# PRIVACY PRESERVING CONTENT ANALYSIS, INDEXING AND RETRIEVAL FOR SOCIAL SEARCH APPLICATIONS

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades

DOKTOR DER NATURWISSENSCHAFTEN

**Dr. rer. nat.**

genehmigte Dissertation
von

**M.Sc. Sergej Zerr**

geboren am 27. April 1977, in Karaganda, ehem. UdSSR

2015

**Referent: Prof. Dr. techn. Wolfgang Nejdl**
**Korreferent: Prof. Dr. Heribert Vollmer**
**Korreferent: Prof. Dr. Wolf-Tilo Balke**
**Tag der Promotion: 9 Dezember 2015**

# ABSTRACT

Most of the modern Web platforms have recognized the demand for resource sharing among their users and provide powerful tools for upload and sharing of user-generated content and personal documents. The need for effective information sharing and search within the growing amount of documents pushed forward further development of search infrastructures for enterprise data management systems. Applications for creation of the real, as well as, of the virtual communities are developed and connect people that are working on similar topics or share similar interests. The power of such tools and the easiness of their use, often make users careless about possible threats with respects to confidentiality of the information that is disclosed during the sharing. Privacy-preserving document exchange among collaboration groups in social web and across enterprises requires efficient techniques for sharing and search of access-controlled information in largely untrusted environments. Thus, privacy preservation mechanisms need to be, by design, an inseparable part in sharing systems and tools. In this thesis we propose solutions for confident information storage, indexing and sharing. In particular, we address: (1) Efficient privacy-preserving indexing and retrieval of shared documents; (2) Scalable content diversity analysis for large document corpora suitable for supporting efficient access control for an inverted index; and (3) Privacy-oriented content analysis of shared content to automatically distinguish public from potentially sensitive content.

First, we introduce *ZERBER+R* - a facility, which enable indexing sensitive documents in a way that the information leakage from the index is bounded by a tunable parameter. This facility is able to efficiently and securely locate and select *top-k* relevant documents the user is allowed to access without leaking information about contained documents and their term distribution statistics. To provide tunable resistance to statistical attacks, *ZERBER+R* employs a novel term merging scheme that has minimal impact on the index lookup costs and relies on *largely untrusted* index servers. Our experiments have shown that *ZERBER+R* makes economical use of network bandwidth, requires minimal key management, and answers queries almost as fast as an ordinary inverted index. Second, we develop two novel, efficient algorithms for content analysis of large document collections to estimate their topical diversity with probabilistic guarantees. Among many application scenarios, this measure is potentially useful to improve access control properties by supporting the index partitioning. Whilst state-of-the-art diversity measurement algorithms require quadratic computational time with respect to the dataset size, our methods, called TrackDJ and SampleDJ, can be executed in linear time and constant time, respectively. Finally, we present a framework for privacy degree estimation and corresponding classification of photographs as they are shared and published by the users of social web applications. The system supports users in selecting adequate privacy settings. To this end we trained classification models on a large-scale dataset obtained in a social annotation game. We analysed the usefulness of textual and visual information in the context of privacy-oriented image classification. In addition, we developed privacy-oriented search and privacy-based search result diversification techniques. Our classification and ranking experiments have shown excellent performance for textual, and visual information and their combinations.

The results presented in this thesis can be directly applied in various privacy-aware data sharing scenarios and Web applications such as Flickr, Facebook or Twitter and due to high efficiency the developed algorithms can be integrated in large public and enterprise search engines. Additionally, our theoretical findings open a number of research directions such as studying the efficiency of secured indexes, intelligent index partitioning according to access rights, and sensitivity degree estimation of personal text and multimedia objects like videos or emails.

