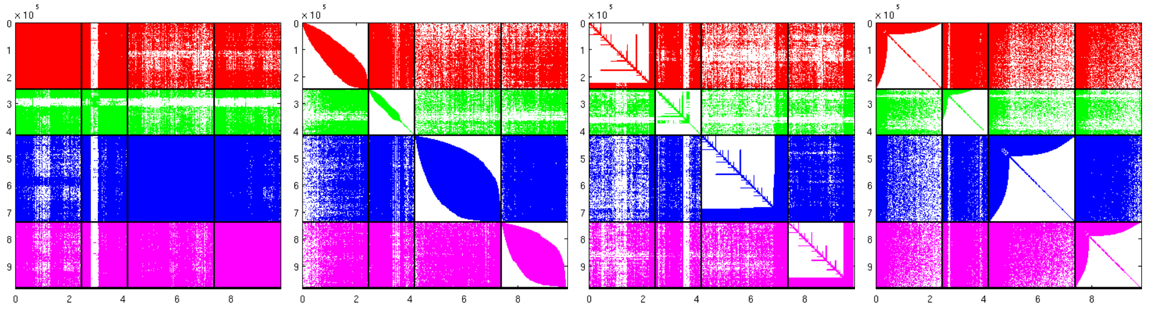
Figures 4.12(a), (b), and (c) show the number of updates per seconds (*nups*) for each algorithm when RCM, AMD, and SlashBurn are used, respectively. This experiment counts the number of updates that occur in memory, even if they are nilpotent, i.e., they do not change a value. The configuration which takes advantage of partitioning most is the COO-Half equipped with AMD ordering for which *nups* increases from 270 million to 340 million. As the figure shows, SlashBurn does not bring a performance improvement relative to AMD and RCM. Hence, considering its complexity, we can suggest that using AMD and RCM is more practical, especially when the time spent for the ordering is important.

Table 4.5: Number of nonzeros inside and outside of the diagonal blocks of $\mathbf{B}'$ after reordering.

K	2	4	8	16	32	64
nnz in	5.62M	4.92M	4.39M	3.80M	3.46M	2.95M
nnz out	0.34M	1.04M	1.57M	2.16M	2.50M	3.01M

Although the nonzeros of $\mathbf{B}'$ seem evenly distributed in Fig. 4.11, as Table 4.5 shows, for $K = 2$ and 4, the percentage of the nonzeros inside diagonal blocks of $\mathbf{B}'$ are 94% and 83%, respectively. Hence, the nonzeros are densified inside these blocks as expected. The number of nonzeros outside the diagonal blocks is more when K increases. However, the diagonal blocks tend to get smaller and denser which may


(a) $\mathbf{B}'$ (left) with RCM, AMD, and SlashBurn for $K = 2$


(b) $\mathbf{B}'$ (left) with RCM, AMD, and SlashBurn for $K = 4$

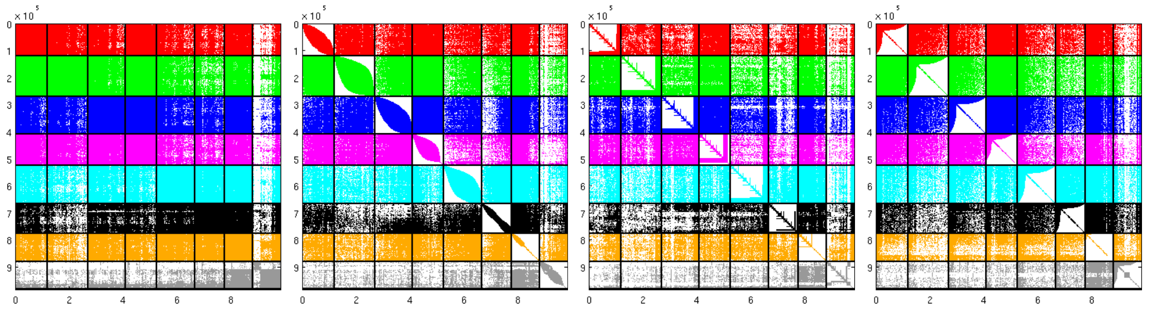
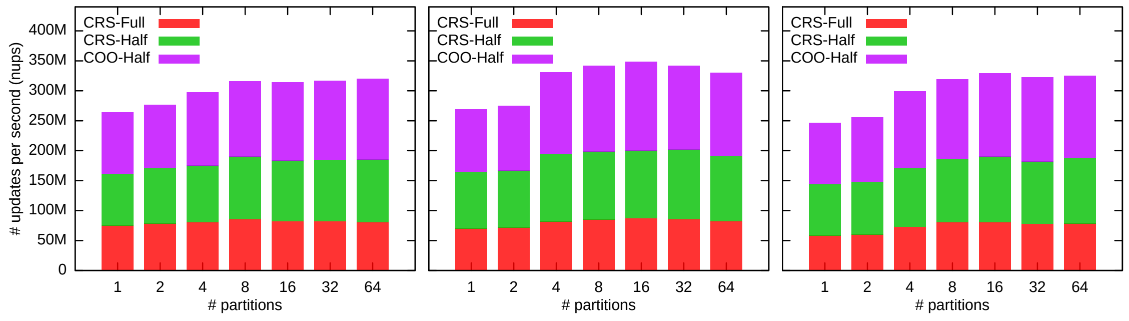

(c) $\mathbf{B}'$ (left) with RCM, AMD, and SlashBurn for $K = 8$

Figure 4.11: The nonzero pattern of the permuted adjacency matrix $\mathbf{B}$ with different partitions and ordering heuristics. The row set of each part is shown with a different color. The diagonal blocks contain b_{ij}s where row i and row j of $\mathbf{B}$ are in the same part.



(a) RCM (b) AMD (c) SlashBurn

Figure 4.12: Number of updates per second (*nups*) for each algorithm and ordering on **Arch1** when DaRWR is executed for 20 iterations.

improve the cache locality. As Figures 4.12 shows, a consistent improvement can be observed for all the algorithms CRS-Full, CRS-Half, and COO-Half, and ordering heuristics RCM, AMD, and SlashBurn up to $K = 8$ and 16. But when K gets more than 16, no significant improvement can be observed and the performance can even get worse. There are two main reasons for this phenomena: first, the fraction of block-diagonal nonzeros continues to decrease, and second, the diagonal blocks becomes smaller than required. That is the cache locality may be optimized to the most, hence, a further reduction on the block size is unnecessary. Here, the best K in terms of performance depends on the configuration. We tested the algorithms with different K values to find the best configuration. As the figure shows, with CRS-Full, the maximum *nups* is around 80 million for $K = 8$ (RCM), 16 (AMD), and $K = 16$ (SlashBurn). By compressing the matrix increases the *nups* for CRS-Half up to 190 million with $K = 16$ (AMD). And with blocking used in COO-Half, *nups* increases to 340 million with $K = 16$ (AMD) which is the maximum for this experiment.

The *nups* of COO-Half seems to be much superior in Fig. 4.12. However, the algorithm itself needs to do more computation since it cannot skip any of the updates even if they are nilpotent. To compare the actual response times of the configurations, we used 286 queries and measured the average response time on each architecture. Figure 4.13 shows the execution times of DaRWR and DaKatz for different algorithms, Ks, and ordering heuristics on the target architecture **Arch1**. For the rest of the figures, SB denotes the SlashBurn ordering heuristic.

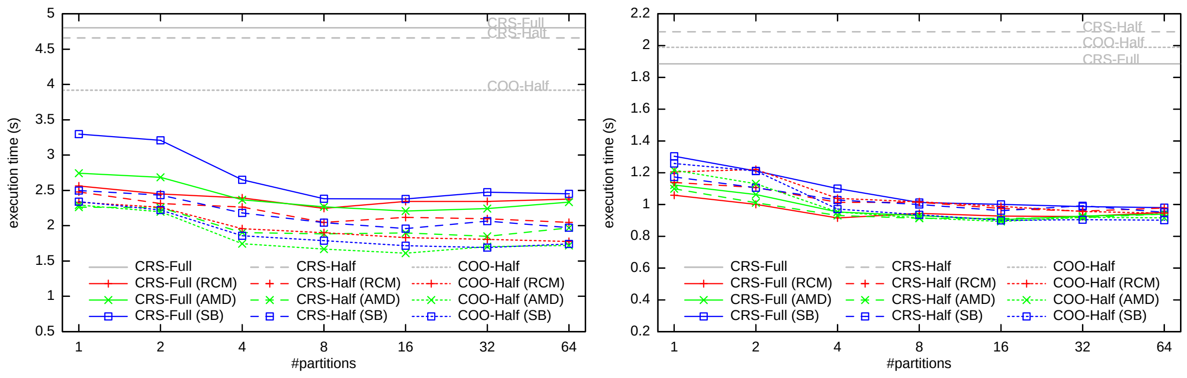


Figure 4.13: Execution times in seconds for DaRWR (left) and DaKatz (right) using each algorithm with different Ks and ordering heuristics on **Arch1**. The values are the averages of the running times for 286 queries. For each query, the algorithms perform 20 DaRWR iterations and 10 DaKatz iterations.

As concordant with *nups* values, for DaRWR, the fastest algorithm is COO-Half where $K = 16$ and the diagonal blocks are ordered with AMD. The average query response time for this configuration, which is being used in the**advisor**, is 1.61 seconds. Compared with the execution time of CRS-Full with the original ordering, which is 4.80 seconds, we obtain 3 times improvement. When $K = 1$, i.e., if there is no partitioning, the execution time of COO-Half is 2.29. Hence, we obtain a speedup of more than 2 due to ordering and 42% additional improvement due to partitioning.

For DaKatz experiment on **Arch1**, the best configuration is not very visible. In our experiments, the minimum average query response time, 0.89 seconds, is obtained again by COO-Half when it is equipped with AMD and when K is 16. On the other hand, the best CRS-Full configuration answers a query in 0.90 seconds on average (with AMD and $K = 16$). Although the ordering and partitioning still help a lot, the improvements due to algorithmic modifications are only minor for DaKatz. As described above and shown by Fig. 4.9, the number of updates required by CRS-Full only matches that of COO-Half after the 8th iteration and DaKatz converges only in 10 iterations for our citation graph. That is the total work required by CRS-Full is much less than both CRS-Half and COO-Half for DaKatz. Hence, the update overhead cannot be easily compensated by reducing the bandwidth and improving cache locality. Still, the average query response time is reduced from 1.89 to 0.89 thanks to ordering and partitioning.

For completeness, in Fig. 4.14, we give the results when 20 DaKatz iterations are performed. On all architectures, CRS-Full configurations are much slower than CRS-Half and COO-Half configurations as expected. And the differences are more visible. Furthermore, the relative performance of DaKatz algorithms is more similar to that of DaRWR algorithms with 20 iterations.

We tested our modifications on two other architectures described above. As shown in Fig. 4.15, the results are similar for DaRWR on **Arch2**. COO-Half with AMD is the fastest configuration, but this time with $K = 8$. However, matrix compression

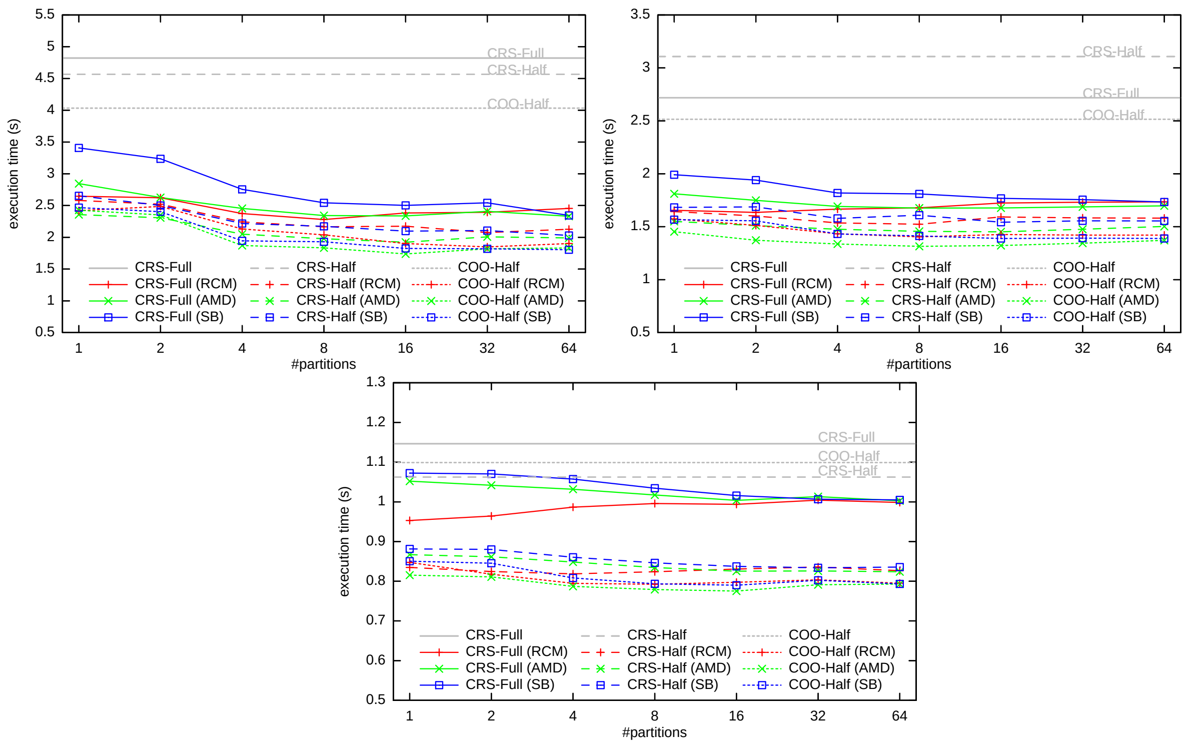
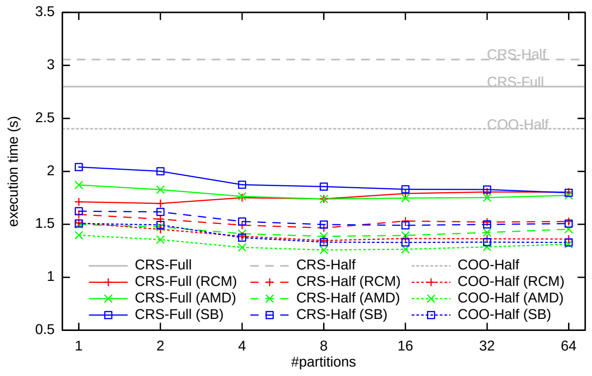


Figure 4.14: Execution times of DaKatz algorithms in seconds on architectures on **Arch1** (a), **Arch2** (b), and **Arch3** (c) with different Ks and ordering heuristics. The values are the averages of the running times for 286 queries. For each query, 20 DaKatz iterations are performed.



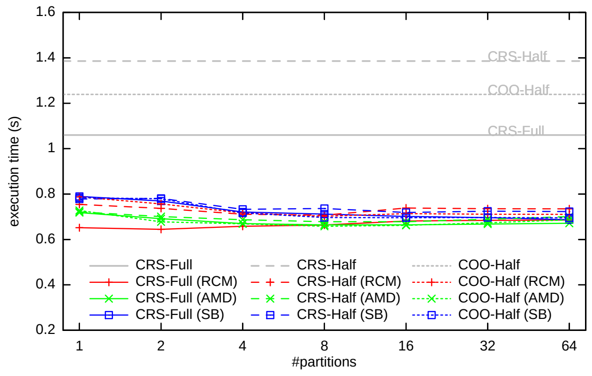
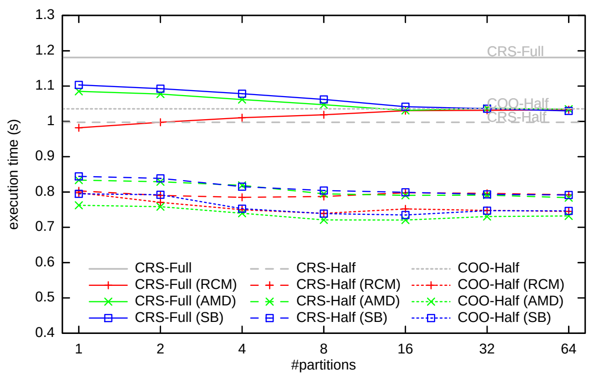


Figure 4.15: Execution times in seconds for DaRWR (left) and DaKatz (right) using each algorithm with different Ks and ordering heuristics on **Arch2**. The values are the averages of the running times for 286 queries. For each query, the algorithms perform 20 DaRWR iterations and 10 DaKatz iterations.

seems to have a negative effect on CRS-Half. Its execution time is more than CRS-Full with the original ordering. This is unexpected since both on **Arch1** (Fig. 4.13) and **Arch3** (Fig. 4.16), CRS-Half is faster than CRS-Full. On **Arch3**, the fastest DaRWR algorithm is again COO-Half where the average query response time is 0.72 seconds with $K = 8$ and the AMD heuristic. Compared to the time of CRS-Full based DaRWR with no ordering, which is 1.18 seconds, the improvement is 39%. If we apply only matrix compression with COO-Half, the query response time is 1.04 seconds. Hence, we can argue that we obtained 31% improvement by permuting the reduced matrix. If we only use AMD, i.e., when $K = 1$, the query response time of COO-Half is 0.76. This implies roughly 5% improvement due to partitioning alone. Since **Arch3**'s cache is larger than the others, we believe that when the matrix gets large, i.e., when the number of papers in our database increases, the improvements will be much higher on all architectures but especially on the last architecture.



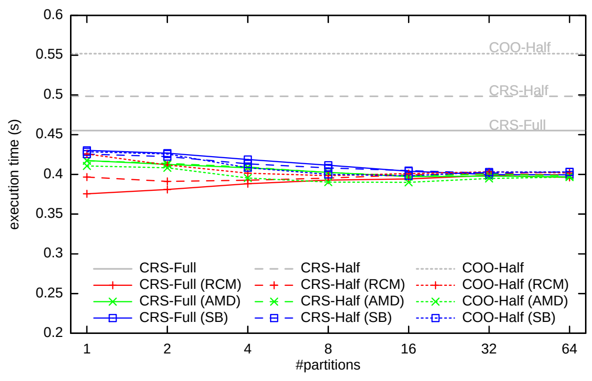


Figure 4.16: Execution times in seconds for DaRWR (left) and DaKatz (right) using each algorithm with different Ks and ordering heuristics on **Arch3**. The values are the averages of the running times for 286 queries. For each query, the algorithms perform 20 DaRWR iterations and 10 DaKatz iterations.

For DaKatz on **Arch2** (Fig. 4.15) and **Arch3** (Fig. 4.16), the best algorithm is clearly CRS-Full. The RCM ordering yields 40% (0.45 to 0.37 seconds) and 18% (1.06 to 0.65 seconds) improvements on the average query response time, respectively. However, the partitioning is not useful in practice for DaKatz on these architectures since it improves the query response times of other configurations but has a negative effect on CRS-Full with RCM. A similar pattern is also visible for the same DaRWR and DaKatz configurations especially on architectures **Arch2** and **Arch3**. The partitioning and the corresponding permutation on **B** are designed while taking the halved matrix into account: an access to a nonzero b_{ij} yields also the processing of b_{ji}. That is, the nonzeros to be processed are coming from a two-dimensional region. Hence, having the nonzeros inside diagonal blocks, COO-Half should be the algorithm which utilizes the optimizations the most, especially considering its blocked access pattern. On the contrary, for the same reasons, CRS-Full should be the worst algorithm for exploiting the optimizations, since the upcoming accesses in CRS-Full are only in column dimension. Furthermore, partitioning and permutation increase the bandwidth of the matrix which is arguably the most important criterion for CRS-Full. Since RCM aims to minimize the bandwidth of the matrix, the performance can be reduced when an additional partitioning is used. On the other hand, when the cache is not enough or the bandwidth is still large after RCM, partitioning may take effect and improve the performance. This can be observed in Fig. 4.13 for the target architecture **Arch1** which has 6 and 8 times less cache than **Arch2** and **Arch3**, respectively.

Considering the number of updates of CRS-Full is much less than COO-Half for the first few iterations, it is expected to be faster than COO-Half. On the other hand, when 20 DaKatz iterations are performed instead of 10, COO-Half with AMD is again the best configuration as shown in Fig. 4.14.

4.2.5 Summary

In this work, we proposed an efficient implementation of an SpMxV-type problem which arises in our publicly available citation, venue, and expert recommendation service, the**advisor**. We proposed compression and bandwidth reduction techniques to reduce the memory usage and hence, the bandwidth required to bring the matrix from the memory at each iteration. We also used matrix ordering techniques to reduce the number cache misses. Experimental results show that these modifications greatly help to reduce the query execution time.

As a future work, we are planning to develop new ideas to further reduce the query response time. As far as the service is running, this will be one of the tasks we are interested in. Note that in SpMxV operations, it is very hard to obtain linear speedup with shared memory parallelization. Hence, to maximize the throughput we chose to use one processor per query. However, we believe that such parallelism can still be effective for the**advisor** especially when the number concurrent requests is less than the number of processors allocated in the cluster.

Another work we are interested in is to make the service much faster via a hybrid implementation of DaRWR (or DaKatz) which uses a combination of CRS-Full, CRS-Half, and COO-Half. In its simple form, the hybrid approach can use CRS-Full in the first few iterations then switch to COO-Half to utilize the efficiency of the algorithms to the most. The overhead of such a scheme is storing the citation matrix multiple times and a transition cost incurred while switching from one algorithm to another. We believe that a hybrid implementation is promising and we aim to do a thorough investigation in the near future.

4.3 Diversifying Bibliographic Search

Diversifying the results of the search process is necessary to increase the amount of information one can reach via an automized search. This study targets the problem of result diversification in citation-based bibliographic search, assuming that the citation graph itself is the only information available, and no categories or intents are known. The contributions of this work is three-fold: We survey various random-walk-based diversification methods and enhance them with the *direction awareness* property to allow the users to reach either old, well-cited, well-known research papers or recent, less-known ones. Next, we propose a set of novel algorithms based on vertex selection and query refinement. A set of experiments with various evaluation criteria shows that the proposed γ-RLM algorithm performs better than the existing approaches and is suitable for real-time bibliographic search in practice.

4.3.1 Introduction

Diversifying the results of the search process is an important task to increase the amount of information one can reach via an automized search tool. There exists many recommender systems that personalize the output with respect to user's query/history. For several applications *personalization* can be an important limitation while reaching all the relevant information [36], and *diversification* can be used to increase the coverage of the results and hence, improve user satisfaction [4, 30, 46, 101].

Most diversification studies in the literature rely on various assumptions, e.g., items and/or queries are categorized beforehand [135], or there is a known distribution that specifies the probability of a given query belonging to some categories [4]. In the context of information retrieval or web search, since the search queries are often *ambiguous* or *multifaceted*, a query should represent the *intent* of an average user with a probability distribution [135]. Intent-aware methods in the literature aim to cover various relevant categories with one or more objects.

In this work, we target the bibliographic search problem assuming that the citation graph itself is the only information we have, and no categories or intents are available. Hence, we aim to diversify the results of the citation/paper recommendation process with the following objectives in mind: (1) the direction awareness property is kept, (2) the method should be efficient enough to be computable in real time, and (3) the results are relevant to the query and also diverse among each other. The contribution of this work is three-fold:

- We survey various random-walk-based diversity methods (i.e., GrassHopper [147], DivRank [101], Dragon [128]) and enhance them with the direction awareness property.

- We propose new algorithms based on vertex selection (LM, γ-RLM) and query refinement (GSparse).

- We perform a set of experiments with various evaluation criteria including relevance metrics, diversity metrics and intent-aware metrics. The experiments show that the proposed γ-RLM algorithm is suitable in practice for real-time bibliographic search.

All of the algorithms in this work are implemented and tested within the**advisor** and the best one (γ-RLM) is currently being used to power the system.

4.3.2 Background

Result Diversification on Graphs

The importance of diversity in ranking has been discussed in various data mining fields, including text retrieval [18], recommender systems [149], online shopping [130], and web search [30]. The topic is often addressed as a multi-objective optimization problem [36], which is shown to be NP-hard [19], and, therefore, some greedy [4, 150] and clustering-based [94] heuristics were proposed. Although there is no single definition of diversity, different objective functions and axioms expected to be satisfied by a diversification system were discussed in [46].