**Keywords:** *privacy preserving content indexing, scalable content diversity analysis, privacy oriented content analysis*

# ZUSAMMENFASSUNG

Das heutige Web bietet Benutzern eine Vielzahl von Applikationen zu Veröffentlichung und zum Austausch von Bildern, Texten, und Informationen. Dies resultierte in der Entwicklung leistungsstarker Werkzeuge zur Veröffentlichung und gemeinsamen Nutzung von benutzergenerierten Inhalten und persönlichen Dokumenten. Um effektiven Informationsaustausch und Suche für die stark anwachsende Datenmenge zu gewährleisten, wurden Suchinfrastrukturen und Datenmanagementsysteme sowohl in dem Bereich Industrie, als auch im Web-Umfeld entwickelt. Zusätzlich verbinden Applikationen für reale, oder virtuelle Arbeitsgemeinschaften Gruppen von gleichgesinnten Nutzern auf der ganzen Welt. Die Leistungsfähigkeit solcher Werkzeuge und die Einfachheit in deren Anwendung, führen jedoch häufig zu leichtfertigem Umgang mit vertraulichen Informationen. Dies macht eine Berücksichtigung von Datenschutzmechanismen bereits in der Systemdesignphase notwendig.

In dieser Arbeit beschreiben wir Lösungen zum Schutz der Privatsphäre der Benutzer in gemeinschaftlich genutzten Systemen. Insbesondere schlagen wir Lösungen für sichere Informationsspeicherung, Indizierung und den Austausch von Dokumenten vor. Dabei fokussieren wir uns auf: (1) Effiziente und sichere Indizierung und Suche auf gemeinsam verwendeten Dokumenten, (2) Skalierbare Analyse der inhaltlichen Vielfalt für große Dokumentenkorpora, sowie (3) Datenschutzorientierte Inhaltsanalyse von gemeinsam genutzten Inhalten mit automatischen Methoden zur Unterscheidung zwischen öffentlichen und potenziell vertraulichen Inhalten.

Zunächst stellen wir *ZERBER+R* vor - ein System, das die Indizierung vertraulicher Dokumente ermöglicht und dabei den Informationsverlust in dem Index je nach Parametereinstellung unterschiedlich stark begrenzt und verhindert. Unser System ermöglicht effiziente und sichere Suchvorgänge, sowie die Auswahl der relevantesten Dokumente unter der Berücksichtigung der Zugriffsrechte für den Benutzer, ohne dass dabei Informationen über diese Dokumente für potentielle Angreifer sichtbar werden. Des Weiteren entwickelten wir effiziente Algorithmen zur Inhaltsanalyse von großen Dokumentensammlungen um ihre Themenvielfalt abschätzen zu können. Dies kann, unter anderem, zu Partitionierung einer Indexdatei hinsichtlich der Zugangsberechtigungen einzelner Benutzergruppen verwendet werden um die Leistungsfähigkeit des Systems weiter zu steigern. Schließlich präsentieren wir ein Framework, das den Benutzer automatisch bei der Auswahl der adäquaten Datenschutzeinstellungen für seine Dokumente, und insbesondere Bilder unterstützt. Dabei untersuchen wir die Verwendbarkeit von ausgewählten Text- und Bildinformationen zur Bildklassifizierung im Kontext des Schutzes der Privatsphäre. Unsere Experimente zeigen, dass unsere Modelle, unter Benutzung sowohl textueller als auch visueller Merkmale, qualitativ hochwertige Klassifikationsergebnisse liefern.

Die in dieser Arbeit entwickelten Methoden und effizienten Algorithmen ermöglichen sowohl die direkte Anwendung in verschiedene datenschutzrelevante Nutzungsszenarien und Web-Anwendungen wie Flickr, Facebook oder Twitter, als auch die Integration in hochskalierbare öffentliche und industrielle Suchsysteme. Darüber hinaus eröffnen unsere theoretischen Erkenntnisse eine Reihe von möglichen Forschungsrichtungen in den Bereichen Indexeffizienz, Zugriffsrechteverwaltung, und automatische Klassifikation von persönlichen Daten.

**Schlagwörter:** *Datenschutz bei Indizierung und Suche, Skalierbare Analyse der Themenvielfalt, Automatische Unterstützung bei Datenschutzeinstellungen*

# ACKNOWLEDGMENTS

I would like to acknowledge everyone who has supported me throughout my doctoral studies over the last years. First of all, I would first like to acknowledge my advisor and doctor father Prof. Dr. Wolfgang Nejdl, for shaping me as a researcher and for giving me the exciting opportunity to work in the excellent research environment together with the top-scientists in the area of Web Science at the L3S Research Center. I also thank Prof. Dr. Wolf-Tilo Balke, Prof. Dr. Heribert Vollmer and Prof. Dr.-Ing. Christian Müller-Schloer for agreeing to consider and evaluate my PhD thesis. A special thank is to Ms Ance Vais for the organizational support during the course of the promotion. I truly appreciate all of their time and assistance.

I would like to thank Prof. Dr. Jŕgen Biermann and Prof. Dr. Roland Schwänzl for opening me the entrance into the exiting world of Computer Science and Cryptography during my Bachelor and Master studies and equipped me with the fundamental concepts and knowledge, required for starting this thesis. I am very grateful to my mentors at L3S Research Center, guiding me especially at the beginning of my PhD studies: Dr. Daniel Olmedilla and Prof. Marianne Winslett who supported my initial steps as a researcher and Dr. Claudia Niederée, who provided invaluable guidance in the area of scientific project management. A very special thanks is due to my colleagues and friends, Dr. Elena Demidova and Dr. Stefan Siersdorfer for the fruitful collaborations that lead to excellent scientific achievements.

Additional gratitude is offered to my L3S colleagues for many fruitful discussions as well as to the students I was honored to have supervised during my time at the L3S Research Center. I would also like to thank my family for their encouragement, continuous support throughout my education and their enthusiasm as I neared my goal.

# FOREWORD

The methods and algorithms presented in this thesis have been published at various conferences, as follows:

Chapter 3 addresses the problem of privacy preserving content indexing and retrieval and describes the contributions included in:

- Sergej Zerr, Elena Demidova, Daniel Olmedilla, Wolfgang Nejdl, Marianne Winslett, Soumyadeb Mitra. *Zerber: r-confidential indexing for distributed documents*. In Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, EDBT'08, pages 287-298, New York, NY, USA, 2008. ACM. [ZDO+08]

- Sergej Zerr, Daniel Olmedilla, Wolfgang Nejdl, Wolf Siberski: *Zerber+R: top-k retrieval from a confidential index*. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT'09, pages 439-449, New York, NY, USA, 2009. ACM. [ZONS09]

Chapter 4 focuses on the scalable analysis of content diversity and builds upon the work published in:

- Fan Deng, Stefan Siersdorfer, Sergej Zerr. *Efficient Jaccard-based diversity analysis of large document collections*. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM'12, pages 1402-1411, New York, NY, USA, 2012. ACM. [DSZ12]

Chapter 5 addresses the problem of privacy-oriented image content analysis and includes the contributions published in:

- Sergej Zerr, Stefan Siersdorfer, Jonathon Hare, Elena Demidova. *Privacy-aware image classification and search*. In Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'12, pages 35-44, New York, NY, USA, 2012. ACM. [ZSHD12]

- Sergej Zerr, Stefan Siersdorfer, Jonathon Hare. *PicAlert!: A system for privacy-aware image classification and retrieval.* In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM'12, pages 2710-2712, New York, NY, USA, 2012. ACM. [ZSH12]

During the course of the Ph.D. studies I have also published a number of papers touching different aspects of content analytics, retrieval and social search. Not all aspects are touched in this thesis due to space limitation. The complete list of publications is as follows:

**Published journal articles**

- Volker Baethge-Kinsky, Sergej Zerr. *Die Erschließung von Primärmaterial qualitativer Studien für die Sekundäranalyse als Herausforderung für Sozialwissenschaften und Informatik* Datenbank-Spektrum, Springer Berlin Heidelberg, 15(1):33-39 (2015). [BKZ15]

- Ivana Marenzi, Sergej Zerr. *Multiliteracies and active learning in CLIL - the development of LearnWeb2.0.* IEEE Transactions on Learning Technologies (TLT) 5(4): 336-348 (2012). [MZ12]

- Ivana Marenzi, Sergej Zerr, Fabian Abel, Wolfgang Nejdl. *Social sharing in LearnWeb2.0.* International Journal of Continuing Engineering Education and Life-Long Learning (IJCEELL), Vol.19, No.4/5/6, pages 276-290 (2009). [MZAN09]