Diversification of the results of random-walk-based methods on graphs only attracted attention recently. GrassHopper is one of the earlier algorithms and addresses diversified ranking on graphs by vertex selection with absorbing random walks [147]. It greedily selects the highest ranked vertex at each step and turns it into a sink for the next steps. Since the algorithm has a high time complexity, it is not scalable to large graphs. DivRank [101], on the other hand, combines the greedy vertex selection process in one unified step with the vertex reinforced random walk model. This algorithm updates the transition matrix at each iteration with respect to the current or cumulative ranks of the nodes to introduce a *rich-gets-richer* mechanism to the ranking. But since the method updates the full transition matrix at each iteration, more iterations are needed for convergence; therefore, the computation cost increases. The shortcomings of those techniques were discussed in [90] in detail. [128] formalizes the problem from an optimization viewpoint, proposes the *goodness* measure to combine relevancy and diversity, and presents a near-optimal algorithm called Dragon. These algorithms are further discussed in Section 4.3.3.

Coverage-based methods (such as [76],[90]) are also interesting for diversification purposes; however, they do not preserve the direction awareness property of the ranking function. Since our aim is to diversify the results of our paper recommendation service, we omitted the results of those coverage-based methods in our experiments.

Problem Definition

Let $G = (V, E)$ be a directed citation graph where $V = \{v_1, \dots, v_n\}$ is the vertex set and E, the edge set, contains an edge (u, v) if paper u cites paper v. Let $\delta^+(u)$ and $\delta^-(u)$ be the number of references of and citations to paper u, respectively. We define the weight of an edge, $w(u, v)$, based on how important the citation is; however, for the sake of simplicity we take $w(u, v) = 1$ for all $(u, v) \in E$. Therefore, the nonsymmetric matrix $\mathbf{W} : V \times V$ becomes a 0-1 matrix. Table 4.6 summarizes the notation used throughout the paper.

We target the problem of paper recommendation assuming that the researcher has already collected a list of papers of interest [67]. Therefore, the objective is to return papers that extend that list: given a set of m seed papers $\mathcal{Q} = \{q_1, \ldots, q_m\}$, s.t. $\mathcal{Q} \subseteq V$, and a parameter k, return top-k papers which are relevant to the ones in $\mathcal{Q}$. With the diversity objective in mind, we want to recommend papers to be not only relevant to the query set $\mathcal{Q}$, but also covering different topics around the query set.

PageRank, Personalized PR, and direction-aware PPR

Let $G' = (V, E')$ be an undirected graph of the citation graph, $p(u, v)$ be the transition probability between two nodes (states), and d be the damping factor.

PageRank (PR) [15]

We can define a random walk on G' arising from following the edges (links) with equal probability and a random restart at an arbitrary vertex with $(1-d)$ teleportation probability. The probability distribution over the states follows the discrete time evolution equation

$$\mathbf{p}_{t+1} = \mathbf{P}\ \mathbf{p}_t, \tag{4.26}$$

where $\mathbf{p}_t$ is the vector of probabilities of being on a certain state at iteration t, and $\mathbf{P}$ is the transition matrix defined as:

$$\mathbf{P}(u, v) = \begin{cases} (1-d)\frac{1}{n} + d\frac{1}{\delta(v)}, & \text{if } (u, v) \in E' \\ (1-d)\frac{1}{n}, & \text{otherwise.} \end{cases} \tag{4.27}$$

If the network is ergodic (i.e., irreducible and non-periodic), (6.1) converges to a stationary distribution $\pi = \mathbf{P}\pi$ after a number of iterations. And the final distribution π gives the PageRank scores of the nodes based on *centrality*.

Table 4.6: Notation for diversity on bibliographic data

	Symbol	**Definition**
Graph	G	directed citation graph, $G\!=\!(V,E)$
	G'	undirected citation graph, $G'\!=\!(V,E')$
	n	$\|V\|$, number of vertices
	$w(u,v)$	weight of the edge from u to v
	$\mathbf{W}$	weight matrix
	$\delta^-, \delta^+(v)$	# incoming or outgoing edges of v
	$\delta(v)$	$\delta^-(v) + \delta^+(v)$, # neighbors of v
	$d(u,v)$	shortest distance between u and v in G'
	$N_\ell(S)$	ℓ-step expansion set of $S \subseteq V$
Query	$\mathcal{Q}$	a set of seed papers $\{q_1, \ldots, q_m\}$, $\mathcal{Q} \subseteq V$
	m	$\|\mathcal{Q}\|$, number of seed papers
	k	required number of results, $k \leq n$
	R	a set of recommended vertices, $R \subseteq V$
	d	damping factor of RWR, $0\!<\!d\!\leq\!1$
	κ	direction-awareness parameter, $0 \leq \kappa \leq 1$
	γ	relaxation parameter of γ-RLM
Random walk	p^*	prior probability distribution
	t	iteration, or timestamp
	$\mathbf{p}_t$	probability vector at iteration t
	η_t	vector of number of visits at iteration t
	$\mathbf{A}$	symm. $n \times n$ transition matrix based on G
	$\mathbf{A}'$	struct.-symm. $n \times n$ trans. matrix based on G'
	$\mathbf{P}$	$n \times n$ transition matrix
	π	$\mathbf{p}_\infty$, stationary probability vector, $\sum \pi(.)\!=\!1$
	ϵ	convergence threshold
Measures	S	a subset of vertices, $S \subseteq V$
	$\hat{S}$	top-k results according to π
	$rel(S)$	normalized relevance of the set
	$diff(S)$	difference ratio of two sets
	$use(S)$	usefulness of the set
	$dens_\ell(S)$	ℓ-step graph density
	$\sigma_\ell(S)$	ℓ-expansion ratio

In practice, the algorithm is said to be converged when the probability of the papers are stable. Let

$$\Delta_t = (\mathbf{p}_t(1) - \mathbf{p}_{t-1}(1), \ldots, \mathbf{p}_t(n) - \mathbf{p}_{t-1}(n)) \tag{4.28}$$

be the difference between probability distributions at iteration t and $t-1$. The process is in the *steady state* when the L2 norm of Δ_t is smaller than the convergence threshold ϵ.

Personalized PageRank (PPR) [54]

In our problem, a set of nodes $\mathcal{Q}$ was given as a query, and we want the random walks to teleport to only those given nodes. Let us define a prior distribution p^* such that:

$$p^*(u) = \begin{cases} 1/m, & \text{if } u \in \mathcal{Q} \\ 0, & \text{otherwise.} \end{cases} \tag{4.29}$$

If we substitute the two $(1/n)$s in (6.2) with p^*, we get a variant of PageRank, which is known as *personalized PageRank* or *topic-sensitive PageRank* [54]. PPR scores can be used as the relevance scores of the items in the graph. The rank of each seed node is reset after the system reaches to a steady state, i.e., $\forall q \in \mathcal{Q}$, $\pi_q \leftarrow 0$, since the objective is to extend $\mathcal{Q}$ with the results.

Direction-aware Random Walk with Restart DaRWR [72]

We defined a *direction awareness* parameter $\kappa \in [0,1]$ to obtain more recent or traditional results in the top-k documents [72]. Given a query with inputs k, a *seed* paper set $\mathcal{Q}$, damping factor d, and direction awareness parameter κ, Direction-aware Random Walk with Restart (DaRWR) computes the steady-state probability vector π. The ranks of papers after iteration t is computed with the following linear equation:

$$\mathbf{p}_{t+1} = p^* + \mathbf{A}\mathbf{p}_t, \tag{4.30}$$

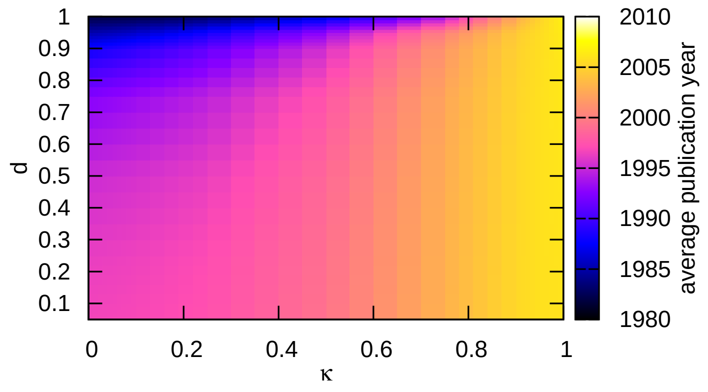


Figure 4.17: Average publication year of top-10 recommendations by DaRWR based on d and κ.

where p^* is an $n \times 1$ restart probability vector calculated with (6.3), and $\mathbf{A}$ is a structurally-symmetric $n \times n$ matrix of edge weights, such that

$$a_{ij} = \begin{cases} \frac{d(1-\kappa)}{\delta^+(i)}, & \text{if } (i,j) \in E \\ \frac{d\kappa}{\delta^-(i)}, & \text{if } (j,i) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{4.31}$$

The transition matrix $\mathbf{P}$ of the RWR-based methods is built using $\mathbf{A}$ and p^*; however, the edge weights in rows can be stored and read more efficiently with $\mathbf{A}$ in practice [70].

Figure 4.17 shows that the direction-awareness parameter κ can be adjusted to reach papers from different years with a range from late 1980's to 2010 for almost all values of d. In our service, the parameter κ can be set to a value of user's preference. It allows the user to obtain recent papers by setting κ close to 1, or older papers by setting κ close to 0.

4.3.3 Diversification methods

We classify the diversification methods for the paper recommendation problem based on whether the algorithm needs to rank the papers only once or multiple times. The first set of algorithms run a ranking function (e.g., PPR, DaRWR, etc.) once and select a number of vertices to find a diverse result set. The algorithms in the second set run the ranking function k times to select each result, and refine the search with some changes at each step. Although the former class of algorithms are preferred for practical use, they may not be able to reach to the intended diversity levels due to the highly greedy nature of the vertex selection process.

Diversification by vertex selection

The following approaches are used after getting the direction-aware relevancy (prestige) rankings of the vertices for a given set of seed nodes. The ranking function is selected as DaRWR with parameters (κ, d).

DivRank: **Vertex-reinforced random walks** [101]

For the random walk based methods mentioned in Section 4.3.2, the probabilities in the transition matrix $\mathbf{P}$ do not change over the iterations. Using vertex-reinforced random walk, DivRank adjusts the transition matrix based on the number of visits to the vertices. The original DivRank assumes that there is always an organic link for all the nodes returning back to the node itself with probability $(1-\alpha)$:

$$p_0(u,v) = \begin{cases} \alpha \frac{w(u,v)}{\delta(i)}, & \text{if } u \neq v \\ 1-\alpha, & \text{otherwise,} \end{cases} \tag{4.32}$$

where $w(u,v)$ is equal to 1 for $(u,v) \in E'$, and 0 otherwise. The transition matrix $\mathbf{P}_t$ at iteration t is computed with

$$\mathbf{P}_t(u,v) = (1-d)\ p^*(v) + d\ \frac{p_0(u,v)\ \eta_t(v)}{\sum_{z \in V} p_0(u,z)\ \eta_t(z)}, \tag{4.33}$$

where $\eta_t(v)$ is the number of visits of vertex v. It ensures that the highly ranked nodes collect more value over the iterations, resulting in the so called *rich-gets-richer* mechanism.

For each iteration of the defined vertex-reinforced random walk, the transition probabilities from a vertex u to its neighbors are adjusted by the number of times they are visited up to that point $\eta_t(v)$. Therefore, u gives a high portion of its rank to its frequently visited neighbors. Since the tracking of η_t is nontrivial, the authors propose to estimate it using two different models. One way is to employ the *cumulative ranks*, i.e., $\mathrm{E}[\eta_t(v)] \propto \sum_{i=0}^{t} \mathbf{p}_i(v)$, and since the ranks will converge after sufficient number of iterations, it can also be estimated with *pointwise ranks* as $\mathrm{E}[\eta_t(v)] \propto \mathbf{p}_t(v)$.

While adapting DivRank to our directional problem, we identified two problems: first, the initial ranks of all nodes should be set to a nonzero value; otherwise, the ranks cannot be distributed with (6.5) for both *pointwise* and *cumulative* estimation of η_t. Therefore, we set $p_0(v) = 1/n$ for all $v \in V$. Second, an organic link returning back to node itself enables the node to preserve its rank. This is problematic since p^* is only set for seed papers, and they tend to get richer over time. However, our objective is to distribute the probabilities over $V \setminus \mathcal{Q}$ to get a meaningful ranking. We solved this problem by removing the organic links of seed papers, hence, distributing all of their ranks towards their neighbors instead of only α of them.

With the listed modifications, we propose the *direction-aware* DivRank algorithm using the transition probabilities

$$p'_0(u,v) = \begin{cases} 0, & \text{if } u \in \mathcal{Q}, u = v \\ \frac{(1-\kappa)}{\delta^+(u)}, & \text{if } u \in \mathcal{Q}, u \neq v, (u,v) \in E \\ \frac{\kappa}{\delta^-(u)}, & \text{if } u \in \mathcal{Q}, u \neq v, (v,u) \in E \\ (1-\alpha), & \text{if } u \notin \mathcal{Q}, u = v \\ \alpha\frac{(1-\kappa)}{\delta^+(u)}, & \text{if } u \notin \mathcal{Q}, u \neq v, (u,v) \in E \\ \alpha\frac{\kappa}{\delta^-(u)}, & \text{if } u \notin \mathcal{Q}, u \neq v, (v,u) \in E \end{cases} \tag{4.34}$$

which can be directly used in (6.5). Depending on the estimation method to be whether *cumulative* or *pointwise*, we refer to the *direction-aware* variants of the algorithm as CDivRank and PDivRank, respectively.

Dragon: **Maximize the goodness measure** [128]

One of many diversity/relevance optimization functions found in the literature is the *goodness* measure. It is defined as:

$$f_{G'}(S) = 2\sum_{i\in S}\pi(i) - d\sum_{i,j\in S}\mathbf{A}'(j,i)\pi(j) - (1-d)\sum_{j\in S}\pi(j)\sum_{i\in S}p^*(i), \tag{4.35}$$

where $\mathbf{A}'$ is the row-normalized adjacency matrix of the graph. The original algorithm runs on the undirected citation graph G' and uses a greedy heuristic to find a near-optimal solution set. Accordingly, the *direction-aware goodness* measure f_G can be defined as:

$$f_G(S) = 2\sum_{i\in S}\pi(i) - d\kappa\sum_{i,j\in S}\mathbf{A}(j,i)\pi(j) - d(1-\kappa)\sum_{i,j\in S}\mathbf{A}(i,j)\pi(i), \tag{4.36}$$

where $\mathbf{A}$ is the row-normalized adjacency matrix based on directed graph, and the last part of (6.15) is always zero $\left(\sum_{i\in S}p^*(i) = 0\right)$ since seed papers are never included in S. The direction-aware variant of the algorithm, running on the directed citation graph and using the ranking vector DaRWR, is referred to as Dragon.

LM: **Choose local maximas**

Because of the smoothing process of random walks, frequently visited nodes tend to increase the ranks of its neighbors [101]. Therefore, we argue that computing local maxima and returning top-k of them will guarantee that the nodes returned this way are recommended by taking the smoothing process of random walks into account.

Once the ranks are computed, the straightforward approach to find the local maxima is to iterate over each node and check if its rank is greater than all of its neighbors' with an $\mathcal{O}(|E|)$ algorithm. However, the algorithm runs much faster in practice since every rank comparison between two unmarked nodes (either local maxima or not) will mark one of them. The LM algorithm is given in Algorithm 12.

Algorithm 12: Diversify with local maxima (LM)

```
Input: $G' = (V, E')$, $\pi$, $k$
Output: An ordered set of recommendations $S$
$L \leftarrow$ empty list of $(v, \pi_v)$
for each $v \in V$ do
    $lm[v] \leftarrow$ LocalMax
for each $v \in V$ do
    if $lm[v] =$LocalMax then
        for each $v' \in adj[v]$ do
            if $\pi_{v'} < \pi_v$ then
                $lm[v'] \leftarrow$ NotLocalMax
            else
                $lm[v] \leftarrow$ NotLocalMax
                break
        if $lm[v] =$LocalMax then
            $L \leftarrow L \cup \{(v, \pi_v)\}$
PartialSort$(L, k)$ w.r.t $\pi_i$ non-increasing
$S \leftarrow L[1..k].v$, i.e., top-$k$ vertices
return $S$
```

γ-RLM: **Choose relaxed local maximas**

The drawback of diversifying with local maxima is that for large k's (i.e., $k > 10$), the results of the algorithm are generally no longer related to the queried seed papers, but some popular ones in unrelated fields, e.g., a set of well-cited physics papers can be returned for a computer science related query. Although this might improve the diversity, it hurts the relevance, hence, the results are no longer useful to the user.

In order to keep the results within a reasonable relevancy to the query and to diversify them, we relax the algorithm by incrementally getting local maxima within the top-γk results until $|S| = k$, and removing the selected vertices from the subgraph for the next local maxima selection. We refer this algorithm as parameterized relaxed local maxima (γ-RLM) where γ is the relaxation parameter. Note that 1-RLM reduces to DaRWR and ∞-RLM reduces to LM. The outline of the algorithm is given in Algorithm 13. In the experiments, we select $\gamma = k$ and refer this algorithm as k-RLM. In Section 4.3.4, we devise other experiments to see the effects of γ with respect to different measures.

Algorithm 13: Diversify w/ relaxed local maxima (γ-RLM)

```
Input: $G' = (V, E')$, $\pi$, $k$, $\gamma$
Output: An ordered set of recommendations $S$
$R \leftarrow$ PartialSort($V, \gamma k$) w.r.t. $\pi_i$ non-increasing
$R \leftarrow R[1 : \gamma k]$
while $|S| < k$ do
    $R' \leftarrow$ FindLocalMaxima($G, R, \pi$)
    if $|R'| > k - |S|$ then
        Sort($R'$) w.r.t. $\pi_i$ non-increasing
        $R' \leftarrow R'[1 : (k - |S|)]$
    $S \leftarrow S \cup R'$
    $R \leftarrow R \setminus R'$
return $S$
```

Diversification by query refinement

In this set of diversification algorithms, the ranking function is called multiple times while some of the parameters or graph structure are altered between those rankings.

GrassHopper: **Incremental ranking using absorbing random walks** [147]

GrassHopper is a well-known diversification algorithm which ranks the graph multiple times by turning at each iteration the highest-ranked vertex into a sink node[32]. Since the probabilities will be collected by the sink vertices when the random walk converges, the method estimates the ranks with the number of visits to each node before convergence.

The original method uses a matrix inversion to find the expected number of visits; however, inverting a sparse matrix makes it dense, which is not practical for the large and sparse citation graph we are using. Therefore, we estimate the number of visits by iteratively computing the cumulative ranks of the nodes with DaRWR.

GSparse: **Incremental ranking by graph sparsification**

In this algorithm, in contrast with GrassHopper, after executing the ranking function, we propose to sparsify the graph by removing all reference and citation edges around the highest ranked node and repeat the process until all k nodes are selected. Note that GrassHopper converts the selected node into a sink node while GSparse disconnects it from the graph (see Alg. 14). This way, the graph around the node becomes less dense, hence, they will attract less visits in a random walk.

[32]A sink node only has a single outgoing edge to itself, so that all its rank stays trapped within the sink.

Algorithm 14: Diversify by graph sparsification (GSPARSE)

```
Input: $G = (V, E)$, $\mathcal{Q}$, $k$
Output: An ordered set of recommendations $S$
$S \leftarrow \emptyset$
$G' \leftarrow G$
for $iter = 1 \rightarrow k$ do
    $ranks \leftarrow$ DARWR($G' = (V', E')$, $\mathcal{Q}$)
    $v \leftarrow \text{argmax}(ranks)$
    $S \leftarrow S \cup \{v\}$
    for each $v' \in adj[v]$ do
        $E' \leftarrow E' \setminus \{(v, v')\}$
    $V' \leftarrow V' \setminus \{v\}$
return $S$
```

4.3.4 Experiments

Evaluation measures

We previously investigated the shortcomings of evaluating result diversification as a bicriteria optimization problem with a relevance measure that ignores diversity, and a diversity measure that ignores relevance to the query in [76]. Since the problem is similarly bicriteria, we argue that the relevance and diversity of the results should be evaluated with separate measures instead of a combined one.

Normalized relevance: The relevancy score of a set can be computed by comparing the original ranking scores of the resulting set with the top-k ranking list [128], defined as

$$rel(S) = \frac{\sum_{v \in S} \pi_v}{\sum_{i=1}^{k} \hat{\pi}_i}, \tag{4.37}$$

where $\hat{\pi}$ is the sorted ranks in non-increasing order.

Difference ratio: The results of a diversity method are expected to be somewhat different than the top-k relevant set of results since, as our experiments will show, the set of nodes recommended by the original DARWR are not diverse enough. This is expected since highly ranked nodes will also increase the ranks of their neighbors [101]. Nevertheless, the original result set has the utmost relevancy. This fact can mislead the evaluation of the experimental results. Therefore, we decided to measure the difference of each result set from the set of original top-k nodes. Given the top-k relevant set $\hat{S}$, the difference ratio is computed with

$$\mathit{diff}(S, \hat{S}) = 1 - \frac{|S \cap \hat{S}|}{|S|}. \tag{4.38}$$

Usefulness: The original ranking scores π actually show the usefulness of the nodes. Since these scores usually follow a power law distribution, the high ranked nodes collect most of the scores and the contribution of two low-ranked nodes to the *rel* measure can be almost the same even though the gap between their positions in the ranking is huge. Yet, the one with the slightly higher score might be useful where the other might not due to this gap. We propose the *usefulness* metric to capture what percentage of the results are actually *useful* regarding their position in the ranking:

$$\mathit{use}(S) = \frac{|\{v \in S : \pi_v \leq \tilde{\pi}\}|}{|S|}, \tag{4.39}$$

where $\tilde{\pi} = \hat{\pi}_{10\times k}$, i.e., the relevancy score of the node with rank $10 \times k$, for $k = |S|$, and $\mathit{use}(S)$ gives the ratio of the recommendations that are within top $10 \times k$ of the relevancy list.

ℓ-step graph density: A variant of graph density measure is the ℓ-step graph density [128], which takes the effect of in-direct neighbors into account. It is computed with

$$\mathit{dens}_\ell(S) = \frac{\sum_{u,v \in S, u \neq v} d_\ell(u, v)}{|S| \times (|S| - 1)}, \tag{4.40}$$

where $d_\ell(u, v) = 1$ when v is reachable from u within ℓ steps, i.e., $d(u, v) \leq \ell$, and 0 otherwise. The inverse of $D_\ell(S)$ is used for the evaluation of diversity in [101].

ℓ-expansion ratio: Other diversity measures, the *expansion ratio* and its variant *ℓ-expansion ratio* [90] measure the coverage of the graph by the solution set. It is computed with

$$\sigma_\ell(S) = \frac{|N_\ell(S)|}{n}, \tag{4.41}$$

where $N_\ell(S) = S \cup \{v \in (V - S) : \exists u \in S, d(u, v) \leq \ell\}$ is the ℓ-step expansion set.

Goodness: Given in (4.36).

Average year: The average publication year of the recommendation set.