- Ivana Marenzi, Elena Demidova, Wolfgang Nejdl, Daniel Olmedilla, Sergej Zerr. *Social software for lifelong competence development: Challenges and infrastructure.* International Journal of Emerging Technologies in Learning (iJET) 3(S1): 18-23 (2008). [MDN+08]

**Papers published in conference proceedings**

- Markus Rokicki, Sergej Zerr, Stefan Siersdorfer. *Groupsourcing: Team competition designs for improving the cost effectiveness of crowdsourcing.* In Proceedings of the 24 International World Wide Web Conference, WWW 2015, May 2015 (to appear). [RZS15]

- Markus Rokicki, Sergiu Chelaru, Sergej Zerr, Stefan Siersdorfer. *Competitive game designs for improving the cost effectiveness of crowdsourcing*. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM'14, pages 1469-1478, New York, NY, USA, 2014. ACM. [RCZS14]

- Sergej Zerr, Stefan Siersdorfer, José San Pedro, Jonathon S. Hare, Xiaofei Zhu. *NicePic!: A system for extracting attractive photos from Flickr streams.* In Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'14, pages 1259-1260, New York, NY, USA, 2014. ACM. [ZSP+14]

- Nam Khanh Tran, Sergej Zerr, Kerstin Bischoff, Claudia Niederée, Ralf Krestel. *Topic Cropping: Leveraging latent topics for the analysis of small corpora.* In Proceedings of the International Conference on Theory and Practice of Digital Libraries, TPDL 2013, volume 8092 of Lecture Notes in Computer Science, pages 297-308. Springer Berlin Heidelberg, 2013. [TZB+13]

- Mihai Georgescu, Dang Duc Pham, Nattiya Kanhabua, Sergej Zerr, Stefan Siersdorfer, Wolfgang Nejdl. *Temporal summarization of event-related updates in Wikipedia.* In Proceedings of the 22nd International World Wide Web Conference, Companion Volume, WWW'13, pages 281-284. International World Wide Web Conferences Steering Committee, 2013. [GPK+13]

- Fan Deng, Stefan Siersdorfer, Sergej Zerr. *Efficient Jaccard-based diversity analysis of large document collections*. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM'12, pages 1402-1411, New York, NY, USA, 2012. ACM. [DSZ12]

- Sergej Zerr, Stefan Siersdorfer, Jonathon Hare, Elena Demidova. *Privacy-aware image classification and search.* In Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'12, pages 35-44, New York, NY, USA, 2012. ACM. [ZSHD12]

- Sergej Zerr, Stefan Siersdorfer, Jonathon Hare. *PicAlert!: A system for privacy-aware image classification and retrieval.* In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM'12, pages 2710-2712, New York, NY, USA, 2012. ACM. [ZSH12]

- Sergej Zerr, Kerstin Bischoff, Sergey Chernov. *GuideMe! The world of sights in your pocket.* In Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE'11, pages 1348-1351, Washington, DC, USA, 2011. IEEE Computer Society. [ZBC11]

- Ivana Marenzi, Rita Kupetz, Wolfgang Nejdl, Sergej Zerr. *Supporting active learning in CLIL through collaborative search.* In Proceedings of 9th International Conference on Web-based Learning, ICWL'10. Springer, 2010. [MKNZ10]

- Fabian Abel, Ivana Marenzi, Wolfgang Nejdl, Sergej Zerr. *Sharing distributed resources in LearnWeb2.0.* In Proceedings of the 4th European Conference on Technology Enhanced Learning, EC-TEL'09, pages 154-159. Springer, 2009. [AMNZ09b].

- Sergej Zerr, Daniel Olmedilla, Wolfgang Nejdl, Wolf Siberski. *Zerber+R: top-k retrieval from a confidential index.* In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database

Technology, EDBT'09, pages 439-449, New York, NY, USA, 2009. ACM. [ZONS09]

- Sergej Zerr, Daniel Olmedilla, Juri Luca De Coi, Wolfgang Nejdl, Piero A. Bonatti, Luigi Sauro. *Policy based protection and personalized generation of web content.* In Proceedings of the 2009 Latin American Web Congress, LA-WEB'09, pages 112-119, Washington, DC, USA, 2009. IEEE Computer Society. [ZOC+09]

- Sergej Zerr, Elena Demidova, Daniel Olmedilla, Wolfgang Nejdl, Marianne Winslett, Soumyadeb Mitra. *Zerber: r-confidential indexing for distributed documents.* In Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, EDBT'08, pages 287-298, New York, NY, USA, 2008. ACM. [ZDO+08]

- Sergej Zerr, Wolfgang Nejdl. *Privacy preserving document indexing infrastructure for a distributed environment.* Proceedings of the VLDB Endowment (PVLDB), 1(2):1638-1643, 2008. [ZN08]

**Papers published in workshop proceedings**

- Sergej Zerr, Odysseas Papapetrou, Elena Demidova. *HEALTH+Z: Confidential provider selection in collaborative healthcare P2P networks.* In Proceeding of the 1st International Workshop on Privacy-Preserving IR co-located with 37th Annual International ACM SIGIR conference, volume 1225 of PIR'14, pages 1-6. CEUR-WS.org, 2014. [ZPD14]

- Sergej Zerr, Mathieu d'Aquin, Ivana Marenzi, Davide Taibi, Alessandro Adamou, Stefan Dietze. *Towards Analytics and Collaborative Exploration of Social and linked Media for Technology-Enchanced Learning Scenarios.* In Proceedings of the 1st InternationalWorkshop on Dataset PROFIling&fEderated Search for Linked Data co-located with the 11th Extended Semantic Web Conference, volume 1151, PROFILES'14. CEUR-WS.org, 2014. [ZdM+14]

- Sergej Zerr, Elena Demidova, Sergey Chernov. *deskWeb2.0: Combining desktop and social search*. In Proceedings of the Desktop Search Workshop, In conjunction with the 33rd Annual International ACM SIGIR 2010 Conference, Geneva, Switzerland, 2010. [ZDC10]

- Fabian Abel, Eelco Herder, Ivana Marenzi, Wolfgang Nejdl, Sergej Zerr. *Evaluating the benefits of social annotation for collaborative search.* In Proceedings of the 2nd Annual Workshop on Search in Social Media, co-located with ACM SIGIR 2009 Conference on Information Retrieval, Boston, USA, SSM'09, July 2009. [AHM+09]

- Fabian Abel, Ivana Marenzi, Wolfgang Nejdl, Sergej Zerr. *LearnWeb2.0: Resource sharing in social media*. In Proceedings of the Workshop on Social Information Retrieval for Technology-Enhanced Learning at the International Conference on Web-based Learning (ICWL'09), volume 535 of SIRTEL'09. CEUR-WS.org, August 2009 [AMNZ09a]

- Juri Luca De Coi, Daniel Olmedilla, Sergej Zerr, Piero A. Bonatti, Luigi Sauro. *A trust management package for policy-driven protection & personalization of web content.* In Proceedings of the 9th IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY'08, pages 228-230. IEEE Computer Society, 2008. [COZ+08]

- Ivana Marenzi, Sergej Zerr, Wolfgang Nejdl. *Providing social sharing functionalities in LearnWeb2.0*. In Proceedings of the TENCompetence Workshop: Stimulating Personal Development and Knowledge Sharing, pages 30-31, Sofia, Bulgaria, 2008. [MZN08]

# Contents

# List of Figures

# List of Tables

# 1 Introduction

In the last years, popularity of systems for collaborative work and content sharing increased significantly. Large amounts of data are shared among people in social networks as well as working groups in professional industry, cooperative working teams and Web 2.0 communities. Most of the Web platforms recognized the demand for resource sharing among their users and provide powerful tools for upload and sharing of the user generated content. The need for effective information distribution and search within the growing amount of documents in this context pushed forward further development of search infrastructures for enterprise data management systems. Special software supports the creation of the real, as well as of the virtual communities connecting people working on similar topics or sharing interests. However, state-of-the-art systems increasingly face challenges related to scalable and privacy preserving analysis, indexing and retrieval of shared content.

First, the power of content sharing tools and the easiness of their use often makes users careless about possible threats with respect to confidentiality of the information that can be disclosed through sharing. Thus, privacy preservation mechanisms need to become an inseparable part in all kind of sharing systems and tools. Supporting privacy preserving interconnections and confidential information exchange within enterprises, working groups and web communities in general remains a challenging research topic. User access levels and access control have to be reflected in the index structures and retrieval algorithms, as well as in ranking of search results. Therefore, one of the development goals of such systems is to consider privacy issues already during the design phase in order to hinder or even avoid data protection infringement already by design [Sch10].