Average pairwise distance: Pairwise shortest distance between the results is a measure of how connected or distant the recommendations are to each other. It is computed with

$$APD(S) = \frac{\sum_{u,v \in S, u \neq v} d(u, v)}{|S| \times (|S| - 1)}. \tag{4.42}$$

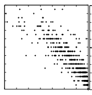
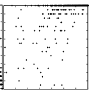
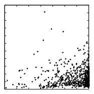
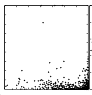
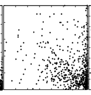
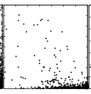
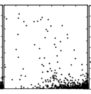
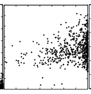
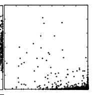
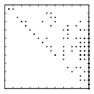
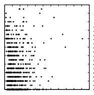
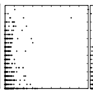
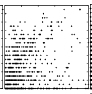
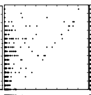
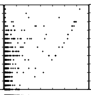
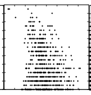
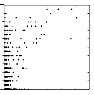
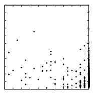
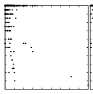
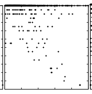
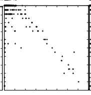
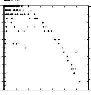
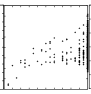
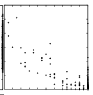
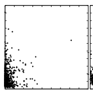
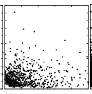
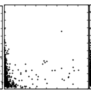
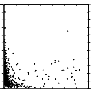
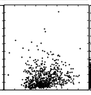
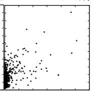
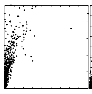
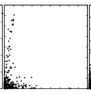
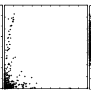
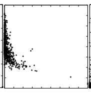
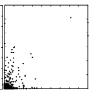
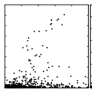
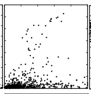
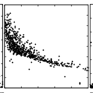
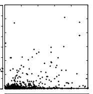
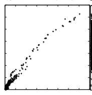
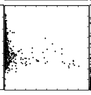
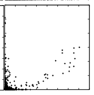
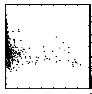
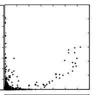
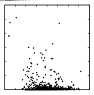
Average MIN distance to $\mathcal{Q}$: Distance of the recommendations to the closest seed paper is a measure of relevance regarding the query:

$$AMD(S) = \frac{\sum_{v \in S} \min_{p \in \mathcal{Q}} d(s, p)}{|S|}. \tag{4.43}$$

Note that the intent-aware measures, such as α-normalized discounted cumulative gain (α-nDCG@k) [30], intent-aware mean average precision (MAP-IA) [4], are not included to the discussions, but they are important measures for evaluating the diversity of the results when the data and queries have some already known categorical labels. Our problem has no assumptions of a known distribution that specifies the probability of an item belonging to a category.

As we list a number of measures, it is important to show that our experiments do not favor any group of measures that correlate with each other. Here, we investigate

Table 4.7: Correlations of various measures. Pearson correlation scores are given on the lower triangle of the matrix. High correlations are highlighted.

	rel	*diff*	*use*	goodness	$dens_1$	$dens_2$	σ_1	σ_2	APD	AMD
rel	–									
diff	**-0.89**	–								
use	0.60	-0.59	–							
goodness	0.06	0.02	-0.21	–						
$dens_1$	-0.13	0.17	-0.48	0.16	–					
$dens_2$	-0.22	0.24	-0.32	0.05	0.58	–				
σ_1	-0.51	0.46	-0.75	0.07	0.14	0.32	–			
σ_2	-0.52	0.47	-0.73	0.04	0.13	0.32	**0.99**	–		
APD	0.27	-0.29	0.27	0.15	-0.47	-0.78	-0.27	-0.30	–	
AMD	-0.37	0.42	-0.78	0.51	0.52	0.24	0.47	0.43	-0.15	–

the listed measures (except average publication year and runtime) by computing their pairwise correlations based on the results of the mentioned algorithms in Section 4.3.3.

Table 6.1 shows the correlations of 10 measures as scatter plots as well as their correlation scores. For the graph diversity measures, ℓ-step expansion ratios (σ_1 and σ_2) are highly correlated among each other, showing that the reachable sets expand independent of the seed nodes (queries), and also proportional to a ratio, which is the average degree of the graph. On the other hand, none of the relevance or diversity measures has a high correlation with other measures.

Dataset collection and queries

We retrieved the metadata information on 2.2M computer science articles (as of May 2013) from DBLP[33], 830K technical reports on physics, mathematics, and computer science from arXiv[34], and 3M medical publications from PMC open access subset[35]. This data is well-formatted and disambiguated; however, it contains very few citation information (less than 470K edges). To increase the number of edges and inter-connect different disciplines, we imported the publications and reference relations from CiteSeer[36], ArnetMiner[37], and Related-Work project[38]. However, most of the data are automatically generated and are often erroneous. We mapped each document to at most one document in each dataset with the title information (using an inverted index on title words and Levenshtein distance) and publication years. Using the disjoint sets, we merged the papers and their corresponding metadata from four datasets. The papers without any references or incoming citations are discarded. The final citation graph has about 11.4M papers and 33.1M directed edges, and will be used in the next version of our service.

The query set is composed of the actual queries submitted to the**advisor** service. We selected about 1840 queries where each query is a set $\mathcal{Q}$ of paper ids obtained from

[33]http://dblp.uni-trier.de/

[34]http://arxiv.org/

[35]http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

[36]http://citeseerx.ist.psu.edu/

[37]http://arnetminer.org/DBLP_Citation

[38]http://blog.related-work.net/data/

the bibliography files submitted by the users of the service who agreed to donating their queries for research purposes. $|\mathcal{Q}|$ varies between 1 and 697, with an average of 33.62.

Results

We run the algorithms on the**advisor** citation graph with varying k values (i.e., $k \in \{5, 10, 20, 50, 100\}$) and with the following parameters: α in (6.4) is selected as 0.25 as suggested in [101]. For the DaRWR ranking, we use the default settings of the service, which are $d = 0.9$ for damping factor, and $\kappa = 0.75$ to get more recommendations from recent publications. In each run, the selected algorithm gives a set of recommendations S, where $S \subseteq V$, $|S| = k$, and $S \cap \mathcal{Q} = \emptyset$. The relevance and diversity measures are computed on S, and the average of each measure is displayed for different k values. The standard deviations are negligible, hence they are omitted.

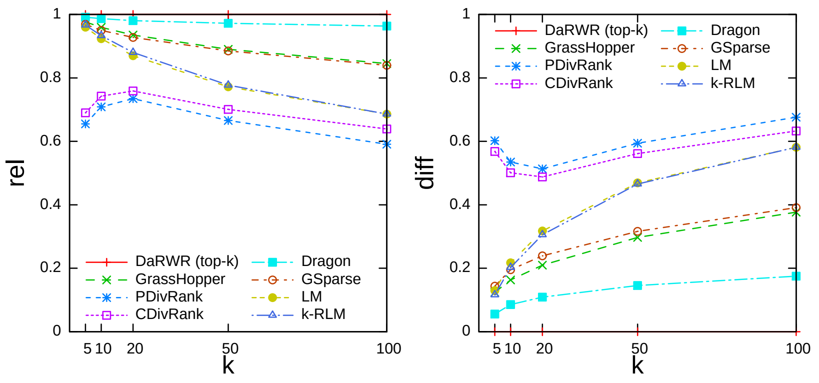


Figure 4.18: Normalized relevance (left) and difference ratio (right) of the result set with respect to top-k results. Note that $rel_{\text{DARWR}} = 1$ and $diff_{\text{DARWR}} = 0$ since we compare the result set against itself. DRAGON returns almost the same result set as top-k.

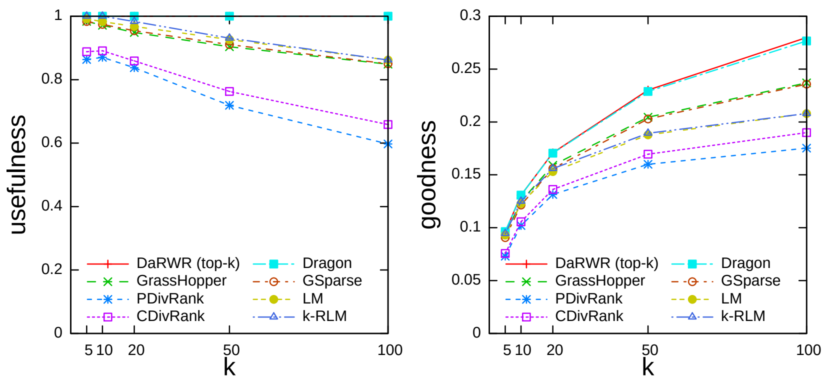


Figure 4.19: Scores based on usefulness (left) and goodness (right) measures. Dragon only slightly improves the goodness measure of the top-k results.

Figure 4.18 shows the normalized relevancy and difference ratio of the recommendations compared to top-k results. It is arguable that a diversity-intended algorithm should maximize the relevancy since top-k results will always get the highest score, yet those have almost no value w.r.t. diversity. However, having a very low relevancy score indicates that the vertices have no connection to the query at all.

Since the normalized relevancy does not give us a clear idea of what is expected from those diversity-intended methods, we compare the set difference of the results from top-k relevant recommendations. Figure 4.18-right shows that Dragon gives a result set that is only 10-15% different than the top-k. In other words, the results of Dragon differ in only one element when $k = 10$. Dragon and the original top-k results score well on direction-aware goodness (Fig. 4.19-right); however, this also means that the goodness measure gives more importance to relevancy and less to diversity.

Graph density is frequently used as a diversity measure in the literature [128, 90]. LM, k-RLM, and DivRank variants seem very promising (see Fig 4.20) for such a

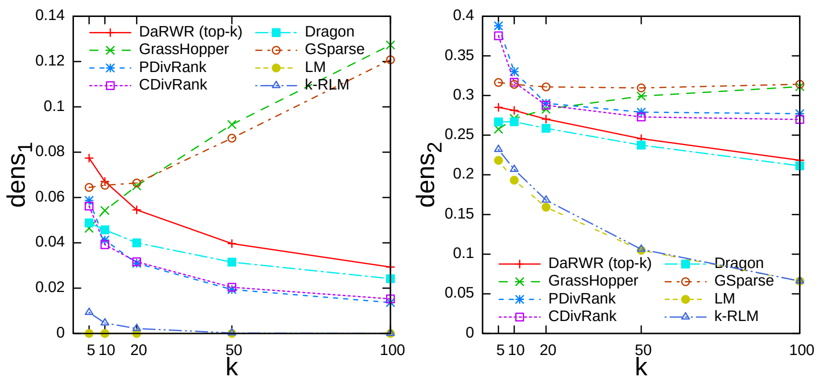


Figure 4.20: ℓ-step graph density ($dens_\ell$) of the results. Note that $dens_1 \simeq 0$ for LM by construction. Both GRASSHOPPER and GSPARSE improve the diversity based on graph density for $k \leq 20$.

diversity objective. The same algorithms also perform good on ℓ-step expansion ratio (see Fig. 4.21), which is related to the coverage of the graph with the recommendations. GRASSHOPPER and GSPARSE perform worse in these diversity metrics. In particular, they are more dense than the results of DARWR.

After evaluating the results on various relevancy and diversity metrics, we are left with only a couple of methods that performed well on almost all of the measures: LM, k-RLM, and DIVRANK variants. However, Figure 4.22 shows that PDIVRANK and CDIVRANK give a set of results that are more connected (i.e., have a low average pairwise distance) and do not recommend recent publications (see Fig. 4.22-right) although κ is set accordingly. Since we are searching for an effective diversification method that runs on top of DARWR, DIVRANK variants are no longer good candidates.

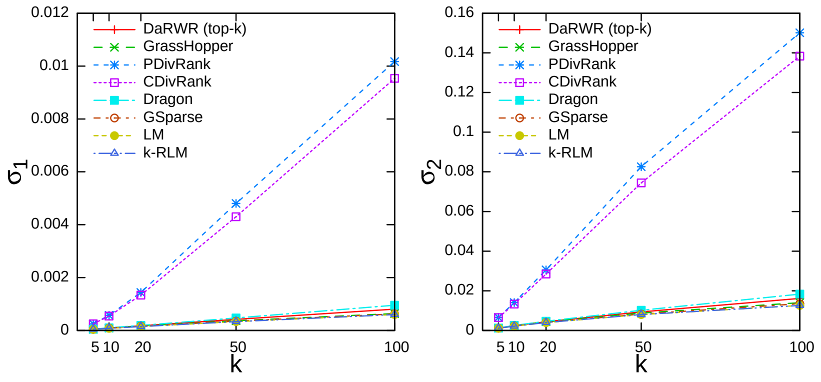


Figure 4.21: ℓ-step expansion ratio (σ_ℓ) of the results. DivRank variants improve the diversity based on σ_ℓ.

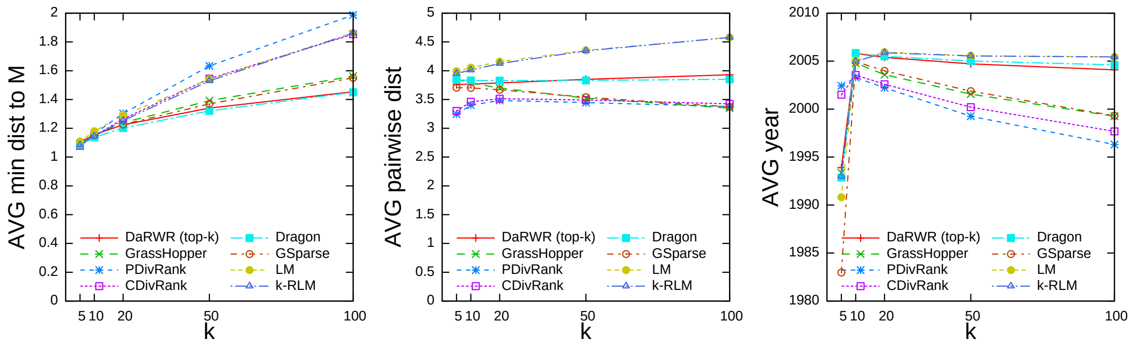


Figure 4.22: Results on average minimum distance to the query, average pairwise shortest distance between recommended papers, and average publication year.

Scalability

The running time of the algorithms is also crucial for the web service since all the recommendations are computed in real-time. The experiments were run on the same architecture that the service is currently using. It has a 2.4GHz *AMD Opteron* CPU and 32GB of main memory. The CPU has 64KB L1 and 1MB L2 caches. The DaRWR method and the dataset are optimized based on the techniques given in [70]. In order to get a consistent runtime, the experiments are repeated ten times and averaged over these executions. Although the target architecture has 8 cores, the entire node was allocated for the experiment, but only one core was used.

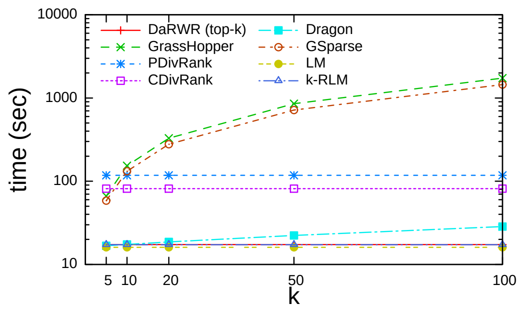


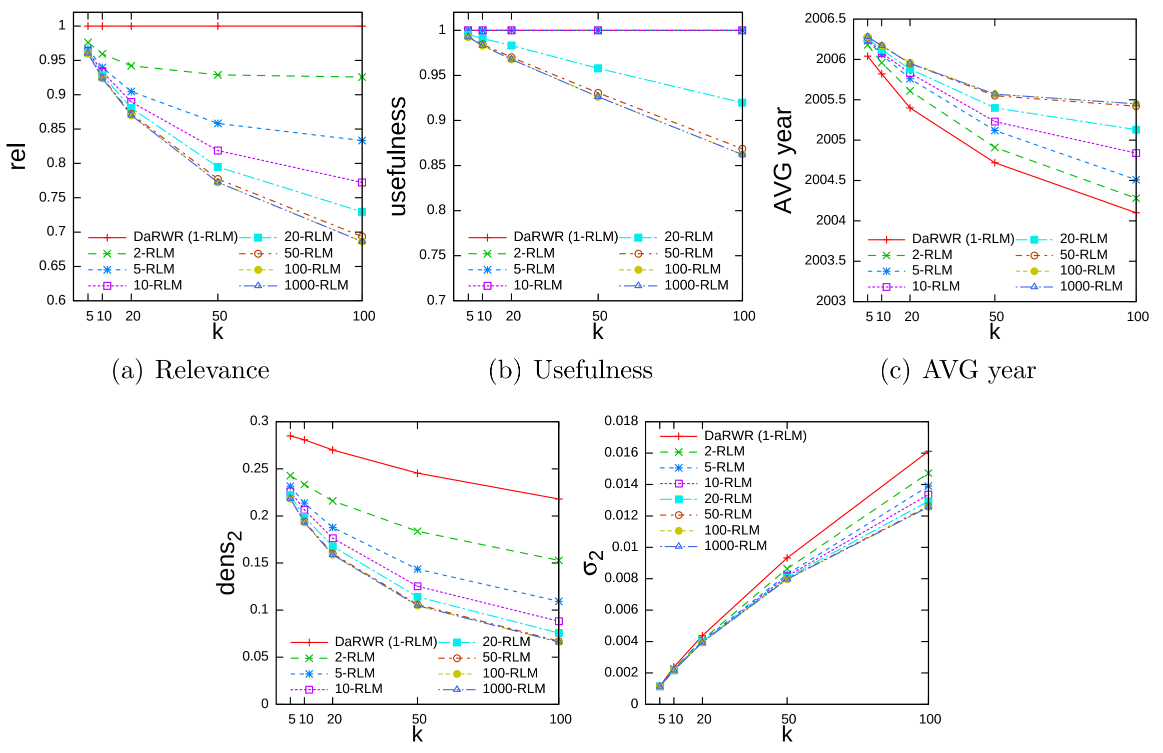
Figure 4.23: Running times of the algorithms for varying k. DaRWR, LM, k-RLM are equal.

It was expected that the complexity of the methods based on query refinement depend on and increase linearly with k. Figure 4.23 shows that GrassHopper, and GSparse have the longest runtimes, even though they were faster than DivRank variants for $k \leq 10$. This behavior was previously mentioned in [101]. The running time of Dragon is slightly higher than LM and k-RLM since it updates the goodness vector after finding each result.

In short, the query refinement-based methods (GrassHopper, GSparse) have linearly increasing runtimes. DivRank variants require more iterations, therefore, more time to converge. Finally, Dragon, and especially LM and k-RLM are extremely efficient compared to other methods.

Parameter test

Our experiments on different relevance and diversity measures show that:



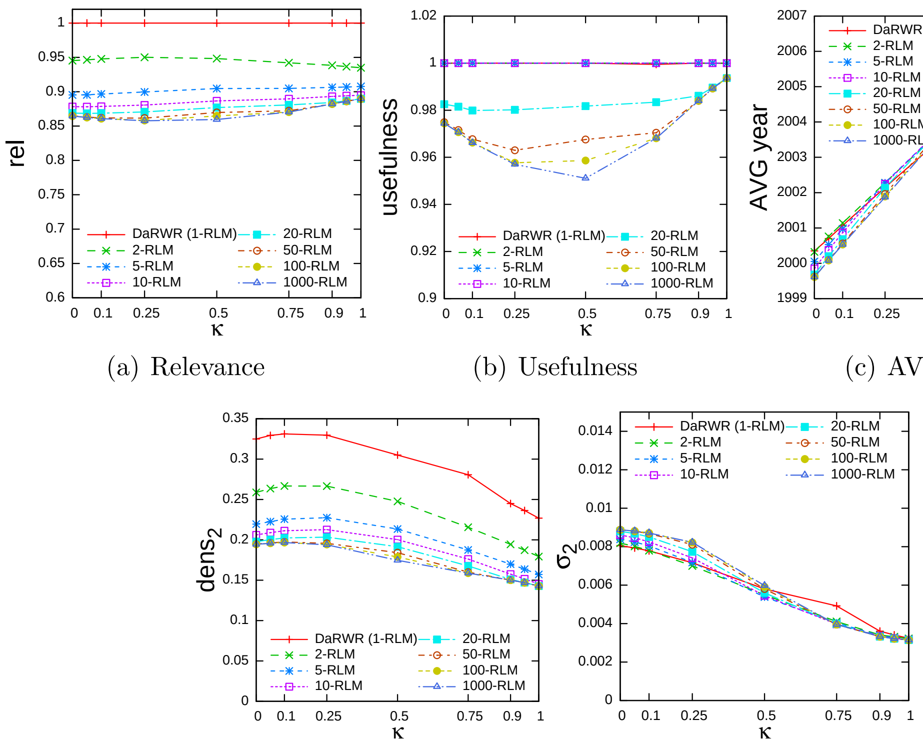
(a) Relevance (b) Usefulness (c) AVG year

(d) Density (e) Expansion ratio

Figure 4.24: Parameter test on γ-RLM with varying γ and k parameters for $\kappa = 0.75$. As the method outputs more results with increasing k, the result set's relevance deteriorates and its diversity improves with increasing γ.

- Dragon returns almost the same result set as top-k, while the graph density and expansion ratio measures also imply low diversity for their results,
- GrassHopper and GSparse perform worse based on the diversity measures,
- DivRank variants sacrifice direction-awareness for the sake of diversity,

whereas LM and k-RLM perform relatively good in almost all experiments, with a negligible computation cost on top of DaRWR. k-RLM is slightly better than LM since it also improves the relevancy of the set to the query.



(a) Relevance (b) Usefulness (c) AVG year

(d) Density (e) Expansion ratio

Figure 4.25: Parameter test on γ-RLM with varying γ and κ parameters for $k\!=\!20$. γ-RLM significantly improves the diversity of the results. Average publication year of the results adapt better with the given κ for $\gamma\geq 5$.

In order to understand the effects of the γ parameter to the quality of the result set, we display the results of γ-RLM with varying γ and k parameters in Figure 4.24. The experiments suggest that γ-RLM is able to sweep through the search space between all relevant (results of DaRWR) and all diverse (results of LM) with a varying γ parameter. Therefore, this parameter can be set depending on the data and/or diversity requirements of the application.

Figure 4.25 shows the results of γ-RLM with varying γ and κ parameters for $k\!=$ 20. γ-RLM significantly improves the diversity of top-k results for any κ parameter.

For $\gamma \geq 5$, average publication year of the results adapts better with the given κ, returning more recent papers as κ is closer to 1, and more traditional papers otherwise.

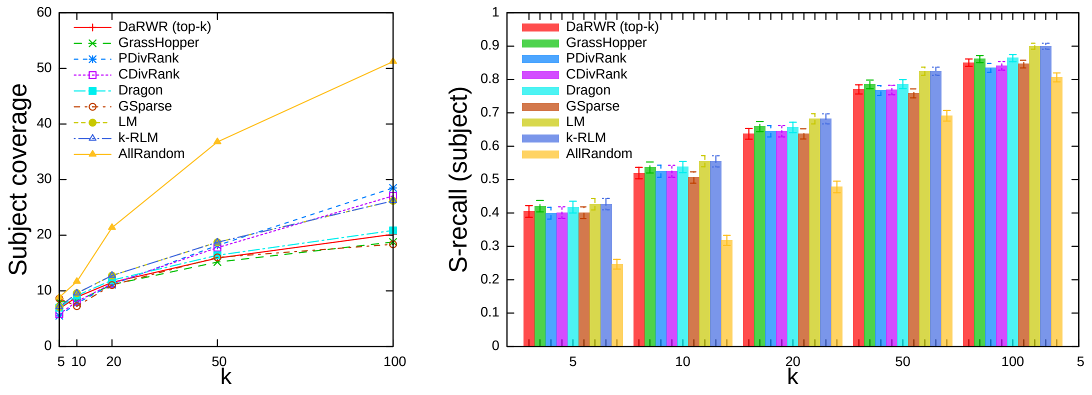
Intent-aware experiments

Here we present an evaluation of the intent-oblivious diversification algorithms against intent-aware measures. This evaluation provides a validation of the techniques with an external measure, such as *group coverage* [90] and *S-recall* [144].

From the citation graph we obtain from different sources, we extract a subgraph of 545K vertices and 3.1M edges which corresponds to the citation graph of arXiv articles. We use this subgraph in intent-aware experiments because the authors of those articles assign at least one subject (e.g., "High Energy Physics - Phenomenology", "Mathematics - Combinatorics", "Computer Science - Computational Geometry", etc.) out of 142 categories. On average 1.52 subjects were assigned to each paper in the dataset.