Second, the rapidly growing amount of shared content makes it necessary to develop scalable algorithms that enable efficient analysis and retrieval of shared content. In this context, one important research direction towards scalable content analysis is automated partitioning of large-scale shared document collections and their indexes to create coherent subsets of manageable size. In order to estimate a number of possible clusters, techniques for scalable content diversity analysis need to be developed. Analyzing diversity of Web collections and in other large-scale text corpora *as a whole* enables estimation of the num-

ber of possible clusters in an inverted index with respect to the covered topics and access control policies. Partitioning the index according to such clustering is a promising way to further increase retrieval efficiency.

Finally, content sharing systems need to assist users in making informed decisions about the content to be shared and warn them about possible privacy breaches. In particular, young users often underestimate the consequences of their sharing behaviour [Bar06, SH10]. Also multimedia documents uploaded in large batches can contain sensitive data that require particular care. However, the amount of shared data (especially photos) oft makes it impossible for the users to analyse manually. Therefore, automatic decision support with respect to content sharing becomes crucial for privacy-protecting content sharing systems. Therefore, such systems require efficient algorithms that perform privacy-oriented content analysis automatically and warn users about potentially sensitive content, that either should not be shared at all, or requires increased protection.

## 1.1 Problems Addressed in this Thesis

In this thesis we address the problems related to scalable and privacy preserving analysis, indexing and retrieval of shared content.

**Problem 1** *How to build an indexing facility that can quickly locate relevant documents that a user is allowed to access, without: (1) leaking information about the remaining documents, (2) imposing a large management burden as users, groups, and documents evolve, and (3) requiring users to agree on a central completely trusted authority?*

The number of sensitive documents shared over social networks and enterprise intranets is growing rapidly. Sharing of access-controlled documents has traditionally been accomplished through point to point sharing techniques such as email, or through centralized repositories such as shared file directories on intranet servers, access-controlled web pages, wikis, and even source code control systems. Each of these approaches has drawbacks with respect to security, management overhead and usability. Point-to-point sharing techniques do not scale up well: emailing new versions is awkward when more than a handful of collaborators are reading and updating a document. Centralized techniques address these problems but place too much trust in the administrator controlling the central server; in a large enterprise, conflicting interests make it unlikely that everyone will be willing to give their sensitive documents to a particular superuser. Further, if the central server is compromised, all the documents stored there are also compromised.

Another option is for each user to publish their own documents on access-controlled web pages on their own web server, whose administrator they presumably trust. Alternatively, a document can be placed on a public server after encrypting it with a key known to all members. In both cases, just securing the documents is insufficient, as they need to be indexed to support efficient search and retrieval. We describe this data structure in

more detail later in background Section 2.1.1. The index contains sufficient information not only to locate relevant documents, but also to reconstruct the set of words in a sensitive document, so it needs to be protected against unauthorized access. It is unlikely that all project groups can agree on a single trusted central authority to enforce access control on index entries; even if they can, centralized indexes are attractive targets for attack and will need additional protection. For example, even if the exact content of the elements is obscured, the lengths of the posting lists can tell an industrial spy which compounds are used in the development of a new chemical process [BC05]. Protecting an inverted index is a challenging problem when there is no single trusted central authority to enforce access control on posting list elements - as is the case in the project group scenarios we target.

One possible solution is for each document owner to keep an inverted index over the documents it owns locally. Then a user's query for the term "ImClone" can be broadcast to all document owners, and the resulting answers can be collected by the user and, if desired, ranked. Each document owner retains absolute control over her index as well as her documents, and can enforce access control on each index lookup. However, this shotgun approach to querying is relatively slow, and wastes network bandwidth and computing power, since most document owners will not have posting list elements matching most queries.