The queries are selected with respect to the scenarios explained in [76]. Since our aim is to evaluate the results based on the coverage of different groups, we randomly generate 1000 query sets that represent multiple interests. Specifically, for each query set, up to 10 random papers are selected from the citation graph as different interests of the user, and a total of 10 to 100 vertices within distance-2 of those interests are added to the query set. The intent of each query set $\mathcal{Q}$ is extracted by collecting the subjects of each seed node.

One measure we are interested in is the *group coverage* as a diversity measure [90]. It computes the number of groups covered by the result set and defined on subjects based on the intended level of granularity. However, this measure omits the actual



(a) (b)

Figure 4.26: Average intent-coverage and S-recall scores for the results of different diversification algorithms based on subjects. 95% confidence intervals for S-recall are also provided.

intent of a query, assuming that the intent is given with the subjects of the seed nodes.

Subtopic recall (*S-recall*) has been defined as the percentage of relevant subjects covered by the result set [144]. It has also been redefined as *Intent-Coverage* [146], and used in the experiments of [135]. *S-recall* of a result set S based on the set of intents of the query I is computed with

$$S\text{-}recall(S, I) = \frac{1}{|I|} \sum_{i \in I} B_i(S), \tag{4.44}$$

where $B_i(S)$ is a binary variable indicating whether intent i is found in the results.

We give the results of group coverage and *S-recall* on subjects in Figure 6.8. The results of AllRandom are included to give a comparison between the results of top-k relevant set (DaRWR) and ones chosen randomly.

As the group coverage plots show, top-k ranked items of DaRWR do not have the necessary diversity in the result set, hence, the number of groups that are covered by these items are the lowest of all. On the other hand, a randomized method brings irrelevant items from the search space without considering their relevance to the user query. The results of all of the diversification algorithms reside between those two extremes, where DivRank and LM variants cover the most, and GSparse and GrassHopper cover the least number of groups.

However, *S-recall* index measures whether a covered group was actually useful or not. Obviously, AllRandom scores the lowest as it dismisses the actual query (you may omit the *S-recall* on topics since there are only 6 groups in this granularity level). Among the algorithms, LM and k-RLM score the best overall while GrassHopper have similar *S-recall* scores for $k = 10$ and 20, even though LM and k-RLM are much faster algorithms than GrassHopper (cf. Figure 4.23).

Empirical results

Here, we try to exemplify the effects of diversifying recommendations with k-RLM method on a real world query[39]. The recommended and top-100 ranked papers are manually clustered and labeled into categories, i.e., graph mining (GM), generic SpMV (Sp), compression (C), multicore (MC), partitioning (P), GPU (GPU), and eigensolvers (E).

The query is the bibliography of a submitted paper related to SpMV optimization for emerging architectures, hence a multidisciplinary paper. The query includes a couple of graph mining papers, and five out of ten relevance-only recommendations are related to graph mining, where three of them are neighbors. Figure 4.27 shows

[39]Available at http://theadvisor.osu.edu/csfeedback.php?q=e302d9fea1f22310cbf64c39a0a20d4e.ris,0.75

	top-k results		**k-RLM diversified**	
#	paper	label	paper	label
1	Govan09	GM	Govan09	GM
2	Kourtis08	C	Kourtis08	C
3	Lao10	GM	Lao10	GM
4	**Abbey10**	GM	Bradley10	GM
5	Bradley10	GM	Hoemmen10	Sp
6	Hoemmen10	Sp	**Saak64**	GPU
7	**Knight06**	GM	**Guo10**	GPU
8	**Davis97**	P	**Lee10**	MC
9	**Toledo97**	Sp	**Im04**	GPU
10	**Im00**	Sp	**Kaiser10**	MC

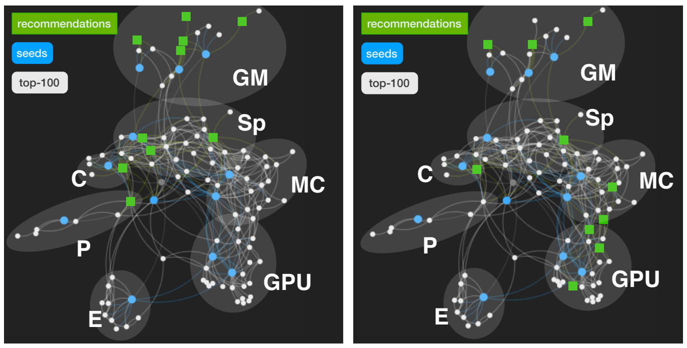


Figure 4.27: top-10 and k-RLM diversified results for the given query (a), original and diversified recommendations are visualized with their categories (b,c). Diversified results bring about the same number of papers from categories that seed papers belong to.

that the recommendations with k-RLM diversification improve the set of recommendations by eliminating redundant results and by covering other fields of interest. Indeed, no results from the multicore and GPU categories were returned before. After diversification, these two topics are covered. Moreover, the distribution of categories of k-RLM results resembles the one of the query, while top-k results do not.

4.3.5 Summary

In this work, we addressed the diversification of paper recommendations of the**advisor** service, which ranks papers in the literature with a direction-aware personalized PageRank algorithm. While giving a survey of diversity methods designed specifically for random-walk-based rankings, we adapted those methods to our direction-aware problem, and proposed some new ones based on vertex selection and query refinement. Our experiments with various relevance and diversity measures show that the proposed γ-RLM algorithm can be preferred for both its efficiency and effectiveness.

Chapter 5: Exploratory Search and Result Diversification

Provided the tremendous amount of documents the academic community has published, we argue that a useful academic recommendation system must have three particular properties. First, the system should allow the researcher to execute a complex **personalized** search. To obtain the highest accuracy, the query should be processed at a conceptual level. Yet, even the best algorithms may not be able to pin the important documents precisely, only the user will recognize them. Therefore, the tool should also allow the user to **explore** its database in multiple ways to enable finding and discovering the interesting documents. Furthermore, the system needs to be **efficient** to keep the response time short enough to encourage the user for more complex queries, and since the amount of data will increase, the system must also be **scalable** to stay efficient in the future.

We previously evaluated the direction awareness feature of the framework 4.1, provided the details of various efficient implementations 4.2, and presented preliminary results on result diversification 4.3. The contributions of this chapter can be summarized as follows:

- we show how relevance feedback and result diversification affect and improve the service in practice.

- we show how the different techniques complement each other to provide a powerful document discovery engine.

5.1 Motivation and Contributions

Many academic services have been developed in the past. DBLP is a publicly available web service that references more than two million papers in computer science with complete bibliographic information, venue of publication, and author disambiguation (proper handling of homonyms) [86]. CiteSeer$^\chi$ is another web service which harvests the web for scientific publications in computer and information sciences [87]. The service analyses the documents and automatically extracts the text, title, authors, and reference list of them. A manual editing for this data (crowdsourcing) is possible to fix the mistakes produced by the automatic system. These data are used to feed multiple services; citation analysis within CiteSeer$^\chi$, expert finding in SeerSeer, collaborator suggestions in CollabSeer [26], acknowledgment search in AckSeer [64], and automatic paper recommendation based on text similarity in RefSeer [55]. Also MyCiteSeer$^\chi$ can provide documents that are within the area of interest of a given researcher.

Publishers such as ACM, IEEE, Elsevier, and JSTOR have digital libraries that contain all the articles they published. The documents are properly annotated and typically contain clean reference information to other documents. These libraries often have basic article suggestion features. However, the licensing of the data they employ do not typically allow the third parties to use it for any purpose other than merely reading them.

CiteULike is a social tagging application where researchers can manage their bibliography, tag them with relevant keywords, and share it publicly with other researchers. Using this information, CiteULike can provide a recommendation based on the common interests between researchers. A study of its usefulness in bioinformatics can be found in [47]. Services similar to CiteULike help researchers select and group other papers, letting its users specify different area of interests, but the recommendation part is commonly very limited.

Google Scholar is a popular and generic academic text-based search engine which exposes citations and references. It is paired with a social network which allows a user to track citations to his/her articles. In addition, based on the publication list of a researcher, the service also provides personalized suggestions as new documents are indexed. Microsoft Academic Search is another text-based search engine which also features filtering by areas, topics, co-authorship information, and temporal trends. Similarly, ArnetMiner [118] is a tool to explore the academic social network. It allows to search with respect to trends, rankings, and topics of researchers and conferences. Google Scholar provides personalized suggestions; however, only based on the current publications of its users. This strategy is also beneficial to stay up-to-date on published topics, but when a researcher starts working in a new discipline, such a system cannot provide relevant information.

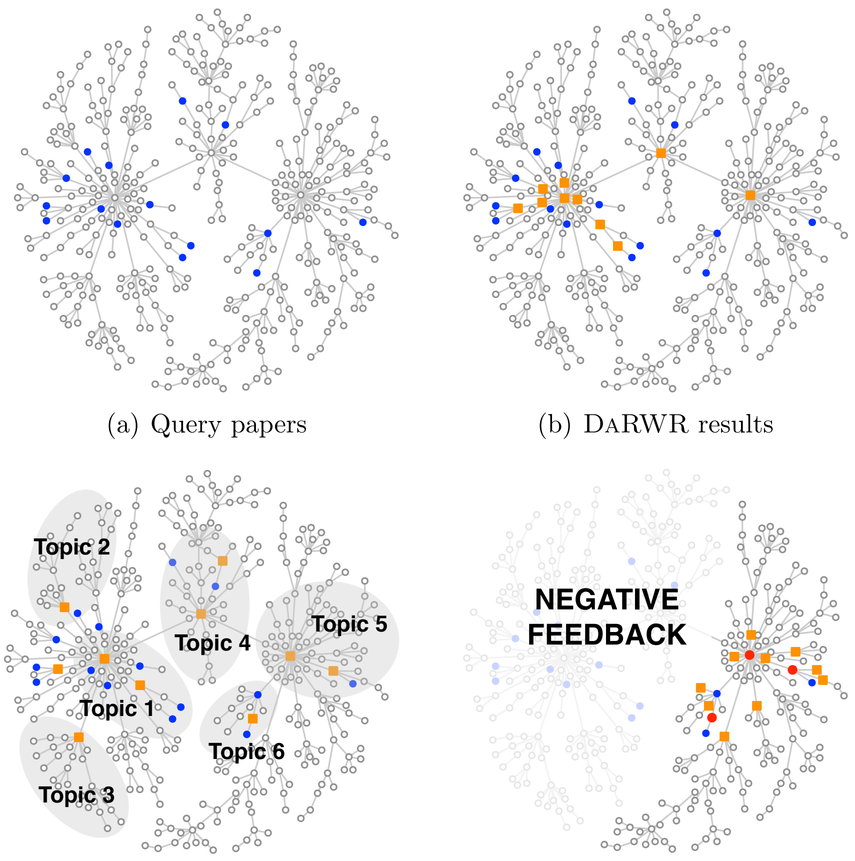
All these existing services allow to perform simple queries on the data they host such as "which documents are popular in a given topic or relevant to a keyword?", "which researchers are close to a topic or another researcher?", or "are there any documents that match a given set of keywords?". While writing an article, these questions are good starting points, which we refer to as *first-level bibliographic search*. A recommender systems employing features like personalization, exploratory search, etc., could provide a better literature search. Existing web services generally address the scalability concerns; however, none of these services are personalized and exploratory to a satisfactory extent.

In this work, we present the components of the web service called the**advisor** which is developed to improve the literature search process with a personalized and exploratory search. The service enables exploration of the scientific literature at three different levels. Result diversification answers the question "what is out there?", while relevance feedback allows to guide the search process in a more localized region. Finally visualization provides a direct interaction with the documents, enabling an exhaustive exploration in a structured fashion.

5.2 Exploratory Search

In addition to providing personalized and accurate recommendations, giving various options to researchers to find other related papers, in other words, enabling the users to do *exploratory search* on the relevant parts of the database, has its advantages.

In our service, we provide the exploratory search functionality with (a) **result diversification** so that researchers may find papers from different aspects of the query, (b) **relevance feedback** so that researchers can focus on only the areas that they are interested in, and (c) **visualization** so that relevant papers that did not appear in the top-10 results are visible with their relationships to the seed papers as well as other recommendations. We give the details of each component and their



(a) Query papers (b) DaRWR results

(c) Diversified results (d) DaRWR results after feedback

Figure 5.1: Exploratory properties on a sample graph. Seed papers in $\mathcal{Q}$ are highlighted in (a), top-k recommendations of DaRWR are dominated by the papers in the left part of the graph (b), diversified results with k-RLM introduce papers from different aspects of the query (c). When positive feedback is given to the papers in Topics 5 and 6, and negative feedback is given to the rest of the results, the recommendations are more focused on the right part of the graph (d).

experimental results below; however, since the exploratory search is more of a personalized and subjective matter, it is typically difficult to quantify its effects. We give an overview of how result diversification, relevance feedback, and visualization are useful in Figure 5.1.

5.2.1 Result diversification

Methods such as DaRWR tend to naturally return many recommendations from the same area of the graph, which leads to a poor coverage of the potential interests of the users. Diversifying the recommendations refers to the methods that increase the amount of distinct information one can reach via an automatized search. Diversification for the random-walk-based methods attracted attention recently: GrassHopper addresses diversified ranking on graphs by vertex selection with absorbing random walks [147]. DivRank uses a greedy vertex selection process and updates the transition matrix at each iteration with respect to the current node ranks by introducing a *rich-gets-richer* mechanism to the ranking [101]. Dragon approaches the problem from an optimization point, proposes the *goodness* measure to combine relevancy and diversity, and presents a near-optimal algorithm [128].

We can use any of these algorithms in the**advisor** since they can easily be enhanced with the direction-awareness property; however, the main criteria we need to consider here is the efficiency, and hence, whether or not the diversification algorithm can be used in a real-time service. We previously showed that GrassHopper has a high time complexity and it is not scalable to large graphs [73]. DivRank updates the full transition matrix at each iteration, hence more iterations are needed for convergence; therefore, the computation cost increases. Dragon, on the other hand, could not provide a diverse enough set of results.

We argue that finding the vertices which are locally maximum in the graph w.r.t. their ranks and returning the k most relevant ones will diversify the results and increase the coverage of citation graph. Once the ranks are computed, the straightforward approach for identifying the local maximas is to iterate over each node in the graph and check if its rank is greater than all of its neighbors' with a $\mathcal{O}(|E|)$ algorithm. The drawback of diversifying with local maximas is that for large k's (i.e., $k > 10$), the results of the recommendation algorithm are generally no longer related to the queried seed papers. Popular papers in unrelated fields can be returned, e.g., a set of well-cited physics papers for a computer science related query. Although this might improve the diversity, it hurts the relevancy, hence, the results will no longer be useful to the user.

In order to keep the results within a reasonable relevancy threshold and to diversify them at the same time, we relax the algorithm by incrementally getting local maximas only within the top-γk results and removing the selected vertices from the subgraph for the next local maxima selection until $|S| = k$. We refer to this algorithm as parameterized relaxed local maxima (γ-RLM). Note that 1-RLM reduces to DaRWR. We described the algorithm first in [78].

5.2.2 Relevance feedback

Users of the**advisor** are given the option of providing explicit relevance feedback to the set of recommended papers. The feedbacks can be either positive or negative, making a recommendation relevant or irrelevant for the query. When the user refines the query with the relevance feedback, the relevant results are added to $\mathcal{Q}$, and the irrelevant results are removed from the citation graph with all of their incident edges.

Relevance feedback can also be incorporated well with the diversification feature explained in the previous section. As the diversified set of results may represent different sets of papers from different areas of interest, the user of the service can easily guide the recommendation process towards the fields that she is interested in.

5.2.3 Data visualization

Representation of the recommendations in a web service is crucial for user experience. Aside from displaying the full bibliographic entries for the list of suggested papers, we also visualize the results and their relationships to the references and papers that are given positive feedback before. The sample graph (see Fig. 5.2) consist of the seed papers $\mathcal{Q}$ (*blue*), recommendations (*green*), and top-100 relevant papers (*white*). The subgraph is extracted from those vertices and all the edges within this subset. We then apply a force-directed layout algorithm [127] to improve the representation as well as expose the paper clusters.

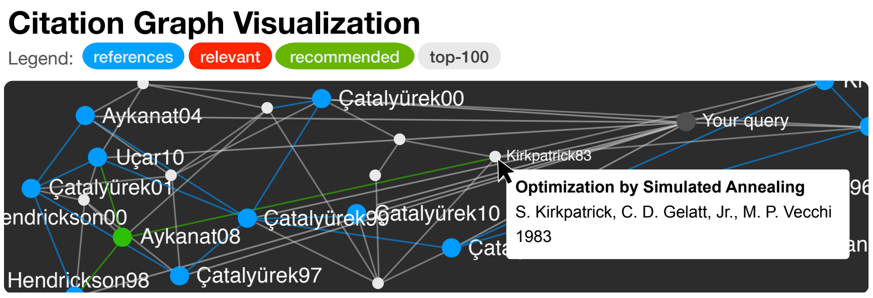
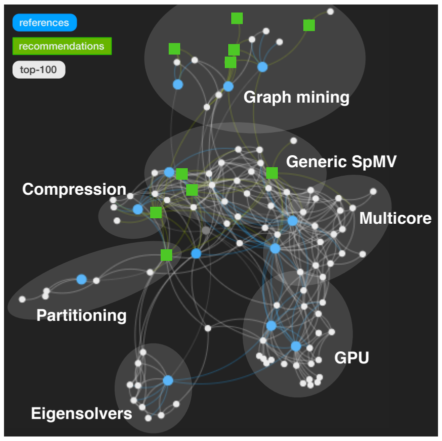


Figure 5.2: Visualization of a sample query.

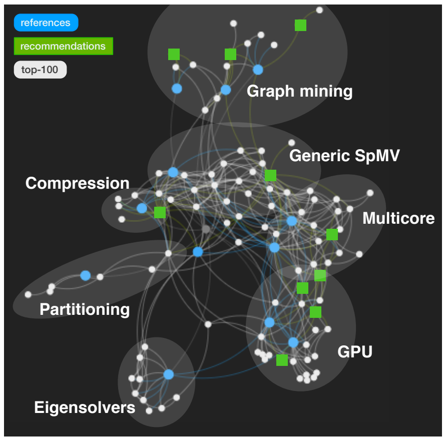
5.3 Experiments

5.3.1 Experiments on diversity

We have given the quantitative analysis of γ-RLM and compared against the listed diversification methods with various relevance and diversity measures in [73]. Here, we try to exemplify the effects of reranking the recommendations with a diversification



(a) DaRWR results



(b) DaRWR results with diversification

	top-k results		**k-RLM diversified**	
#	paper	label	paper	label
1	Govan09	GM	Govan09	GM
2	Kourtis08	C	Kourtis08	C
3	Lao10	GM	Lao10	GM
4	**Abbey10**	GM	Bradley10	GM
5	Bradley10	GM	Hoemmen10	Sp
6	Hoemmen10	Sp	**Saak64**	GPU
7	**Knight06**	GM	**Guo10**	GPU
8	**Davis97**	P	**Lee10**	MC
9	**Toledo97**	Sp	**Im04**	GPU
10	**Im00**	Sp	**Kaiser10**	MC

(c) Recommendations in each topic

topic	**query**	**top-k**	**k-RLM**
Multicore	2	0	2
GPU	2	0	3
Eigensolver	1	0	0
Graph Mining	3	5	3
Compression	1	1	1
Generic SpMV	1	3	1
Partitioning	1	1	0

(d) Number of papers in each topic returned by the algorithms

Figure 5.3: Comparison of the recommendations of DaRWR and diversified DaRWR for a given query related to SpMV optimization for emerging architectures. The graph is manually annotated with topical information. Without diversification, five out of ten recommendations are related to graph mining, where three of them are neighbors. Diversification allows to cover more topics by eliminating redundant results and including items from uncovered topics.

method on a real world query[40]. The recommendations are diversified and visualized within the**advisor**, and we manually clustered and labeled the citation subgraph of $\mathcal{Q}$ and the top-100 ranked papers in Figure 5.3.

[40]The query is available at http://theadvisor.osu.edu/csfeedback.php?q=e302d9fea1f22310cbf64c39a0a20d4e.ris,0.75

The query is the bibliography of a submitted paper related to SpMV optimization for emerging architectures, hence a multidisciplinary paper. The query includes a couple of graph mining papers, and five out of ten relevance-only recommendations are related to graph mining (Fig. 5.3(a)), where three of them are neighbors. The recommendations with k-RLM diversification (Fig. 5.3(b)) improve the set of recommendations by eliminating redundant results and by covering other fields of interest. Indeed, no results from the Multicore and GPU categories were returned before. After diversification, these two topics are now covered. Notice that a recommendation from the Partitioning category was removed by the diversification algorithm because one of its neighbors from the Compression with a higher rank have already been included.

In case the user is interested in finding more GPU related papers, she may investigate other papers within the top-100 relevant set using the visualization, or may give a positive feedback on the current recommendations on GPU and refine the search without diversification. We believe that diversification is an essential part of the paper recommendation process, especially when the query set is composed of multidisciplinary papers and/or papers from separate disciplines.

5.3.2 Experiments on relevance feedback

Relevance feedback is an important part of the recommendation system since users may give positive and negative feedbacks on the results in order to reach to desired papers or topics. In this test, 2500 source papers are randomly selected. For each source paper s, the graph is pruned by removing the papers published after s. Then, a target paper t is selected from the pruned graph, such that it is the most relevant paper at distance 3 from the source (i.e., $t = \text{argmax}_{t \in V} \mathbf{p}_t$, s.t. $d(s,t) = 3$

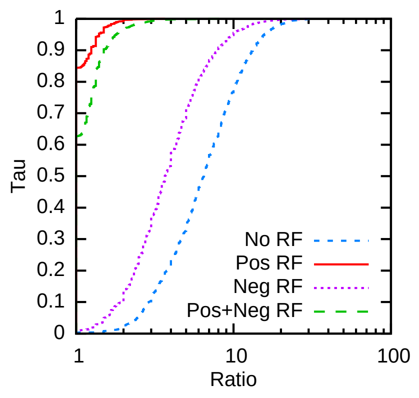
and year[t]≤year[s]). Assuming that a user can only display 10 results at a time, we measure the number of pages that the user has to go through until she reaches t. We compare the feedback mechanism with the following user behaviors:

No feedback: There is no feedback mechanism; therefore, user should keep going to the next page until she finds the target paper.

Only positive feedback (+RF): Relevant results are added to $\mathcal{Q}$ in the next step, irrelevant results should not be displayed again.

Only negative feedback (-RF): Irrelevant results are removed from the graph, relevant results are kept but will not be displayed again.

Both positive and negative (±RF): Results are labeled as either relevant to be added to $\mathcal{Q}$ or irrelevant to be removed from the graph.



Q	**−RF**	**+RF**	**±RF**
1	88.23	**17.64**	**17.64**
2	66.66	**33.33**	44.44
3	36.84	**10.52**	13.15
4	96.66	36.66	**33.33**
5	78.94	**31.57**	42.10
6	55	35	**30**
7	30.43	34.78	**26**
8	100	**15**	**15**
9	70	**30**	**30**
10	**22.22**	103.7	40.74
...			
AVG	**66.99**	**19.61**	**22.85**

Figure 5.4: Relevance feedback experiments: performance profiles (left) and sample of the number of pages one has to go through expressed as a percentage of the number of pages without using any feedback (right), e.g., using only positive feedback allows to reduce the number of pages by 80.39% on average.

Figure 5.4-(left) presents the performance profile of the experiments. A feedback policy passing through point $(ratio, \tau)$ achieves a result at worse *ratio* times the number of pages of the best policy in a fraction τ of the case. Figure 5.4-(right) presents the percentage of pages needed to find a target paper at distance-3 with positive and/or negative relevance feedback compared to not using feedback at all. Using negative feedback only reduces the number of pages one has to go through by 30.01% in average and using positive feedback allows to reduce the number of pages by 80.39% on average. Using both negative and positive feedback reduces the number of pages by 77.15% on average. The results show that the feedback mechanism, especially positive feedback, allows to speedup the process of searching for specific references.