Another potential solution is for the document owner to encrypt any sensitive information in each posting list element, and insert the element in a global inverted index [cCM05, SWP00]. Encryption schemes usually have complex key management schemes that make the system hard to use and administer, and can compromise its effectiveness. Further, in practice, each element includes a *term score*, reflecting importance of that term for that document. If the index provides ranked query answers, term frequencies must not be encrypted, as they are a major factor in the relevance score computed by the ranking algorithm. But from plain-text term frequencies, one can reverse-engineer the terms themselves [BC05].

In Chapter 3 we introduce a solution for securing inverted index and at the same time keeping its major properties like efficiency and top-k retrieval.

**Problem 2** *How to provide a data analytics facility for estimation of the number of structural elements in datasets that scale to Web size?*

Topical diversity of user generated content on the Web is typically very high, as meta data schema and sharing conditions in social application are relaxed to a large extend. Openly available content spans from public (like photos of landscapes and blog reports about sport events) up to highly sensitive objects such as family photos, personal diary entries and so on. Topical diversity value of a data corpora can provide insides into its structure and especially can be used to estimate a number of possible clusters in this dataset. It has been studied in different disciplines and contexts for decades. The diversity of a population can reveal certain cultural properties of a country [Lie69, Fea03], e.g. with respect to religion, ethnic groups, or political orientations. In the area of ecology, biodiversity is used as a measure of the health of biological systems [LM03]. Recently, diversity

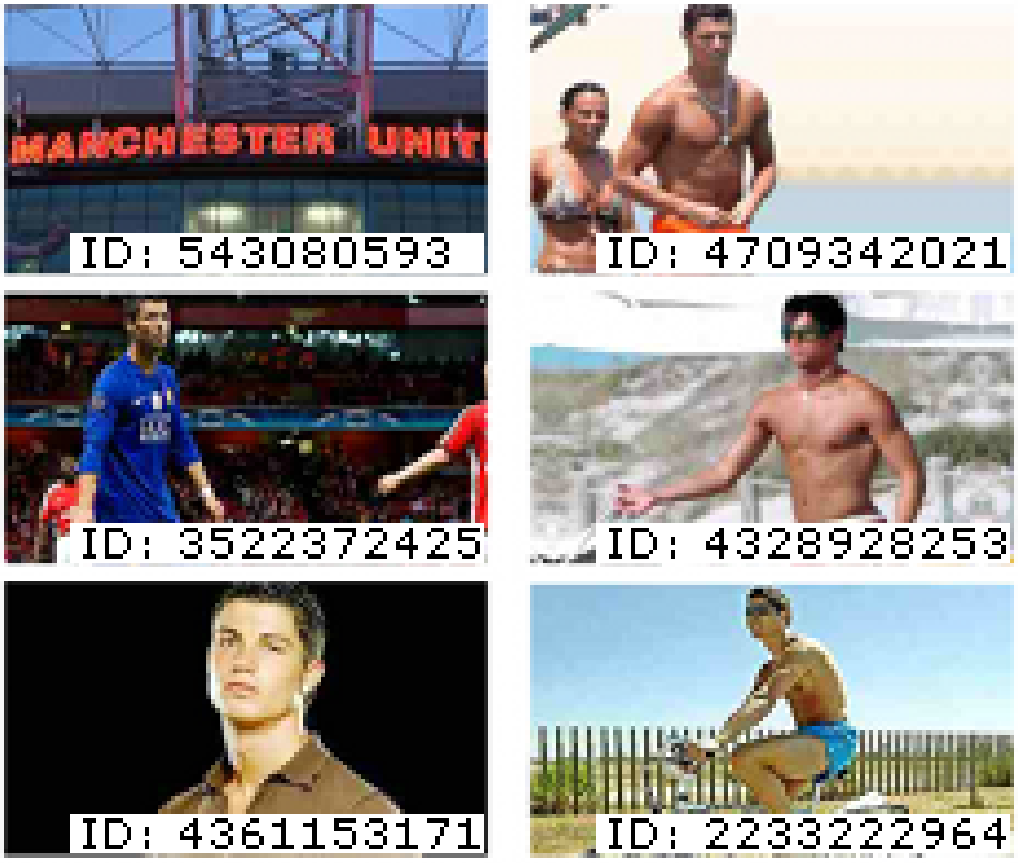
also drew the attention of scientists in the database and Information Retrieval communities. For instance, Vee et al. [VSS+08] use the sum of similarities of all object pairs to measure diversities of relational records, while Ziegler et al. [ZMKL05] employ a similar measure for diversifying items in a recommender system. In the area of search result diversification [GS09, ZMKL05], inter-object similarity is used to obtain a subset of the most dissimilar results, providing an overview over the result space. However, due to the quadratic computational complexity, the mentioned approaches are applied on relatively small sets, limited to the number of life forms in a biosystem, or top-k objects selected in the context of search result diversification. In contrast, in this chapter, we focus on analyzing diversity on Web collections and in other large-scale text corpora *as a whole* to be able to estimate a number of possible clusters in an inverted index with respect to access control policies. Portioning the index according to such clustering is a promising way to increase retrieval efficiency.