5.4 Summary

In this work, we identify the properties that an academic recommendation service should provide to its users as personalized search, exploration of the data, and efficient and scalable methods. We argue that the existing academic services lack some of the mentioned properties. Personalization is achieved by the user indicating a set of relevant documents. Exploration is achieved by three techniques leading to an overall description of the area (diversification), guiding technique to reinforce interest (relevance feedback) and manual discovery (visualization). All the features of the system are based on sound algorithmic decisions and are shown to be very beneficial in practice.

Chapter 6: Graph Diversity and Common Pitfalls in its Evaluation

Algorithms developed for graph-based recommendation are very popular among web services; for instance, Amazon uses co-purchasing information to recommend products to its customers, IMDB recommends movies to its visitors based on the information such as director, cast, and ratings, and Google uses the web-graph and the user histories for personalized web search. The recommendations are usually made based on user preferences, either explicitly expressed or based on what she has been looking at recently. These preferences are used as the objects of known interest to seed the algorithms.

One of the common problems of popular recommendation algorithms is the pollution of top recommendations with many similar items, i.e., *redundancy*. It is typically not interesting to be recommended slight variations of the same product if you have a wide interest. The redundancy problem is solved via result *diversification*, which has gained a lot of attention in many fields recently [4, 37, 69, 90, 101, 111, 128, 131]. Diversification usually refers to the process of returning a set of items which are related to the query, but also dissimilar among each other. The problem of recommending a diversified set is inherently qualitative and is evaluated differently in various contexts [18, 24, 30, 144].

Most diversification studies in the literature rely on various assumptions, e.g., objects and queries are categorized beforehand [135], or there is a known distribution that specifies the probability of a given query belonging to some categories [4]. In the context of information retrieval or web search, since the search queries are often *ambiguous* or *multifaceted*, a query should represent the *intent* of an average user with a probability distribution [135]. Intent-aware methods in the literature aim to cover various relevant categories with one or more objects, or as TREC defines its diversity task, *"documents are judged on the basis of the subtopics, based on whether or not a document satisfies the information need associated with that subtopic"* [29].

In this work, we assume that the graph itself is the only information we have, and no categories or intents are available. We are interested in providing recommendations based on a set of objects of interest. The recommended items should be related to the user's interests while being dissimilar to each other. This particular problem has attracted a lot of attention recently, and many algorithms and evaluations have been proposed [27, 37, 74, 90, 101, 128, 147, 146].

Evaluation of algorithms' quality is one interest of the paper. Usually, algorithms are evaluated by expressing the problem as a bicriteria optimization problem. The first criteria is related to relevancy, e.g., the sum of the personalized PageRank scores, and the second is related to diversity, e.g., the density or the expansion ratio of the subgraph formed by the recommended set. These two criteria are either aggregated (often with a simple linear aggregation) or they are considered simultaneously with Pareto dominance (where the solutions are in the relevancy-diversity objective space). As the first contribution, we show that such an evaluation is inappropriate. Indeed, we design *query-oblivious* algorithms for the two popular combinations of objectives that return most of the recommendations without considering the user's interests, yet, perform the best on these commonly used measures.

We argue that a result diversification algorithm should be evaluated under a measure which tightly integrates the query in its value. The *goodness* measure proposed in [128] has such a property; however, it is shown to be dominated by the relevance. We propose a new measure called *expanded relevance* ($exprel_\ell$) which computes the coverage of the relevant part of the graph. We show that the query-oblivious algorithms cannot optimize $exprel_\ell$.

We also investigate various quality indices by computing their pairwise correlations. This highlights that the *goodness* measure is highly correlated with the sum of ranking scores. That is the algorithms that perform well on *goodness* produce results sets which are not much different from top-k relevant set. The $exprel_\ell$ measure we propose appears to have no high correlation with other measures.

To optimize $exprel_\ell$ of the result set, we propose a greedy algorithm `BestCoverage`. Because of the submodular properties of $exprel_\ell$, `BestCoverage` is a $(1-1/e)$-approximation algorithm with complexity $\mathcal{O}(kn\Delta^\ell)$, where k is the number of recommended items, n is the number of vertices in the graph, and Δ is the maximum degree. We propose a relaxation of `BestCoverage` with complexity $\mathcal{O}(k\bar{\delta}^\ell\Delta^\ell)$, where $\bar{\delta}$ is the average degree of the graph. We experimentally show that the relaxation carries no significant harm to the *expanded relevance* of the results.

6.1 Background

6.1.1 Problem definition

We target the problem of diverse recommendation on graphs assuming that the user has a history or specified interests in some of the items. Therefore, the objective is to return a set of items which extend those interests.

Let $G = (V, E)$ be an undirected graph where $V = \{v_1, \ldots, v_n\}$ is the vertex set and E is the edge set. Given a set of m seed nodes $\mathcal{Q} = \{q_1, \ldots, q_m\}$ s.t. $\mathcal{Q} \subseteq V$, and a parameter k, return top-k items which are relevant to the ones in $\mathcal{Q}$. With diversity in mind, we want to recommend items not only relevant to $\mathcal{Q}$, but also covering different aspects of the query set

6.1.2 PageRank and personalized PageRank

We define a random walk on G arising from following the edges (links) with equal probability and a random restart at an arbitrary vertex with $(1-d)$ teleportation probability. The probability distribution over the states follows the discrete time evolution equation:

$$\mathbf{p}_{t+1} = \mathbf{P}\ \mathbf{p}_t, \tag{6.1}$$

where $\mathbf{p}_t$ is the vector of probabilities of being on a certain state at iteration t, and $\mathbf{P}$ is the transition matrix defined as:

$$\mathbf{P}(u, v) = \begin{cases} (1-d)\frac{1}{n} + d\frac{1}{\delta(v)}, & \text{if } (u, v) \in E \\ (1-d)\frac{1}{n}, & \text{otherwise,} \end{cases} \tag{6.2}$$

where $\delta(v)$ is the degree of the vertex $v \in V$. If the network is ergodic (i.e., irreducible and non-periodic), (6.1) converges to a stationary distribution $\pi = \mathbf{P}\pi$ after a number of iterations. And the final distribution π gives the PageRank scores [15] of the nodes based on *centrality*.

In our problem, a set of nodes $\mathcal{Q}$ was given as a query, and we want the random walks to teleport to only those given nodes. Let us define a prior distribution p^* such that

$$p^*(v) = \begin{cases} 1/m, & \text{if } v \in \mathcal{Q} \\ 0, & \text{otherwise.} \end{cases} \tag{6.3}$$

If we substitute the $(1/n)$s in (6.2) with p^*, we get a variant of PageRank, which is known as *personalized PageRank* (`PPR`) or *topic-sensitive PageRank* [54]. `PPR` scores can be used as the relevance scores of the items in the graph. Note that the rank of each seed node is reset after the system reaches to a steady state, i.e., $\forall q \in \mathcal{Q}$, $\pi_q \leftarrow 0$, since the objective is to extend $\mathcal{Q}$ with the results.

`PPR` is preferred as the scoring function in our discussions because (i) some of the methods in the experiments are variants of `PPR` which compute relevant but diverse set of results, (ii) some measures and objective functions are defined on the stationary distribution of `PPR`, and (iii) alternative scoring functions and probability distributions on graph produce similar results to `PPR`. On the other hand, the discussions on evaluations and some diversification techniques are independent of the preferred scoring function, hence we believe that the discussions will still interest the majority of the readers.

6.1.3 Result diversification on graphs

We classify the diversification methods for the recommendation problem based on whether the algorithm needs to rank the items only once or multiple times.

Diversification by query refinement. This set of algorithms rank the items k times to select the results one by one, and refine the search at each step.

GrassHopper [147] is a well-known diversification algorithm which ranks the graph k times by turning the highest-ranked vertex into a sink node at each iteration. Since the probabilities will be collected by the sink nodes when the random walk converges, the algorithm estimates the ranks with the number of visits to each node before convergence. GrassHopper uses matrix inversion to find the expected number of visits; however, inverting a sparse matrix makes it dense, which is not practical for the large and sparse graphs we are interested in. Therefore, we estimate the number of visits by iteratively computing the cumulative ranks of the nodes with PPR.

GSparse [74] employs an incremental ranking approach similar to GrassHopper, but the algorithm disconnects the selected node from the graph instead of converting it into a sink node. After executing the ranking function, the graph is sparsified for the next iteration by removing all the edges of the highest ranked node. This way, the graph becomes less dense around the selected nodes, hence, the remaining nodes at these regions will attract less visits during the random walk. The process is repeated until k nodes are selected.

Recently, manifold ranking has become an alternative to personalized PageRank and several diversification methods were proposed based on the idea of turning highly ranked nodes into sinks [27, 37, 146]. Aside from the ranking strategy, manifold ranking with sink points is quite similar to GrassHopper when the probabilities are estimated with cumulative scores. Since the manifold ranking is a different ranking of the graph, we carry out our experiments based only on PPR by leaving the discussion of using manifold ranking instead of PPR open for the time being.

Diversification by vertex selection. The following algorithms run the ranking function once, then carefully select a number of vertices to find a diverse result set.

DivRank [101] adjusts the transition matrix based on the number of visits to the vertices so far using a variant of random walks, called *vertex-reinforced random walks* (VRRW) [107]. It assumes that there is always an organic link for all the nodes returning back to the node itself which is followed with probability $(1-\alpha)$:

$$p_0(u,v) = \begin{cases} \alpha \frac{w(u,v)}{\delta(u)}, & \text{if } u \neq v \\ 1-\alpha, & \text{otherwise,} \end{cases} \tag{6.4}$$

where $w(u,v)$ is equal to 1 for $(u,v) \in E'$, and 0 otherwise. The transition matrix $\mathbf{P}_t$ at iteration t is computed with

$$\mathbf{P}_t(u,v) = (1-d)\ p^*(v) + d\ \frac{p_0(u,v)\ \eta_t(v)}{\sum_{z \in V} p_0(u,z)\ \eta_t(z)}, \tag{6.5}$$

where $p^*(v)$ is given in (6.3), and $\eta_t(v)$ is the number of visits of vertex v up to iteration t. It ensures that the highly ranked nodes collect more value over the iterations, resulting in the so called *rich-gets-richer* mechanism. In each iteration of VRRW, the transition probabilities from a vertex u to its neighbors are adjusted by the number of times they are visited up to that iteration t. Therefore, u gives a high portion of its rank to the frequently visited neighbors. Since the tracking of $\eta_t(.)$ is nontrivial, the authors propose to estimate it with *cumulative ranks* (CDivRank), i.e., the sum of the scores upto iteration t, or, since the ranks will converge after sufficient number of iterations, with *pointwise ranks* (PDivRank), i.e., the last score at iteration $t-1$.

A recently proposed algorithm, Dragon [128], employs a greedy heuristic to find a near-optimal result set that optimizes the *goodness* measure, which punishes the score when two neighbors are included in the results (see (6.15)). We will investigate this measure more in the upcoming section.

Frequently visited nodes tend to increase the ranks of their neighbors because of the smoothing process of random walks [101]. Based on this observation, algorithms using local maxima have been proposed. The Relaxed Local Maxima algorithm (k-RLM) [74] incrementally includes each local maxima within top-k^2 results to S until $|S| = k$ by removing it from the subgraph for the next iteration.

6.2 Measures and Evaluation

6.2.1 Classical relevance and diversity measures

Let us first review some classical measures for computing the relevance and diversity of the results with respect to the query. The measures are important since either they are typically used as –or a part of– the objective function of the diversification method, or the results are evaluated based on those measures.

Normalized relevance: The relevancy score of a set can be computed by comparing the original ranking scores of the resulting set with the top-k ranking list [128], defined as

$$rel(S) = \frac{\sum_{v \in S} \pi_v}{\sum_{i=1}^{k} \hat{\pi}_i}, \tag{6.6}$$

where $\hat{\pi}$ is the sorted ranks in non-increasing order.[41] Normalization with $\sum_{i=1}^{k} \hat{\pi}_i$ is preferred over $\sum_{v \in S} \pi_v$ since the distribution of scores in a random walk depends on

[41] $\hat{\pi}$ does not denote estimated or predicted relevance scores.

the graph size, query, connectivity, etc., and normalized scores are comparable among different settings.

Difference ratio: A diversified result set is expected to be somewhat different than the top-k relevant set. Because the highly ranked nodes increase the ranks of their neighbors [101], the top-k results, recommended by the original `PPR`, is not diverse enough as shown in [111] and in our experiments. Nevertheless, the original result set has the utmost relevancy. This fact can mislead the evaluation of the experimental results. Therefore, we decided to measure the difference of each result set from the set of original top-k nodes. Given $\hat{S}$ to be the top-k relevant set, the difference ratio is computed with

$$\mathit{diff}(S, \hat{S}) = 1 - \frac{|S \cap \hat{S}|}{|S|}. \tag{6.7}$$

nDCG: We use normalized discounted cumulative gain (nDCG), for measuring the relevancy as well as the ordering of the results. It is defined as

$$\mathrm{nDCG}_k = \frac{\pi_{s_1} + \sum_{i=2}^{k} \frac{\pi_{s_i}}{\log_2 i}}{\hat{\pi}_1 + \sum_{i=2}^{k} \frac{\hat{\pi}_i}{\log_2 i}}, \tag{6.8}$$

where π is the relevancy vector (e.g., stationary distribution of a random walk), $\hat{\pi}$ is the sorted π in non-increasing order, and $s_i \in S$ is the i^{th} point in result set S.

ℓ-step graph density: A variant of graph density measure is the ℓ-step graph density [128], which takes the effect of in-direct neighbors into account. It is computed with

$$\mathit{dens}_\ell(S) = \frac{\sum_{u,v \in S, u \neq v} d_\ell(u, v)}{|S| \times (|S| - 1)}, \tag{6.9}$$

where $d_\ell(u, v) = 1$ when v is reachable from u within ℓ steps, i.e., $d(u, v) \leq \ell$, and 0 otherwise. The inverse of $\mathit{dens}_\ell(S)$ is used for the evaluation of diversity in [101].

ℓ-expansion ratio: As an alternative to density, *expansion ratio* and its variant *ℓ-expansion ratio* [90] measure the coverage of the graph by the solution set, computed with:

$$\sigma_\ell(S) = \frac{|N_\ell(S)|}{n}, \tag{6.10}$$

where the *expansion set* with 1-distance neighbors is defined as $N(S) = S \cup \{v \in (V - S) : \exists u \in S, (u, v) \in E\}$, and the *$\ell$-step expansion set* is defined in [90] as:

$$N_\ell(S) = S \cup \{v \in (V - S) : \exists u \in S, d(u, v) \leq \ell\}. \tag{6.11}$$

Note that the intent-aware measures, such as intent-aware expected reciprocal rank (ERR-IA) [24], α-normalized discounted cumulative gain (α-nDCG@k) [30], intent-aware mean average precision (MAP-IA) [4], are not included to the discussions, but they are important measures for evaluating the diversity of the results when data and queries have some already known categorical labels. Our problem has no assumptions of a known distribution that specifies the probability of an item belonging to a category.

6.2.2 Bicriteria optimization measures

Maximum Marginal Relevance (MMR) [18] is the most popular diversification method that optimizes a bicriteria objective, *marginal relevance*, which is a linear combination of independently measured relevance and novelty. The method greedily and implicitly optimizes the following objective assuming that the similarity of all items to the query items are already computed in π:

$$f_{MMR}(S) = (1 - \lambda) \sum_{v \in S} \pi_v - \lambda \sum_{u \in S} \max_{\substack{v \in S \\ u \neq v}} sim(u, v), \tag{6.12}$$

where λ is the importance of relevance over novelty and *sim* is a similarity metric. The problem with (6.12) is that two different measures are aggregated without taking their compatibility into account.

The same premise is also valid for any type of linear aggregation of a relevance and a diversity measure. For example, [90] tries to optimize the following diversified ranking measure:

$$f_L(S) = \sum_{v \in S} \pi_v + \lambda \frac{|N(S)|}{n}, \tag{6.13}$$

where λ is the tradeoff between relevance and diversity, and the diversity of the result set is measured with the expansion ratio. Similarly in [89], relevance part is scaled with $(1 - \lambda)$.

Other bicriteria objectives include max-sum diversification, which reduces to MaxSumDispersion problem, max-min diversification, which reduces to MaxMinDispersion problem, etc. For example, *k-similar diversification set problem* [131] is defined based on MaxSumDispersion as:

$$f_{MSD}(S) = (k-1)(1-\lambda) \sum_{v \in S} \pi_v + 2\lambda \sum_{u \in S} \sum_{\substack{v \in S \\ u \neq v}} div(u, v), \tag{6.14}$$

where $div(u, v)$ can be selected as a weighted similarity, *tf/idf* cosine similarity, or the Euclidean distance depending on the problem. We refer the reader to [46] for more information on objectives and distance functions.

6.2.3 Bicriteria optimization is not the answer

We argue that bicriteria optimization is inappropriate, and hence, the diversification methods that seem to optimize both criteria are problematic. Let us return back to our original problem: the items in a graph structure are ranked based on a given query and a ranking method (e.g., PPR), and our aim is to rerank those items so that we can include more results from different aspects of the query and reduce redundancy of top-k relevant set.

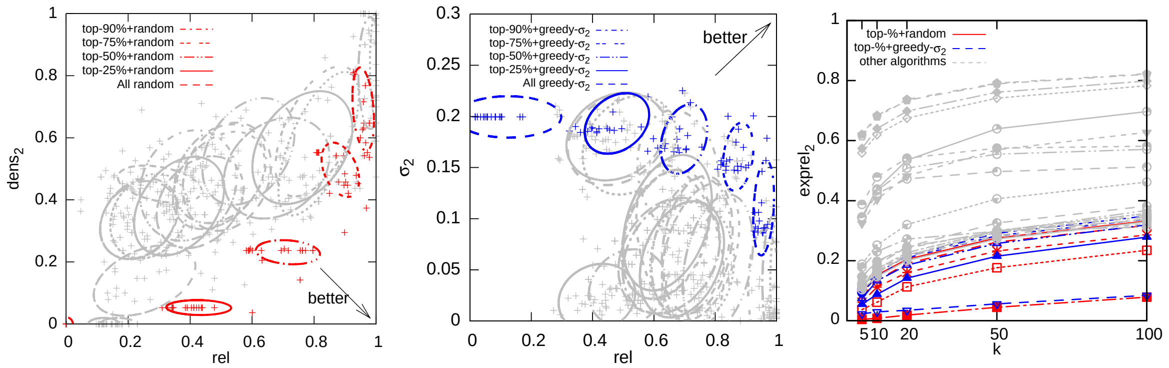
Suppose that we work on the web graph and we want to *diversify* the results of a search engine which displays $k = 10$ results to the user. Do you think the quality of the top-k list would improve if we replace some results from the end of the list with random web pages?

We design two *query-oblivious* algorithms for the two popular combinations of objectives, which are monotonous (e.g., linear or quadratic) aggregations of max-sum relevance and max-sum diversity (graph density *dens* or expansion ratio σ) objectives. The algorithms will return some of the results without considering the user's interests, yet, will perform the best on the following commonly used measures:

- **top-%+random:** returns a given percentage of the results (e.g., 50%, 75%, etc.) from top-k, and the rest randomly from the search space.
- **top-%+greedy-σ_2:** returns a given percentage of the results (e.g., 50%, 75%, etc.) from top-k, and try to maximize σ_2 with the rest of the results without taking the query into account.

To prove our point, we compute the normalized relevance (*rel*) and selected diversity measure ($dens_2$ and σ_2) of the results for the diversification methods in the literature and for the *query-oblivious* algorithms.[42] We fit a multi-variate Gaussian on top of the results to show the mean and moments of the distribution when two

[42]Figure 6.1 gives only the results for AMAZON0601 dataset using scenario 3 queries and $k = 20$. Comparisons of *query-oblivious* methods on given bicriteria measures and $exprel_2$ for other datasets and query types are provided in: http://bmi.osu.edu/hpc/data/Kucuktunc13WWW/randoms.pdf



(a) *rel* vs. $dens_2$ (b) *rel* vs. σ_2 (c) $exprel_2$

Figure 6.1: Evaluation of top-%+random (red) and top-%+greedy-σ_2 (blue) methods versus other algorithms (gray) based on selected relevance/diversity measure pairs and combined $exprel_2$ measure. The other algorithms include GRASSHOPPER, DIVRANK, DRAGON, k-RLM, GSPARSE, and `BestCoverage`, but they are not highlighted here since we do not want to prematurely compare those against each other.

objectives are considered simultaneously. A result which further minimizes $dens_2$ and maximizes *rel* and σ_2 is favorable and better. This is shown with an arrow in the Figs. 6.1(a) and 6.1(b).

Figure 6.1(a) shows the results of **top-%+random** as well as other algorithms with respect to *rel* vs. $dens_2$ evaluation. Figure 6.1(b) similarly shows the results of **top-%+greedy-**σ_2 as well as other algorithms with respect to *rel* vs. σ_2 evaluation. Here, *query-oblivious* methods seem to recommend the best result sets when a bicriteria evaluation is used. Yet, we know that those algorithms are designed to trick the evaluation, as well as produce useless results in user's point of view.

Using only the first half of top-k results gives a normalized relevance score greater than or equal to 0.5 since the ranks are sorted in non-increasing order. Furthermore, the ranks has a power-law distribution that makes *rel* much higher than 0.5. Therefore, the relevance objective is mostly satisfied.

We further argue that no matter which relevance and diversity measures are selected, there always exists a *query-oblivious* algorithm which optimizes both measures in a meaningless way, useless in practice but looks great on the paper. This is the problem of evaluating result diversification as a bicriteria optimization problem with **a relevance measure that ignores diversity**, and **a diversity measure that ignores relevancy**.

6.2.4 Combined measures

As a result of our experiments on bicriteria optimization, we argue that we need a combined measure that tightly integrates both relevance and diversity aspects of the result set. It is reasonable to design the combined measure based on the query, the rankings, and the graph structure we already have.

The *goodness* measure [128] is a step towards a meaningful combined measure. It penalizes the score when two results share an edge, meaning that they are neighbors and they possibly increase their ranks by feeding each other during the random walk. The measure is computed with

$$f_G(S) = 2\sum_{i\in S}\pi_i - d\sum_{i,j\in S}\mathbf{A}(j,i)\pi_j - (1-d)\sum_{j\in S}\pi_j\sum_{i\in S}p^*(i), \tag{6.15}$$

where $\mathbf{A}$ is the row-normalized adjacency matrix of the graph. However, we will show in Section 6.2.5 that *goodness* is highly dominated by relevance, which reflects negatively on the results of Dragon in the experiments.

We present a combined measure of the ℓ-step expansion ratio (σ_2) and relevancy scores (*rel*), which are two popular diversity and relevance measures in the literature, in order to quantify the *relevant-part coverage* of the graph:

ℓ-step expanded relevance:

$$\mathit{exprel}_\ell(S) = \sum_{v \in N_\ell(S)} \pi_v \tag{6.16}$$

where $N_\ell(S)$ is the ℓ-step expansion set of the result set S, and π is the `PPR` scores of the items in the graph.

This new measure explicitly evaluates the diversity of the results in terms of *coverage* with the given set. In other words, when two results are close to each other in the graph, their expansion sets intersect narrowing the covered part of the search space. Therefore, the items having separate expansion sets will increase the coverage. However, coverage is not the only aspect of exprel_ℓ. The proposed measure also takes the ranking scores into account, and hence the quality of the covered part.

The effect of each result is limited with the given ℓ parameter, i.e., a result covers only its neighbors in the graph if $\ell = 1$, or neighbors of neighbors if $\ell = 2$. Higher values of ℓ are generally not preferred since the expansion set tends to cover most of the graph in those cases.

An important property of exprel_ℓ measure is that *query-oblivious* algorithms cannot optimize it. Because, the highest ranked items are mostly not diverse enough, and the rest of the results (randomly selected independent of the query) will not contribute much to the measure. Figure 6.1(c) shows that neither top-%+random (red) nor top-%+greedy-σ_2 (blue) can optimize the measure while the diversification algorithms (gray) can score higher. This proves the validity of the measure for diversification.

6.2.5 Correlations of the measures

We investigate the mentioned relevance, diversity, and combined measures by computing their pairwise correlations based on the results of the algorithms given in Section 6.1.3 as well as the query-oblivious top-%+random methods given in the previous section. Table 6.1 shows the correlations of 10 measures as scatter plots as well as their correlation scores.[43]

For the relevance measures, *rel* is highly correlated with nDCG although the latter considers the order of the results. *rel* is also anti-correlated with *diff*, meaning that as the ratio of results other than top-k start to increase, the normalized relevance decreases accordingly.

For the graph diversity measures, ℓ-step expansion ratios (σ_1 and σ_2) are highly correlated among each other. On the other hand, graph density-based measures ($dens_1$ and $dens_2$) do not seem to have any high correlation with other measures.

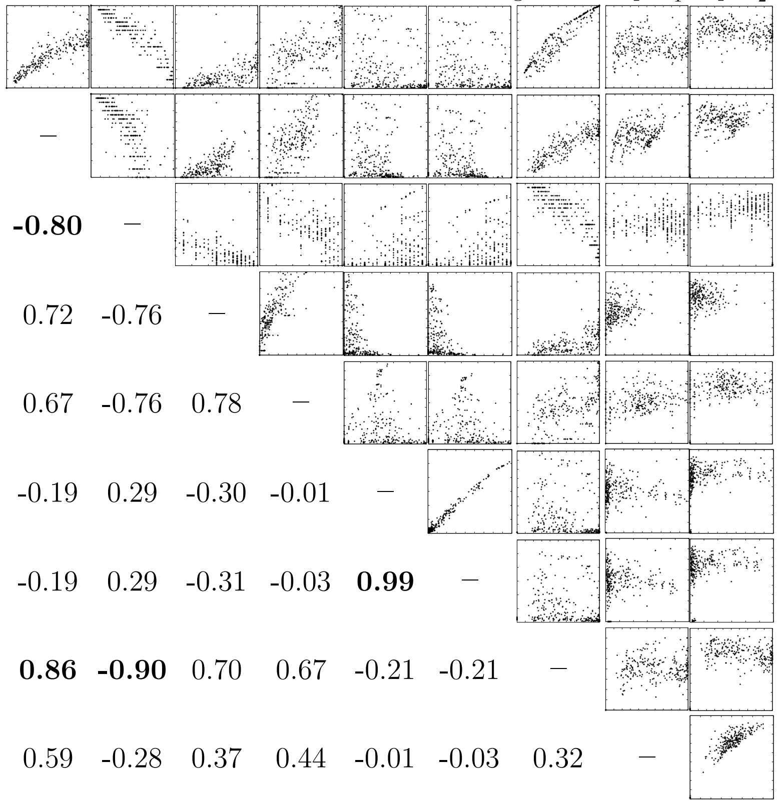
Among the combined measures, *goodness* is highly correlated with *rel*. This highlights that the *goodness* measure is dominated by the sum of ranking scores, meaning that algorithms that perform better on *goodness* do not return results that are much different from the top-k results of `PPR`.

The proposed $exprel_\ell$ measure, on the other hand, appears to have no high correlation with any of the other relevance or diversity measures, proving that it is something different than the already known measures. Although the expanded relevance is based on both *rel* and expansion ratio (σ), very low correlation is observed in the results.

[43]Table 6.1 shows only the measure correlations on AMAZON0601 dataset and with $k = 20$. The results are consistent across various datasets, scenarios, and k values. A complete comparison set is provided in: `http://bmi.osu.edu/hpc/data/Kucuktunc13WWW/corr.pdf`

Table 6.1: Correlations of the different relevance, diversity, and combined measures. Pearson correlation scores are given on the lower triangle of the matrix. High correlations are highlighted.

	rel	nDCG	*diff*	$dens_1$	$dens_2$	σ_1	σ_2	goodness	$exprel_1$	$exprel_2$
rel	–									
nDCG	**0.87**	–								
diff	**-0.95**	**-0.80**	–							
$dens_1$	0.74	0.72	-0.76	–						
$dens_2$	0.76	0.67	-0.76	0.78	–					
σ_1	-0.25	-0.19	0.29	-0.30	-0.01	–				
σ_2	-0.25	-0.19	0.29	-0.31	-0.03	**0.99**	–			
goodness	**0.96**	**0.86**	**-0.90**	0.70	0.67	-0.21	-0.21	–		
$exprel_1$	0.34	0.59	-0.28	0.37	0.44	-0.01	-0.03	0.32	–	
$exprel_2$	0.20	0.37	-0.13	0.17	0.33	0.26	0.23	0.21	**0.86**	–



6.3 Best Coverage Method

Our strategy so far was to review the attempts to find a good objective function for the result diversification problem on graphs. We have shown that a bicriteria optimization of relevance and diversity can be tricked, and a combined measure should be constructed carefully. The proposed $exprel_\ell$ measure seems to cover both aspects of the intended objective, yet cannot be optimized by the query-oblivious algorithms.

We argue that this novel measure can be naturally used as an objective function of a diversification algorithm.

6.3.1 Problem formulation and complexity

Given a graph $G = (V, E)$, a vector of ranking scores π (stationary distribution of `PPR` scores in our case) computed based on the query set $\mathcal{Q}$, and the number of required results k, our objective is to maximize the expanded relevance ($exprel_\ell$) of the result set S:

$$S = \operatorname*{argmax}_{\substack{S' \subseteq V \\ |S'|=k}} exprel_\ell(S') = \operatorname*{argmax}_{\substack{S' \subseteq V \\ |S'|=k}} \sum_{v \in N_\ell(S')} \pi_v, \tag{6.17}$$

where $N_\ell(S')$ is the ℓ-step expansion set. We refer to this problem as *$exprel_\ell$-diversified top-k ranking* (DTRℓ).

However, it is not hard to see that the objective of finding a subset of k elements that maximizes the expanded relevance is NP-hard. Assuming the graph G and the ranking scores π are arbitrary, DTRℓ is a generalization of the *weighted maximum coverage problem* (WMCP) which is NP-Complete [57]. WMCP is expressed as a set O of objects o_i with a value ω_i and z sets of objects $r_j \subseteq O$, $R = \{r_1, r_2, \ldots, r_z\}$. The problem is to select a subset of R, $P \subseteq R$ such that $|P| = x$ which maximizes $\sum_{o_i \in \{r_j : r_j \in P\}} \omega_i$. The key of the reduction for $\ell = 1$ is to construct an instance of DTRℓ with a bipartite graph $G = (V = R \cup O, E)$ where $(r_j, o_i) \in E$ iff $o_i \in r_j$. We set $\pi_{r_j} = 0$, $\pi_{o_i} = \omega_i$ and $k = x$. The solutions of DTRℓ are dominated by sets S where all the vertices are in R. Indeed, since $\pi_{r_j} = 0, \forall r_j$ there is no advantage in selecting a vertex in O. The rest of the reduction is obvious for $\ell = 1$. For other values of ℓ, the reduction is similar, except each edge of the bipartite graph is replaced in a path of ℓ edges.

Note that the proposed objective in (6.17) is independent of ordering since the function is defined over an unordered set. This is usually reasonable because there is an assumption that users will consider all k results [4, 90, 128]. In practice, different users may stop at different number of results, hence, several DCG-based metrics are commonly used to compute the importance of returning results in an *ideal ordering*. The near-optimal solutions that we will present in the following section can still output an ordered set of results based on the marginal utility of each selected item at the moment of its inclusion.

6.3.2 Greedy solution: BestCoverage

Although the optimal solution of the proposed objective function (see (6.17)) is NP-hard, we will show that a greedy solution that selects the item with the *highest marginal utility* at each step is the best possible polynomial time approximation for the problem.

Let us define the *marginal utility* for a given vertex v and result set S as $g(v, S)$, such that $g(v, \emptyset) = \mathit{exprel}_\ell(\{v\})$ before any results are selected, and $g(v, S) = \sum_{v' \in V'} \pi_{v'}$ where $V' = N_\ell(\{v\}) - N_\ell(S)$ represents the ℓ-step expansion set of vertex v without the items that have already been covered by another result. In other words, $g(v, S)$ is the increase on the exprel_ℓ measure if v is included to the result set, i.e., $\mathit{exprel}_\ell(S \cup \{v\}) = \mathit{exprel}_\ell(S) + g(v, S)$.

Algorithm 15 incrementally selects the item with the highest marginal utility in each step, then includes it to the result set S. This way, the items that contribute the most to the expanded relevance of the final results are greedily selected as a solution

Algorithm 15: `BestCoverage`

```
Input: $k, G, \pi, \ell$
Output: a list of recommendations $S$
$S = \emptyset$
while $|S| < k$ do
    $v^* \leftarrow \operatorname{argmax}_v g(v, S)$
    $S \leftarrow S \cup \{v^*\}$
return $S$
```

to the given optimization problem. In order to show that the greedy algorithm solves the problem quite well, we first prove that the $exprel_\ell$ is a submodular function:

Definition 6.3.1. (SUBMODULARITY) *Given a finite set V, a set function $f : 2^V \to \mathbb{R}$ is submodular if and only if for all subsets S and T such that $S \subseteq T \subseteq V$, and $j \in V \setminus T$, $f(S \cup \{j\}) - f(S) \geq f(T \cup \{j\}) - f(T)$.*

Lemma 6.3.2. *$exprel_\ell$ is a submodular function.*

The proof of the lemma follows directly from the definitions of submodularity and $exprel_\ell$. Greedy algorithms are known to generate good solutions when maximizing submodular functions with a cardinality constraint and were used in [4, 90].

Theorem 6.3.3. *[102] For a submodular set function f, let S^* be the optimal set of k elements that maximizes $f(S)$, and S' be the k-element set constructed greedily by selecting an element one at a time that gives the largest marginal increase to f. Then $f(S') \geq (1 - 1/e)f(S^*)$.*

Corollary 6.3.4. *`BestCoverage` is an $(1 - 1/e)$-approximation algorithm for the $exprel_\ell$-diversified top-k ranking problem.*

6.3.3 Analysis and relaxation of the algorithm

`BestCoverage` (`BC`) is a $(1-1/e)$-approximation for maximizing $exprel_\ell$ with complexity $\mathcal{O}(kn\Delta^\ell)$ where n is the number of vertices in the graph, k is the number of recommended objects, and Δ is the maximum degree of the graph.

Obviously, the implementation in Algorithm 15 can be improved by storing the *marginal utility* for every vertex at the expense of $\mathcal{O}(n)$ space, and updating only the vertices that the inclusion of v^* to S would affect. However, for $\ell = 2$, the number of vertices to be updated is $|N_4(\{v^*\})|$, which is $\mathcal{O}(\Delta^4)$ in the worst case. Initializing the marginal utility incurs a cost of $\mathcal{O}(n\Delta^\ell)$. Once a vertex is added to set S, the impact of its distance ℓ neighbors must be adjusted. For a given vertex, adjusting its impact costs $\mathcal{O}(\Delta^\ell)$. For each iteration of the algorithm the impact of at most Δ^ℓ neighbors need to be adjusted. Though, each vertex adjusts its impact only once, so there are $\mathcal{O}(\min\{n, k\Delta^\ell\})$ adjustments. Finally, selecting the vertex with maximal marginal utility requires $\mathcal{O}(n)$ operations[44] per iteration. The overall complexity of the algorithm is $\mathcal{O}(n\Delta^\ell + \min\{n, k\Delta^\ell\}\Delta^\ell + kn)$.

With this optimization, most of the time is spent on initializing the marginal utility. We experimentally found that the returned results are chosen from top-k' results of `PPR` ranks, where k' is proportional to k and the average degree of the graph. We propose a relaxation of `BestCoverage` which only considers including in the result set the top-$k\bar{\delta}^\ell$ highest ranked vertices solely based on the relevance scores where $\bar{\delta}$ is the average degree of the graph. All the vertices of the graph still contributes to marginal utility. The complexity of the relaxed version drops to $\mathcal{O}(\min\{n, k\Delta^\ell\}\Delta^\ell +$

[44]It might appear that using a fibonnacci heap should allow to reach a better complexity, but we require the extract-max and decrease-key operations which are incompatible.

Algorithm 16: `BestCoverage (relaxed)`

```
Input: $k, G, \pi, \ell$
Output: a list of recommendations $S$
$S = \emptyset$
$\textsc{Sort}(V)$ w.r.t $\pi_i$ non-increasing
$S1 \leftarrow V[1..k']$, i.e., top-$k'$ vertices where $k' = k\bar{\delta}^{\ell}$
$\forall v \in S1, g(v) \leftarrow g(v, \emptyset)$
$\forall v \in S1, c(v) \leftarrow \textsc{Uncovered}$
while $|S| < k$ do
    $v^* \leftarrow \mathrm{argmax}_{v \in S1}\, g(v)$
    $S \leftarrow S \cup \{v^*\}$
    $S2 \leftarrow N_\ell(\{v^*\})$
    for each $v' \in S2$ do
        if $c(v') = \textsc{Uncovered}$ then
            $S3 \leftarrow N_\ell(\{v'\})$
            $\forall u \in S3, g(u) \leftarrow g(u) - \pi_{v'}$
            $c(v') \leftarrow \textsc{Covered}$
return $S$
```

$k \min\{n, k\bar{\delta}^{\ell}\})$ since the cost of the computation of the initial marginal utility is now asymptotically dominated by the cost of adjusting them. Algorithm 16 gives the relaxed `BestCoverage` algorithm with all mentioned improvements. The impact of the relaxation on the quality of the solution will be discussed in Section 6.4.3.

6.4 Experiments

6.4.1 Datasets

In the experiments we use one graph instance for each targeted application area, i.e., product recommendation on shopping websites, collaborator and patent recommendation in academia, friend recommendation on social networks, and personalized web search. The graphs are publicly available at Stanford Large Network Dataset

Table 6.2: Properties of the graphs used in experiments.

Dataset	$\|V\|$	$\|E\|$	$\bar{\delta}$	D	$D_{90\%}$	CC
AMAZON0601	403.3K	3.3M	16.8	21	7.6	0.42
CA-ASTROPH	18.7K	396.1K	42.2	14	5.1	0.63
CIT-PATENTS	3.7M	16.5M	8.7	22	9.4	0.09
SOC-LIVEJOURNAL1	4.8M	68.9M	28.4	18	6.5	0.31
WEB-GOOGLE	875.7K	5.1M	11.6	22	8.1	0.60

Collection[45]. In summary, AMAZON0601 is the Amazon product co-purchasing network collected on June 2003. CA-ASTROPH is the collaboration network between authors of the papers submitted to arXiv astrophysics category. CIT-PATENTS is the citation network between U.S. patents granted between 1975 and 1999. SOC-LIVEJOURNAL1 is the graph of LiveJournal social network, and WEB-GOOGLE is the web graph released in 2002 by Google.

The mentioned graphs are re-labeled, converted into undirected graphs. The properties of the graphs are given in Table 6.2. Note that $\bar{\delta}$ is the average degree of the graph, D is the diameter of the graph, i.e., maximum undirected shortest path length, $D_{90\%}$ is the 90-percentile effective diameter, and CC is the average clustering coefficient.

6.4.2 Scenarios and query generation

We generate the queries for the experiments based on three different real-world scenarios:

Scenario 1: A random vertex in the graph is selected as the query. This scenario represents the case where the system does not have any information on the user. For

[45]Available at: http://snap.stanford.edu/data/index.html

product recommendation, the user can be visiting a product page without signing in to the system. For academic recommendation tasks, a professor can be looking for collaborators.

Scenario 2: A random vertex v along with 10–100 vertices within two distance to v are selected as a query. In this scenario, v and the selected vertices represent an area of interest. For example, the user can be searching for a product within a category, or interested in an academic field. In a social network, the friend list of a person can be used as the query for friend suggestion.

Scenario 3: 2 to 10 random vertices are selected as different interests of the user, and a total of 10 to 100 vertices around those interests are added to the query set. Multiple areas of interest is the most common use case for these applications where users are registered to the system and already have a search or purchase history.

For each dataset, 750 queries were generated, where the average number of the seed nodes varies between 1 and 50 for the scenarios 1 and 3, respectively. In total 3,750 query sets representing different real-world cases were used in the experiments.

6.4.3 Results

We experiment with the algorithms given in Section 6.1.3, the datasets described in Section 6.4.1, and the queries defined in Section 6.4.2. For the methods that use the ranking scores of `PPR`, we fix $d = 0.9$ and the number of `PPR` iterations to 20 in order to be consistent between different queries. For the VRRW computation of DivRank methods, we set $\alpha = 0.25$ and the number of iterations to 50 since VRRW usually takes more iterations to converge. All ranking functions are implemented efficiently with sparse matrix-dense vector multiplication (SpMxV) operations.

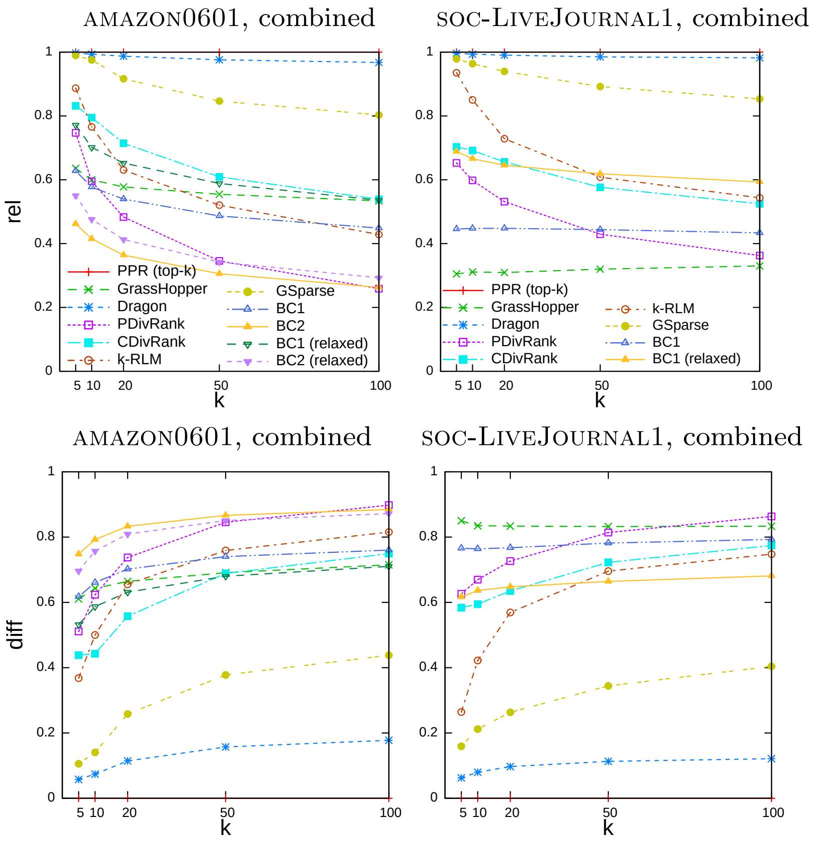


Figure 6.2: Normalized relevance (*rel*) and different ratio (*diff*) scores with varying k. Dragon and GSparse return results around 70% similar to the top-k relevant set, this is generally not enough to improve the diversity of the results.

On amazon0601, ca-AstroPh, and soc-LiveJournal1 datasets, we observed that the results of different scenarios are similar. Hence, we combine the scenarios and display the results on all queries[46]. Also note that the results of $\mathtt{BC}_2$ and its relaxation are omitted from the plots of soc-LiveJournal1 dataset because of the impractical runtimes.

Normalized relevance (*rel*) and difference ratio (*diff*) plots in Figure 6.2 show that Dragon and GSparse methods almost always return the results having 70%

[46]Due to space limitation we only display one plot per observation highlighted in the text. The complete set of plots for each dataset, scenario, and measure is provided in the supplementary material: `http://bmi.osu.edu/hpc/data/Kucuktunc13WWW/results.pdf`

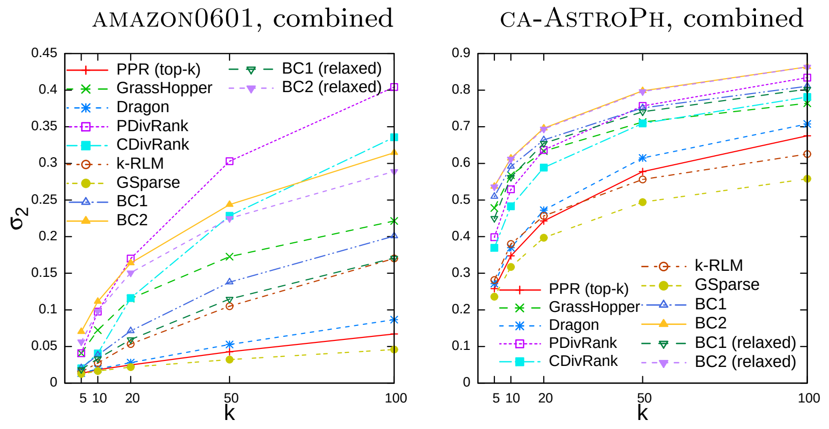


Figure 6.3: Coverage (σ_2) of the algorithms with varying k. `BestCoverage` and DIVRANK variants have the highest coverage on the graphs while DRAGON, GSPARSE, and k-RLM have similar coverages to top-k results.

similar items to top-k relevant set, and more than 80% *rel* score. A low *rel* score is not an indication of being dissimilar to the query (unless *rel* $\to 0$); on the other hand, since the scores have a power-law distribution, a high *rel* score usually implies that the algorithm ignored the diversity of the results and did not change many results in order to keep the relevancy high. The actual *diff* measures are also given in Figure 6.2.

Based on the expansion ratios (σ_2) in Figure 6.3, `BestCoverage` and DIVRANK variants, especially PDIVRANK and $\texttt{BC}_2$, have the highest scores, hence the highest coverage on the graphs with their diversified result set. DRAGON, GSPARSE, as well as k-RLM have expansion ratios similar to the top-k results, meaning that these algorithms do not improve the coverage of the given graphs enough. GSPARSE reduces the expansion ratio even more than the top-k set, proving that it is inappropriate for the diversification task. It is important to note that σ_2 scores are meaningless by itself since query-oblivious greedy-σ_2 algorithm would maximize the coverage.

Figure 6.4 shows the proposed expanded relevance scores (exprel_2) of the result sets. $\texttt{BC}_1$ and $\texttt{BC}_2$ variants are significantly better than the other algorithms, where

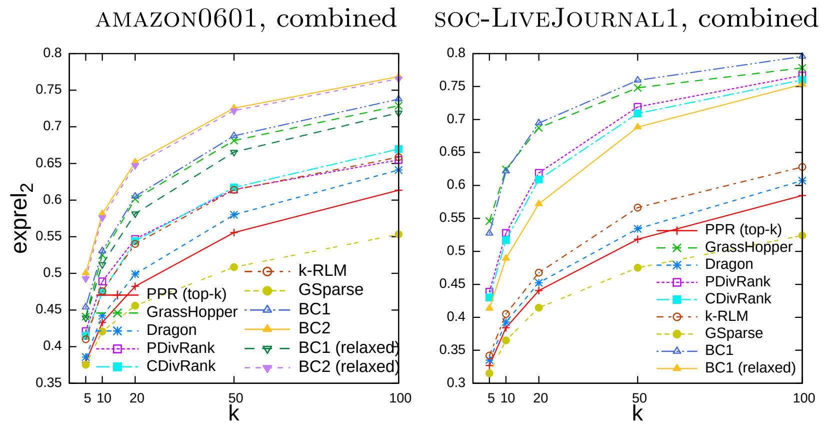


Figure 6.4: Expanded relevance (*exprel*$_2$) with varying k. $\mathtt{BC}_1$ and $\mathtt{BC}_2$ variants mostly score the best, GrassHopper performs high in soc-LiveJournal1. Although PDivRank gives the highest coverage on amazon0601 (Fig. 6.3), it fails to cover the relevant parts.

GrassHopper is able to score closer to `BestCoverage` only in soc-LiveJournal1 dataset. Although DivRank variants perform the highest based on expansion ratio (see Figure 6.3), their results are shown to be unable to cover the relevant parts of the graph as they score lower than `BestCoverage` variants.

For cit-Patents and web-Google datasets, we report the results on queries of scenarios 1 and 3 separately. Here we omit the results of scenario-2 queries since they are in between scenarios 1 and 3. These plots share the conclusions we have made so far based on the results on previous three datasets; however, they present different behavior based on the chosen scenario, so we provide a deeper analysis on those.

Figure 6.5 shows that the *exprel*$_2$ results on cit-Patents dataset vary based on the scenario chosen to generate the queries. In fact, the results are higher than normal for scenario-1 queries. This is because of the low average degree ($\bar{\delta} = 8.7$) and low clustering coefficient ($CC = 0.09$) of the graph. Also note that the relaxations of $\mathtt{BC}_1$

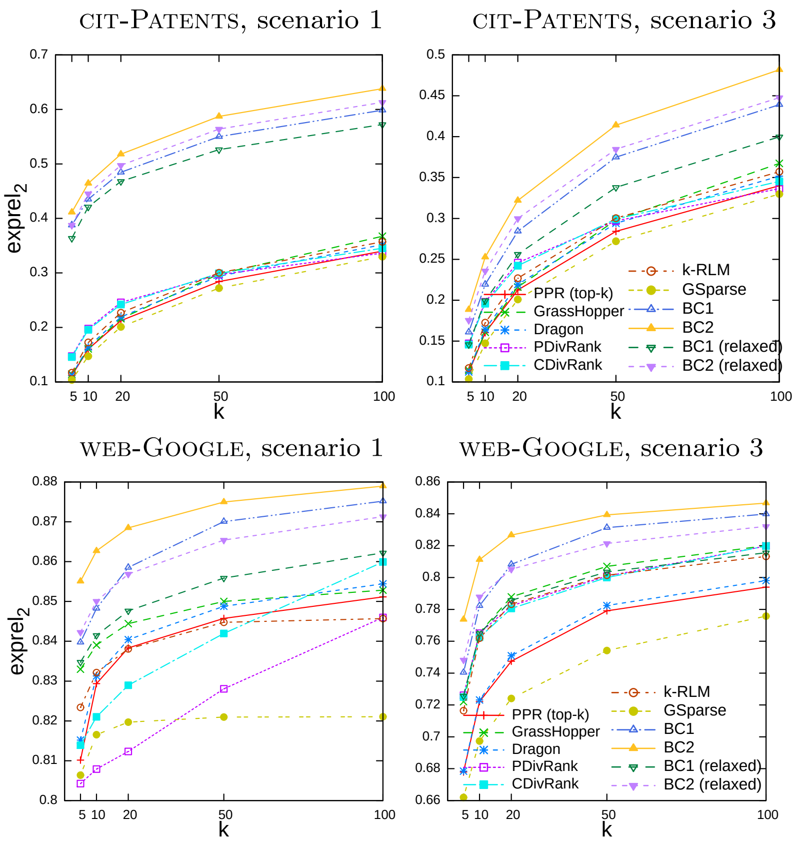


Figure 6.5: Expanded relevance (*exprel*$_2$) with varying k. `BestCoverage` variants perform higher than usual on cit-Patents dataset with scenario-1 queries because of the low average degree ($\bar{\delta} = 8.7$) and low clustering coefficient ($CC = 0.09$) of the graph. The relaxed algorithms perform closer to their originals, meaning that they were both efficient and effective on this type of sparsely connected graphs.

and $\mathtt{BC}_2$ perform closer to $\mathtt{BC}_1$ and $\mathtt{BC}_2$, meaning that the relaxed algorithms are both efficient and also effective on this type of sparsely connected graphs.

It is also more clear on plots in Figure 6.6 that DivRank variants implicitly optimize the expansion ratio (σ_2) of the results, but without considering whether those results are still relevant to the query. As a striking example of scenario-1 queries on web-Google dataset, it is quite interesting to see an algorithm to perform the best with respect to the *size* of the expansion set, but almost the worst with respect to the *relevancy* of the same set (see Figure 6.5).

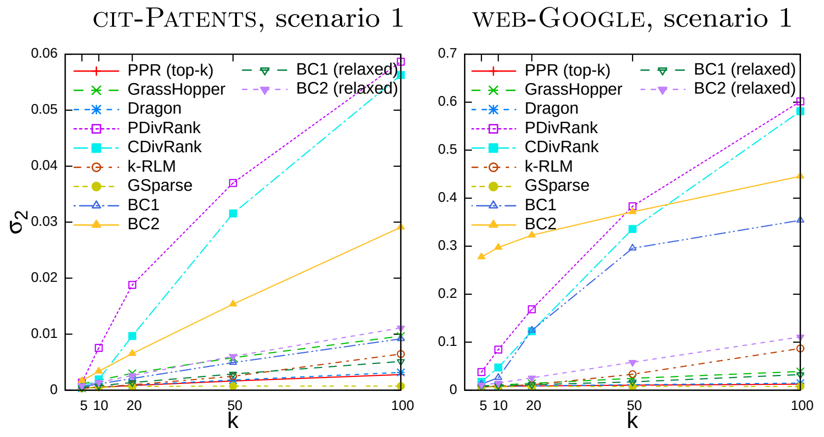


Figure 6.6: Coverage (σ_2) of the algorithms with varying k. DIVRANK variants appear to be implicitly optimizing the *size* of the expansion set, without considering whether those results are still relevant to the query (cf. corresponding *exprel*$_2$ in Figure 6.5).

With the runtime experiments shown in Figure 6.7, we also confirm that the relaxed variants of `BestCoverage` perform closer to their originals (see Figure 6.4) with an order of magnitude or more gain in efficiency. In all cases, even in SOC-LIVEJOURNAL1, which is the largest dataset in our experiments, the $\texttt{BC}_1$ method always performs better with a running time less than GRASSHOPPER and DIVRANK variants, while the relaxed version scores closer enough with a running time slightly higher than the original `PPR` computation. Therefore, in terms of the running times, the efficient algorithms are generally ordered according to $\texttt{PPR} \leq k - \text{RLM} \leq \texttt{BC}_1\texttt{(relaxed)} \leq \text{DRAGON} \leq \texttt{BC}_1$. Confirming the observation in [101], DIVRANK variants are more efficient than GRASSHOPPER for $k > 10$. Runtime of $\texttt{BC}_2$ depends on the dataset properties while its relaxed variant has comparable running times to DIVRANK variants. Both $\texttt{BC}_2$ and its variant has a very high runtime on CA-ASTROPH since this dataset has the highest average degree ($\bar{\delta} = 42.2$) and the

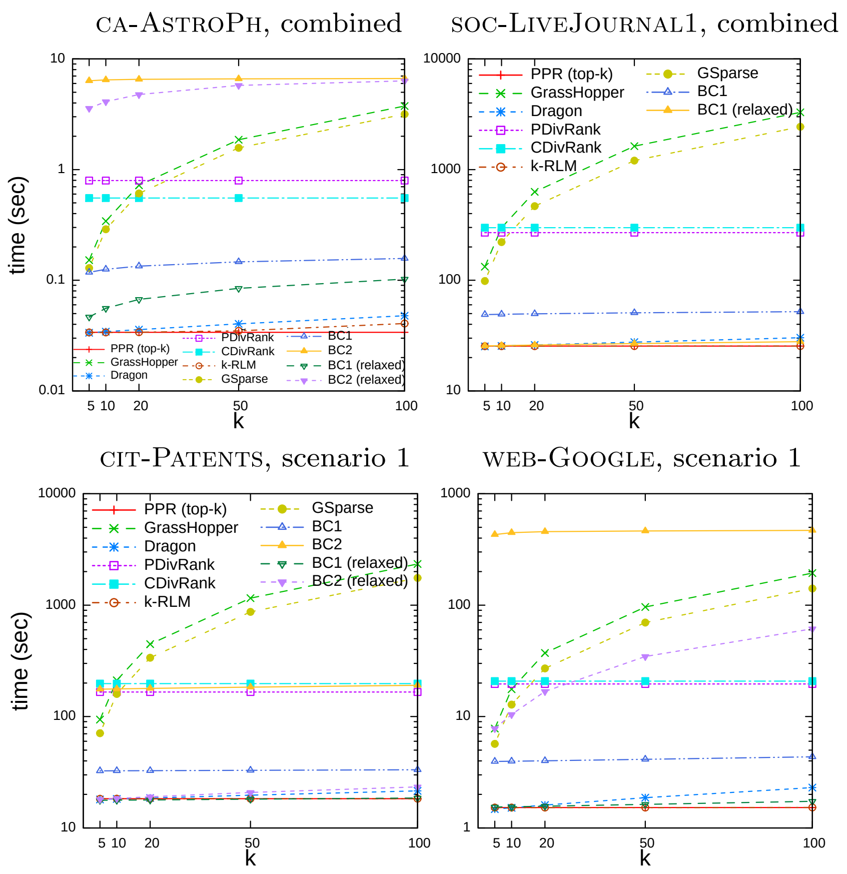


Figure 6.7: Running times of the algorithms with varying k. $\texttt{BC}_1$ method always perform better with a running time less than GRASSHOPPER and DIVRANK variants, while the relaxed versions score similarly with a slight overhead on top of the $\texttt{PPR}$ computation.

clustering coefficient ($CC = 0.63$), hence, each $exprel_2$ computation is more costly than the ones on other datasets.

6.4.4 Intent-aware results

Among the five datasets we selected for the experiments, CIT-PATENTS has the categorical information. One of the 426 class labels was assigned to each patent, where those classes hierarchically belong to 36 subtopics and 6 high-level topics[47].

[47]Available at: http://data.nber.org/patents/

Here we present an evaluation of the intent-oblivious algorithms against intent-aware measures. This evaluation provides a validation of the diversification techniques with an external measure such as *group coverage* [90] and *S-recall* [144].

Intents of a query set $\mathcal{Q}$ is extracted by collecting the classes, subtopics, and topics of each seed node. Since our aim is to evaluate the results based on the coverage of different groups, we only use scenario-3 queries that represent multiple interests.

One measure we are interested in is the *group coverage* as a diversity measure [90]. It computes the number of groups covered by the result set and defined on classes, subtopics, and topics based on the intended level of granularity. However, this measure omits the actual intent of a query, assuming that the intent is given with the classes of the seed nodes.

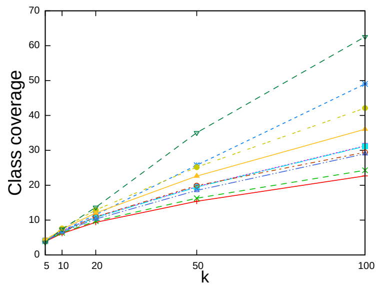
Subtopic recall (*S-recall*) has been defined as the percentage of relevant subtopics covered by the result set [144]. It has also been redefined as *Intent-Coverage* [146], and used in the experiments of [135]. *S-recall* of a result set S based on the set of intents of the query I is computed with

$$S\text{-}recall(S, I) = \frac{1}{|I|} \sum_{i \in I} B_i(S), \tag{6.18}$$

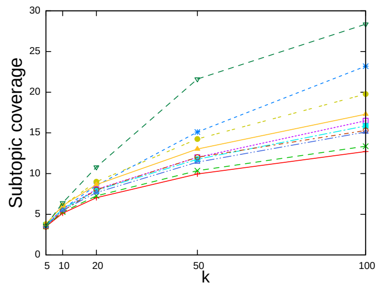
where $B_i(S)$ is a binary variable indicating whether intent i is found in the results.

We give the results of group coverage and *S-recall* on classes, subtopics, and topics in Figure 6.8. The algorithms GrassHopper and GSparse are not included to the results since they perform worse than `PPR`. The results of `AllRandom` are included to give a comparison between the results of top-k relevant set (`PPR`) and ones chosen randomly.

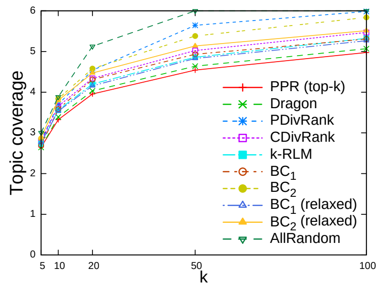
As the group coverage plots show, top-k ranked items of `PPR` do not have the necessary diversity in the result set, hence, the number of groups that are covered by



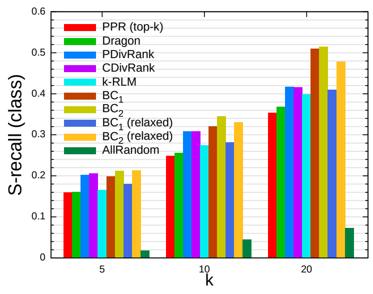
(a) Class coverage



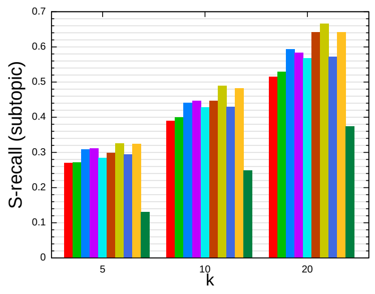
(b) Subtopic coverage



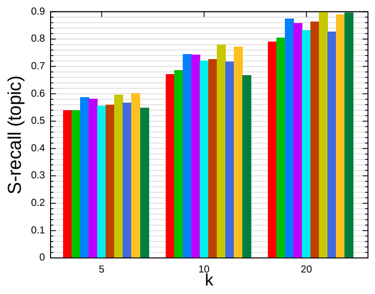
(c) Topic coverage



(d) S-recall on classes



(e) S-recall on subtopics



(f) S-recall on topics

Figure 6.8: Intent-aware results on cit-Patents dataset with scenario-3 queries.

these items are the lowest of all. On the other hand, a randomized method brings irrelevant items from the search space without considering their relevance to the user query. The results of all of the diversification algorithms reside between those two extremes, where the PDivRank covers the most, and Dragon covers the least number of groups.

However, *S-recall* index measures whether a covered group was actually useful or not. Obviously, `AllRandom` scores the lowest as it dismisses the actual query (you may omit the *S-recall* on topics since there are only 6 groups in this granularity level). Among the algorithms, $\texttt{BC}_2$ variants and $\texttt{BC}_1$ score the best while $\texttt{BC}_1$ `(relaxed)` and DivRank variants have similar *S-recall* scores, even though $\texttt{BC}_1$ `(relaxed)` is a much faster algorithm than any DivRank variant (see Figure 6.7).

6.5 Summary

In this work, we address the problem of evaluating result diversification as a bicriteria optimization problem with a relevance measure that ignores diversity, and a diversity measure that ignores relevance to the query. We prove it by running *query-oblivious* algorithms on two commonly used combination of objectives. Next, we argue that a result diversification algorithm should be evaluated under a measure which tightly integrates the query in its value, and presented a new measure called *expanded relevance*. Investigating various quality indices by computing their pairwise correlation, we also show that this new measure has no direct correlation with any other measure. In the second part, we analyze the complexity of the solution that maximizes the *expanded relevance* of the results, and based on the submodularity property of the objective, we present a greedy algorithm called `BestCoverage`, and its efficient relaxation. We experimentally show that the relaxation carries no significant harm to the *expanded relevance* of the solution.

Chapter 7: Conclusions and Future Research Directions

In this dissertation we have investigated result diversification problem in various applications and data types. As the meaning of diversity/novelty differs for different applications and use cases, we provided insightful details, and sometimes a complete recommendation framework (for the paper recommendation application) and presented geometric, index-based, and graph-based solutions. For all the solutions, we consider the overall accuracy of the recommendation, as well as the efficiency of the proposed method. The proposed methods and our findings can be summarized as follows.

7.1 Summaries and our findings

Diverse Browsing on Spatial and Multidimensional Data. In Chapter 2, we investigate the diversification problem in multi-dimensional nearest neighbor search. Because diverse k-nearest neighbor search is conceptually similar to the idea of natural neighbors, we give a definition of *diversity* by making an analogy with the concept of natural neighbors and propose a natural neighbor-based method. Observing the limitations of NatN-based method in higher dimensional spaces, we present a Gabriel graph-based method that scales well with dimensionality. We also introduce an index-based diverse browsing method, which maintains a priority queue with the ranks of the objects depending on both relevancy and diversity, and efficiently prunes non-diverse items and nodes in order to efficiently get the diverse nearest neighbors. To evaluate the diversity of a given result set to a query point, a measure that captures both the relevancy and angular diversity is presented.

We experiment on spatial and multi-dimensional, real and synthetic datasets to observe the efficiency and effectiveness of proposed methods, and compare with index-based techniques found in the literature. We summarize our findings as follows: (1) Geometric approaches are suitable for static data, and index-based diverse browsing is for dynamic databases. (2) Index-based diverse browsing method performed more efficient than k-NN search with distance browsing on R-tree (in terms of the number of disk accesses) and more effective than other methods found in the literature (in terms of MMR). (3) Gabriel graph-based method performed well in high dimensions, which can be investigated more and applied to other research fields where search in high dimensional space is required.

Sentiment Analysis and Opinion Diversification. In Chapter 3, we use a sentiment extraction tool to investigate the influence of factors such as gender, age, education level, the topic at hand, or even the time of the day on sentiments in the context of a large online question answering site. We start our analysis by looking at direct correlations, e.g., we observe more positive sentiments on weekends, very neutral ones in the Science & Mathematics topic, a trend for younger people to express stronger sentiments, or people in military bases to ask the most neutral questions. We then extend this basic analysis by investigating how properties of the (asker, answerer) pair affect the sentiment present in the answer. Among other things, we observe a dependence on the pairing of some inferred attributes estimated by a user's ZIP code. Finally we report results for the task of predicting the attitude that a question will provoke in answers, and discuss the possibility of using demographic features for the opinion diversification task.

Some selected findings of our work are the following: (1) There is a strong dependency on the topic. Topics such as Beauty & Style attract strong and generally positive sentiments, whereas Science & Mathematics attracts answers of low sentimentality. (2) Demographic factors suggest a strong influence in our data, with women generally expressing stronger, more positive sentiments than men, young people being more positive than older people, and people from predominantly black neighborhoods expressing relatively more neutral sentiments. We also observe a trend for more educated people to give less sentimental answers. (3) Sentiments show temporal variation. At a monthly level, the most positive sentiments are observed both during the summer and December. At a daily level, the most positive sentiments are expressed on Saturday and Sunday. At an hourly level, the attitude is at its lowest at around five in the morning. (4) People have stronger tendency to give neutral answers as they gain more experience in the online world. (5) Best answers differ significantly from other answers in terms of expressed sentiments with more neutral answers being preferred in Business & Finance and more positive ones in News & Events.

Graph-based Paper Recommendation and Diversity. Chapter 4 focuses on the paper recommendation problem on academic networks, and how the mentioned ambiguity and redundancy issues interacts with the users' preferences and satisfiability of the results. To observe those effects, we built a paper recommendation service called the**advisor** which recommends new papers to researchers using only the reference-citation relationships between academic papers.

For the efficiency of the recommendation algorithm, we propose compression and bandwidth reduction techniques to reduce the memory usage and hence, the bandwidth required to bring the matrix from the memory at each iteration. We also used matrix ordering techniques to reduce the number cache misses.

Lastly, we address the diversification of paper recommendations of the**advisor** service. While giving a survey of diversity methods designed specifically for random walk-based rankings, we adapted those methods to our direction-aware problem, and proposed some new ones based on vertex selection and query refinement.

For the accuracy of the recommendation algorithms, direction-aware variants outperform the existing algorithms when the objective is to find either traditional or recent papers. Experimental results on efficiency show the modifications greatly help to reduce the query execution time. We tested the algorithms with different K values to find the best configuration. The fastest algorithm is COO-Half where $K = 8$ and the diagonal blocks are ordered with AMD. The average query response time for this configuration, which is being used in the**advisor**, is 1.51 seconds. Compared with the execution time of CRS-Full with the original ordering, which is 4.55 seconds, we obtain 3 times improvement. Finally, our experiments with various relevancy and diversity measures show that the proposed γ-RLM algorithm can be preferred for both its efficiency and effectiveness.

Exploratory Search and Result Diversification. In Chapter 5 we identify the properties that an academic recommendation service should provide to its users as exploration of the data. We argue that the existing academic services lack some of the mentioned properties. In our paper recommendation service, exploration is achieved by three techniques leading to an overall description of the area (through diversification), guiding technique to reinforce interest (through recommendation feedback) and manual discovery (through visualization).

We exemplify the effects of reranking the recommendations with a diversification method on a real world query, and show that the recommendations with k-RLM diversification improve the set of recommendations by eliminating redundant results and by covering other fields of interest. For the relevance feedback, the results show that the feedback mechanism, especially positive feedback, allows to speedup the process of searching for specific references.

Graph Diversity and Common Pitfalls in its Evaluation. In Chapter 6, we address the problem of evaluating result diversification as a bicriteria optimization problem. We prove it by running *query-oblivious* algorithms on two commonly used combination of objectives. Next, we argue that a result diversification algorithm should be evaluated under a measure which tightly integrates the query in its value, and presented a new measure called *expanded relevance*.

The main and most important conclusion of this work is that result diversification should not be evaluated as a bicriteria optimization problem with a relevance measure that ignores diversity and a diversity measure that ignores relevancy.

Investigating various quality indices by computing their pairwise correlation, we show that the proposed measure has no direct correlation with any other measure. We also analyze the complexity of the solution that maximizes the *expanded relevance* of the results, and based on the submodularity property of the objective, we present a greedy algorithm called `BestCoverage`, and its efficient relaxation. We experimentally show that the relaxation carries no significant harm to the *expanded relevance* of the solution.

7.2 Open problems

In this dissertation on various diversification problems, we have introduced novel techniques, efficiency improvements, and new measures to the literature; however, these new developments have also established new research problems and application areas. We would like to discuss some of the open problems for future research.

Diverse Browsing on Spatial and Multidimensional Data. Since there are numerous application areas of diverse k-nearest neighbor search, the proposed geometric and index-based methods can be adapted to work with different types of data and distance metrics. Especially databases with an index built on top of a multidimensional feature column can provide the diverse browsing feature to its users since R-tree and R*-tree are the two most common spatial partitioning methods used in popular DBMSs.

Sentiment Analysis and Opinion Diversification. Given that we observed significant differences concerning sentiments between different demographic groups, this could be used to normalize individual sentiments to obtain a better idea about deviations from the expected behavior. For example, a “pretty good” by an older person might be equivalent to an “absolutely amazing” by a teenager. The availability of a topical classification makes it possible to differentiate sentiments attached to a particular entity according to the context, and to obtain a more faceted representation of the opinions about an entity.

One interesting problem is to apply sentiment prediction techniques in other domains. For example, it would be interesting to predict the sentiments of comments left in response to a news article. Potentially, such techniques could even be used by blog writers to improve user engagement by providing them with indications about how to make their posts more controversial. Similarly, we deem it interesting to generalize our findings concerning the sentiments present in best answers to the more general problem of evaluating content quality. The presence/absence of sentiments in the text of a news article might be an indication of its quality.

Graph-based Paper Recommendation and Diversity. In our recommendation system, the citation graph is built and assumed to be unweighted. However, it is possible and arguable more beneficial if weights are assigned to the edges based on how many times a paper cites another. This way, the citation/reference relationship between two papers can be boosted if one paper repeatedly mentions the other, which generally corresponds to the importance of the reference itself. In addition, a paper recommender service would provide more relevant and up-to-date results with more, recent, and high-quality bibliographic data. Obtaining new sources of bibliographic entries and including papers from new research fields will improve the quality of the paper recommendations for interdisciplinary studies. It is also possible to conduct an intensive user study to obtain a real-world evaluation of the system.

For the efficiency part, developing new ideas to further reduce the query response time of the service will be one of the tasks we are interested in as long as it is running. Since it is very hard to obtain linear speedups with shared memory parallelization in SpMxV operations, to maximize the throughput we chose to use one processor per query. However, we believe that such parallelism can still be effective for the**advisor** especially when the number concurrent requests is less than the number of processors allocated in the cluster.

Diversification of the paper recommendations was obtained using a graph-based method. As there are significant amount of research on intent-aware diversification, it would be possible to understand the intent of the query or the user by analyzing the textual content of the provided references. The next reasonable step for a better paper recommendation, hence, would be to obtain full-text sources of the indexed documents, and provide diversity based on textual similarities.

Exploratory Search and Result Diversification. Although the interplay between diversification, feedback, and visualization has not been explored in the context of paper recommendation, they provide a mechanism similar to drill-down/roll-up operations in data warehousing context, which are shown to be very important features for users to adjust the level of detail. An analysis on the logs and interactions recorded by the service would provide information on how those components are employed by the users of the system regarding their objective to discover new papers in the literature.

Graph Diversity and Common Pitfalls in its Evaluation. The proposed *exprel*$_\ell$ measure and the `BestCoverage` algorithm that finds a near-optimal solution are shown to be useful in the context of graph-based result diversification where the intents of the query or indexed data is unknown. It is possible to investigate the behavior of the *exprel*$_\ell$ measure on applications where categories or intents are explicitly provided, e.g., social networks with ground-truth communities.

Bibliography

[1] Ahmed Abbasi, Hsinchun Chen, and Arab Salem. Sentiment analysis in multiple languages: feature selection for opinion classification in Web forums. *ACM Trans. Inf. Syst.*, 26:12:1–12:34, 2008.

[2] Pankaj K. Agarwal, Lars Arge, and Ke Yi. I/O-efficient construction of constrained Delaunay triangulations. In *ESA'05*, pages 355–366, 2005.

[3] R. C. Agarwal, F. G. Gustavson, and M. Zubair. A high performance algorithm using pre-processing for the sparse matrix-vector multiplication. In *Proc. ACM/IEEE Supercomputing*, pages 32–41, 1992.

[4] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proc. ACM Int'l Conf. Web Search and Data Mining*, pages 5–14, 2009.

[5] Kadir Akbudak, Enver Kayaaslan, and Cevdet Aykanat. Hypergraph-partitioning-based models and methods for exploiting cache locality in sparse-matrix vector multiplication. *CoRR*, abs/1202.3856, 2012.

[6] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal. Appl.*, 17(4):886–905, 1996.

[7] Sinan Aral and Marshall W. Van Alstyne. The diversity-bandwidth tradeoff. *American Journal of Sociology*, 117(1), 2011.

[8] Franz Aurenhammer, Technische Universitt Graz, Rolf Klein, Fernuniversitt Hagen, and Praktische Informatik Vi. Voronoi diagrams. In *Handbook of Computational Geometry*, pages 201–290, 2000.

[9] Xue Bai. Predicting consumer sentiments from online text. *Decis. Support Syst.*, 50:732–742, 2011.

[10] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[11] Philip Beineke, Trevor Hastie, Christopher Manning, and Shivakumar Vaithyanathan. Exploring sentiment summarization. In *Proc. AAAI Spring Symp. Exploring Attitude and Affect in Text: Theories and Applications*, pages 1–4, 2004.

[12] Johan Bollen, Huina Mao, and Alberto Pepe. Determining the public mood state by analysis of microblogging posts. In *Proc. Alife XII Conf.*, pages 667–668, 2010.

[13] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *J. Comput. Sci*, 2:1–8, 2011.

[14] Johan Bollen, Marko A. Rodriguez, and Herbert Van de Sompel. Journal status. *Scientometrics*, 69(3):669–687, 2006.

[15] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. Int'l Conf. World Wide Web*, pages 107–117, 1998.

[16] Aydin Buluç, Samuel Williams, Leonid Oliker, and James Demmel. Reduced-bandwidth multithreaded algorithms for sparse matrix-vector multiplication. In *Proc. IEEE Int'l Parallel & Distributed Processing Symp.*, pages 721–733, 2011.

[17] Sara Ines Calderon. Facebook shares new data on relationship status and sentiment, 2010. http://www.insidefacebook.com/2010/02/15/dr-facebook-is-in-people-in-relationships-are-happiest/.

[18] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 335–336, 1998.

[19] Ben Carterette. An analysis of NP-completeness in novelty and diversity ranking. In *Proc. Int'l Conf. Theory of Information Retrieval: Advances in Information Retrieval Theory*, pages 200–211, 2009.

[20] Ben Carterette. An analysis of NP-completeness in novelty and diversity ranking. *Information Retrieval*, 14(1):89–106, 2011.

[21] Ümit V. Çatalyürek and Cevdet Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE T. Parall. Distr.*, 10:673–693, 1999.

[22] Ümit V. Çatalyürek and Cevdet Aykanat. *PaToH: A Multilevel Hypergraph Partitioning Tool, Version 3.0.* Bilkent University, Computer Eng., Ankara, Turkey. Available at http://bmi.osu.edu/~umit/software.htm, 1999.

[23] Ümit V. Çatalyürek and Cevdet Aykanat. A fine-grain hypergraph model for 2D decomposition of sparse matrices. In *Proc. IEEE Int'l Parallel & Distributed Processing Symp.*, 2001.

[24] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proc. ACM Conf. Information and Knowledge Management*, pages 621–630, 2009.

[25] Harr Chen and David R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 429–436, 2006.

[26] Hung-Hsuan Chen, Liang Gou, Xiaolong Zhang, and Clyde Lee Giles. CollabSeer: a search engine for collaboration discovery. In *Proc. ACM/IEEE Joint Conf. Digital Libraries*, pages 231–240, 2011.

[27] Xueqi Cheng, Pan Du, Jiafeng Guo, Xiaofei Zhu, and Yixin Chen. Ranking on data manifold with sink points. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):177–191, Jan 2013.

[28] Kyle C. Chipman and Ambuj K. Singh. Predicting genetic interactions with random walks on biological networks. *BMC Bioinformatics*, 17(10), 2009.

[29] Charles L.A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. Overview of the TREC 2011 Web Track. In *Proc. Text Retrieval Conference (TREC)*, 2011.

[30] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 659–666, 2008.

[31] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proc. ACM national conference*, pages 157–172, 1969.

[32] Sanjiv R. Das and Mike Y. Chen. Yahoo! for Amazon: sentiment extraction from small talk on the Web. *Manage. Sci.*, 53:1375–1388, 2007.

[33] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proc. Int'l Conf. World Wide Web*, pages 519–528, 2003.

[34] Ann Devitt and Khurshid Ahmad. Sentiment analysis in financial news: a cohesion-based approach. In *Proc. 45th Annual Meeting of the Assoc. for Computational Linguistics*, pages 984–991, 2007.

[35] Marina Drosou and Evaggelia Pitoura. Diversity over continuous data. *IEEE Data Eng. Bull.*, 32(4):49–56, 2009.

[36] Marina Drosou and Evaggelia Pitoura. Search result diversification. *SIGMOD Rec.*, 39(1):41–47, September 2010.

[37] Pan Du, Jiafeng Guo, Jin Zhang, and Xueqi Cheng. Manifold ranking with sink points for update summarization. In *Proc. ACM Conf. Information and Knowledge Management*, pages 1757–1760, 2010.

[38] Rex A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. In *SCG'89*, pages 326–333, 1989.

[39] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. 43rd Annual Meeting on Assoc. for Computational Linguistics*, ACL'05, pages 363–370, 2005.

[40] Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Stat.*, 29(5):1189–1232, 2001.

[41] K. Ruben Gabriel and Robert R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969.

[42] Shima Gerani, Mark J. Carman, and Fabio Crestani. Investigating learning approaches for blog post opinion retrieval. In *Proc. European Conf. Information Retrieval*, pages 313–324, 2009.

[43] Shima Gerani, Mark James Carman, and Fabio Crestani. Proximity-based opinion retrieval. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 403–410, 2010.

[44] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *Proc. ACM Conf. Digital Libraries*, pages 89–98, 1998.

[45] Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. In *Proc. Int'l Conf. Weblogs and Social Media*, 2007.

[46] Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *Proc. Int'l Conf. World Wide Web*, pages 381–390, 2009.

[47] Benjamin Good, Joseph Tennis, and Mark Wilkinson. Social tagging in the life sciences: characterizing a new metadata resource for bioinformatics. *BMC Bioinformatics*, 10(1):313, 2009.

[48] Marco Gori and Augusto Pucci. Research paper recommender systems: A random-walk based approach. In *Proc. IEEE/WIC/ACM Web Intelligence*, pages 778–781, 2006.

[49] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971.

[50] Michelle L. Gregory, Nancy Chinchor, Paul Whitney, Richard Carter, Elizabeth Hetzler, and Alan Turner. User-directed sentiment analysis: visualizing the affective content of documents. In *Proc. Workshop on Sentiment and Subjectivity in Text*, pages 23–30, 2006.

[51] Martin Halvey, P. Punitha, David Hannah, Robert Villa, Frank Hopfgartner, Anuj Goyal, and Joemon M. Jose. Diversity, assortment, dissimilarity, variety: A study of diversity measures using low level features for video retrieval. In *Proc. European Conf. Information Retrieval*, pages 126–137, 2009.

[52] Jayant R. Haritsa. The KNDN problem: A quest for unity in diversity. *IEEE Data Eng. Bull.*, 32(4):15–22, 2009.

[53] Rakebul Hasan, Katharina Siorpaes, Reto Krummenacher, and Fabian Flöck. Towards a knowledge diversity model. In *Proc. Int'l Workshop on Knowledge Diversity on the Web*, 2011.

[54] Taher H. Haveliwala. Topic-sensitive PageRank. In *Proc. Int'l Conf. World Wide Web*, pages 517–526, 2002.

[55] Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and Lee Giles. Context-aware citation recommendation. In *Proc. Int'l Conf. World Wide Web*, pages 421–430, 2010.

[56] Gísli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[57] Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard problems.* PWS publishing company, 1997.

[58] Martin Isenburg, Yuanxin Liu, Jonathan Shewchuk, and Jack Snoeyink. Streaming computation of Delaunay triangulations. In *Proc. ACM Int'l SIGGRAPH*, pages 1049–1056, 2006.

[59] Anoop Jain, Parag Sarda, and Jayant R. Haritsa. Providing diversity in k-nearest neighbor query results. In *Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining*, pages 404–413, 2003.

[60] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proc. Int'l Conf. World Wide Web*, pages 271–279, 2003.

[61] Sepandar D. Kamvar and Jonathan Harris. We feel fine and searching the emotional web. In *Proc. ACM Int'l Conf. Web Search and Data Mining*, pages 117–126, 2011.

[62] U Kang and Christos Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *Proc. IEEE Int'l Conf. Data Mining*, pages 300–309, 2011.

[63] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25, 1963.

[64] Madian Khabsa, Pucktada Treeratpituk, and C. Lee Giles. AckSeer: a repository and search engine for automatically extracted acknowledgments from digital libraries. In *Proc. ACM/IEEE Joint Conf. Digital Libraries*, pages 185–194, 2012.

[65] Heung-Nam Kim and Abdulmotaleb El Saddik. Personalized PageRank vectors for tag recommendations: inside FolkRank. In *Proc. ACM Recommender Systems*, pages 45–52, 2011.

[66] Soo-Min Kim and Eduard H. Hovy. Crystal: analyzing predictive opinions on the Web. In *Proc. 2007 Joint Conf. Empirical Methods in Natural Language and Computational Natural Language Learning*, pages 1056–1064, 2006.

[67] Onur Küçüktunç, B. Barla Cambazoğlu, Ingmar Weber, and Hakan Ferhatosmanoğlu. A large-scale sentiment analysis for Yahoo! Answers. In *Proc. ACM Int'l Conf. Web Search and Data Mining*, 2012.

[68] Onur Küçüktunç and Hakan Ferhatosmanoğlu. Diverse browsing for spatial data. Technical Report OSU-CISRC-1/11-TR02, OSU CSE, Feb 2011.

[69] Onur Küçüktunç and Hakan Ferhatosmanoğlu. λ-diverse nearest neighbors browsing for multidimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):481–493, Mar 2013.

[70] Onur Küçüktunç, Kamer Kaya, Erik Saule, and Ümit V. Çatalyürek. Fast recommendation on bibliographic networks. In *Proc. IEEE/ACM Int'l Conf. Advances in Social Networks Analysis and Mining*, 2012.

[71] Onur Küçüktunç, Kamer Kaya, Erik Saule, and Ümit V. Çatalyürek. Fast recommendation on bibliographic networks with sparse-matrix ordering and partitioning. *Social Network Analysis and Mining (SNAM)*, 2013. (to appear).

[72] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Direction awareness in citation recommendation. In *Proc. Int'l Workshop on Ranking in Databases (DBRank'12) in conjunction with VLDB'12*, 2012.

[73] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Diversifying citation recommendations. Technical Report arXiv:1209.5809, ArXiv, Sep 2012.

[74] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Diversifying citation recommendations. Technical Report arXiv:1209.5809, ArXiv, Sep 2012.

[75] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Recommendation on academic networks using direction aware citation analysis. Technical Report arXiv:1205.1143, ArXiv, Apr 2012.

[76] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Diversified recommendation on graphs: Pitfalls, measures, and algorithms. In *Proc. Int'l Conf. World Wide Web*, 2013.

[77] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Diversifying citation recommendations. *ACM Trans. Information Systems and Technology*, 2013. (under review).

[78] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Result diversification in automatic citation recommendation. In *iConference Workshop on Computational Scientometrics: Theory and Applications*, 2013.

[79] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. TheAdvisor: A webservice for academic recommendation. In *Proc. ACM/IEEE Joint Conf. Digital Libraries*, 2013. (poster).

[80] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Towards a personalized, scalable, and exploratory academic recommendation service. In *Proc. IEEE/ACM Int'l Conf. Advances in Social Networks Analysis and Mining*, Aug 2013.

[81] Ni Lao and William Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81:53–67, 2010.

[82] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *Computer*, 32:67–71, 1999.

[83] Hugo Ledoux and Christopher Gold. An efficient natural neighbour interpolation algorithm for geoscientific modelling. In *SDH'04*, pages 23–25, 2004.

[84] Thomas Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* Willey–Teubner, 1990.

[85] Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald. Sentiment summarization: evaluating and learning user preferences. In *Proc. 12th Conf. European Chapter of the Assoc. for Computational Linguistics*, pages 514–522, 2009.

[86] Michael Ley. DBLP - some lessons learned. *Proc. VLDB Endowment*, 2(2):1493–1500, 2009.

[87] Huajing Li, Isaac Councill, Wang-Chien Lee, and C. Lee Giles. CiteSeerx: an architecture and web service design for an academic document search engine. In *Proc. Int'l Conf. World Wide Web*, pages 883–884, 2006.

[88] Jiang Li and Peter Willett. Articlerank: a PageRank-based alternative to numbers of citations for analyzing citation networks. *Proc. ASLIB*, 61(6):605–618, 2009.

[89] Rong-Hua Li and Jeffrey Xu Yu. Scalable diversified ranking on large graphs. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1, 2012. preprint.

[90] Rong-Hua Li and J.X. Yu. Scalable diversified ranking on large graphs. In *Proc. IEEE Int'l Conf. Data Mining*, pages 1152–1157, 2011.

[91] X.-Y. Li, P.-J. Wan, Y. Wang, and O. Frieder. Sparse power efficient topology for wireless networks. In *Proc. Hawaii Int'l Conf. System Sciences*, page 296.2, 2002.

[92] Yicong Liang, Qing Li, and Tieyun Qian. Finding relevant papers based on citation relations. In *Proc. Int'l Conf. Web-Age Information Management*, pages 403–414, 2011.

[93] David Liben-Nowell and Jon M. Kleinberg. The link-prediction problem for social networks. *JASIST*, 58(7):1019–1031, 2007.

[94] Bin Liu and H. V. Jagadish. Using trees to depict a forest. *Proc. VLDB Endowment*, 2(1):133–144, 2009.

[95] Bing Liu. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition.* CRC Press, Taylor and Francis Group, 2010.

[96] David B. Lomet, editor. *IEEE Data Eng. Bull. (Special Issue on Result Diversity)*, volume 32(4). IEEE Computer Society, 2009.

[97] Wangzhong Lu, J. Janssen, E. Milios, N. Japkowicz, and Yongzheng Zhang. Node similarity in the citation graph. *Knowl. Inf. Syst.*, 11:105–129, 2006.

[98] Nan Ma, Jiancheng Guan, and Yi Zhao. Bringing PageRank to the citation analysis. *Inf. Process. Manage.*, 44:800–810, 2008.

[99] David W. Matula and Robert R. Sokal. Properties of Gabriel graphs relevant to geographical variation research and the clustering of points on the plane. *Geographical Analysis*, 12:205–222, 1980.

[100] Sean M. McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, and John Riedl. On the recommending of citations for research papers. In *Proc. ACM Computer Supported Cooperative Work*, pages 116–125, 2002.

[101] Qiaozhu Mei, Jian Guo, and Dragomir Radev. DivRank: the interplay of prestige and diversity in information networks. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 1009–1018, 2010.

[102] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294, 1978.

[103] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. TR 1999-66, Stanford InfoLab, 1999.

[104] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 653–658, 2004.

[105] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2:1–135, 2008.

[106] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proc. Conf. Empirical Methods in Natural Language Processing*, pages 79–86, 2002.

[107] R. Pemantle. Vertex-reinforced random walk. *Probab. Theory Related Fields*, 92:117–136, 1992.

[108] J. C. Pichel, D. B. Heras, J. C. Cabaleiro, and F. F. Rivera. Performance optimization of irregular codes based on the combination of reordering and blocking techniques. *Parallel Comput.*, 31(8+9):858–876, 2005.

[109] J. C. Pichel, D. B. Heras, J. C. Cabaleiro, and F. F. Rivera. Increasing data reuse of sparse algebra codes on simultaneous multithreading architectures. *Concurr. Comput.: Pract. Exper.*, 21(15):1838–1856, 2009.

[110] Ali Pinar and Michael T. Heath. Improving performance of sparse matrix-vector multiplication. In *Proc. ACM/IEEE Supercomputing*, 1999.

[111] Filip Radlinski and Susan Dumais. Improving personalized web search using result diversification. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 691–692, 2006.

[112] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pages 71–79, 1995.

[113] Gerard Salton. Associative document retrieval techniques using bibliographic information. *J. ACM*, 10:440–457, 1963.

[114] Jonathan R. Shewchuk. *Triangle: engineering a 2D quality mesh generator and Delaunay triangulator*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer Berlin Heidelberg, May 1996.

[115] R. Sibson. A brief description of natural neighbor interpolation. *Interpolating Multivariate Data*, pages 21–36, 1981.

[116] Henry Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *J. Am. Soc. Inf. Sci.*, 24(4):265–269, 1973.

[117] Trevor Strohman, W. Bruce Croft, and David Jensen. Recommending citations for academic papers. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 705–706, 2007.

[118] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: extraction and mining of academic social networks. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 990–998, 2008.

[119] Yufei Tao. Diversity in skylines. *IEEE Data Eng. Bull.*, 32(4):65–72, 2009.

[120] O. Temam and W. Jalby. Characterizing the behavior of sparse algorithms on caches. In *Proc. ACM/IEEE Supercomputing*, pages 578–587, 1992.

[121] Mike Thelwall. Emotion homophily in social network site messages. *First Monday*, 15(4-5), 2010.

[122] Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. Sentiment in Twitter events. *J. Am. Soc. Inf. Sci. Technol.*, 62:406–418, 2011.

[123] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment in short strength detection informal text. *J. Am. Soc. Inf. Sci. Technol.*, 61:2544–2558, 2010.

[124] Mike Thelwall, David Wilkinson, and Sukhvinder Uppal. Data mining emotion in social network communication: gender differences in MySpace. *J. Am. Soc. Inf. Sci. Technol.*, 61(1):190–199, 2010.

[125] Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: determining support or opposition from congressional floor-debate transcripts. In *Proc. Conf. Empirical Methods in Natural Language Processing*, pages 327–335, 2006.

[126] S. Toledo. Improving the memory-system performance of sparse-matrix vector multiplication. *IBM J.Res.Dev.*, 41(6):711–726, 1997.

[127] Ioannis G. Tollis, Giuseppe Di Battista, Peter Eades, and Roberto Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, July 1998.

[128] Hanghang Tong, Jingrui He, Zhen Wen, Ravi Konuru, and Ching-Yung Lin. Diversified ranking on large graphs: an optimization viewpoint. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 1028–1036, 2011.

[129] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proc. 40th Annual Meeting on Assoc. for Computational Linguistics*, pages 417–424, 2002.

[130] Erik Vee, Utkarsh Srivastava, Jayavel Shanmugasundaram, Prashant Bhat, and Sihem Amer-Yahia. Efficient computation of diverse query results. In *Proc. IEEE Int'l Conf. Data Engineering*, pages 228–236, 2008.

[131] Marcos R. Vieira, Humberto L. Razente, Maria C.N. Barioni, Marios Hadjieleftheriou, Divesh Srivastava, Cartano Traina, and Vasillis J. Tsotras. On query result diversification. In *Proc. IEEE Int'l Conf. Data Engineering*, pages 1163–1174, 2011.

[132] Qiang Wang, Vasileios Megalooikonomou, and Christos Faloutsos. Time series analysis with multiple resolutions. *Information Systems*, 35:56–74, January 2010.

[133] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, pages 440–442, June 1998.

[134] Ingmar Weber and Carlos Castillo. The demographics of web search. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 523–530, 2010.

[135] Michael J. Welch, Junghoo Cho, and Christopher Olston. Search result diversity for informational queries. In *Proc. Int'l Conf. World Wide Web*, pages 237–246, 2011.

[136] James B. White and P. Sadayappan. On improving the performance of sparse matrix-vector multiplication. In *Proc. Int'l Conf. High Performance Computing*, pages 66–71, 1997.

[137] Jerry Ye, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng. Stochastic gradient boosted distributed decision trees. In *Proc. ACM Conf. Information and Knowledge Management*, pages 2061–2064, 2009.

[138] Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques. In *Proc. IEEE Int'l Conf. Data Mining*, pages 427–434, 2003.

[139] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. A unified framework for link recommendation using random walks. In *Proc. IEEE/ACM Int'l Conf. Advances in Social Networks Analysis and Mining*, pages 152–159, 2010.

[140] Cong Yu, Laks Lakshmanan, and Sihem Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *Proc. Int'l Conf. Extending Database Technology*, pages 368–378, 2009.

[141] Cong Yu, Laks Lakshmanan, and Sihem Amer-Yahia. Recommendation diversification using explanations. In *Proc. IEEE Int'l Conf. Data Engineering*, pages 1299–1302, 2009.

[142] A. N. Yzelman and Rob H. Bisseling. Cache-oblivious sparse matrix–vector multiplication by using sparse matrix partitioning methods. *SIAM J. Sci. Comput.*, 31:3128–3154, 2009.

[143] A. N. Yzelman and Rob H. Bisseling. Two-dimensional cache-oblivious sparse matrix-vector multiplication. *Parallel Comput.*, 37:806–819, 2011.

[144] Cheng Xiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 10–17, 2003.

[145] Wei Zhang, Clement Yu, and Weiyi Meng. Opinion retrieval from blogs. In *Proc. ACM Conf. Information and Knowledge Management*, pages 831–840, 2007.

[146] Xiaofei Zhu, Jiafeng Guo, Xueqi Cheng, Pan Du, and Hua-Wei Shen. A unified framework for recommending diverse and relevant queries. In *Proc. Int'l Conf. World Wide Web*, pages 37–46, 2011.

[147] Xiaojin Zhu, Andrew B. Goldberg, Jurgen Van Gael, and David Andrzejewski. Improving diversity in ranking using absorbing random walks. In *Proc. HLT-NAACL*, pages 97–104, 2007.

[148] Cai-Nicolas Ziegler and Georg Lausen. Making product recommendations more diverse. *IEEE Data Eng. Bull.*, 32(4):23–32, 2009.

[149] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proc. Int'l Conf. World Wide Web*, pages 22–32, 2005.

[150] Guido Zuccon, Leif Azzopardi, Dell Zhang, and Jun Wang. Top-k retrieval using facility location analysis. In *Proc. European Conf. Information Retrieval*, pages 305–316, 2012.