Whilst inverted index information leakage and topical diversity can be objectively measured, the notion of privacy of a specific textual or multimedial object is rather subjective. Photos taken at a beach, personal wedding or sport event photos have different sensitivity degree and thus obviously, topical diversity will have an important influence on what can be considered sensitive by a crowd. This issue will be considered in the next problem, where we exploit image features to distinguish such diverse classes of photos and identifying private/public images in social photo streams.

**Problem 3** *How to support users in estimating sensitivity degree of textual or multimedia documents and enable systems to automatically determine adequate privacy settings for these documents?*

As discussed previously, security in an inverted index comes to the price of efficiency. In practice, however, it is not necessary to secure all the documents to the same degree, even a large portion of documents often does not have to be secured at all. A mechanism to accurately judge the degree of document sensitivity can further increase retrieval efficiency by adaptable document securing. Such mechanism can contribute in scenarios where textual and multimedia content needs to be shared. Furthermore, with increasing availability of content sharing environments such as Flickr and YouTube, the volume of private multimedia resources publicly available on the Web has drastically increased. In particular, young users often share private images about themselves, their friends and classmates without being aware of the consequences such footage may have for their future lives [Bar06, SH10]. Photo sharing users often lack awareness of privacy issues. A recent study revealed that more than 80% of the photos publicly shared by young people are of such a private nature that they would not show these images to their parents and teachers [SH10]. The popular Facebook platform allows its users not only to publish photos, but also to mark the names of the depicted people. In this way, even people who did not publish any compromising information, can leave discoverable footprints on the Web.

Existing sharing platforms do not support users in making adequate privacy decisions in multimedia resource sharing. On the contrary, these platforms quite often employ rather lax




**(a)** Top-3 public photos

**(b)** Top-3 private photos

**Figure 1.1** Top-3 search results with original Flickr ids for the query "Ronaldo" (a) in the original ranking, and (b) obtained through our privacy-oriented search method (status: Oct. 2010).

default configurations, and mostly require users to manually decide on privacy settings for each single resource. Given the amount of shared information this process can be tedious and error-prone. This problem will be tackled in Chapter 5.

## 1.2 Contributions of the Thesis

In this thesis we address the problems presented above and describe three major contributions in embedding security and privacy protection mechanisms in distributed Information Retrieval systems, namely:

- First, to address Problem 1 we show how efficiency properties of an inverted index, such as fast location of relevant documents and top-k retrieval from large document

corpora can be ensured without violating privacy constrains. To achieve this goal, we propose *r*-confidentiality as a measure of the degree of information that can leak from an index about inaccessible documents, given an adversary's background knowledge of the corpus or language statistics. We use this measure to build *ZERBER* - an *r*-confidential global inverted index for sensitive documents. This index employs a novel term merging scheme that has minimal impact on index lookup costs and relies on a centralized set of *largely untrusted* index servers, encryption and **k** out of **n** secret sharing scheme [Sha79]. Following that, for effective and efficient retrieval of the top-k documents most relevant to a given query from an *r*-confidential global inverted index, we present *ZERBER+R* , a novel ranking model and a relevance score transformation function which minimizes information leakage by top-k retrieval from a confidential outsourced inverted index. The proposed solution provides tunable resistance to statistical attacks, guarantees freshness of shared documents at low cost, makes economical use of network bandwidth, requires no key management, and answers most of the queries almost as fast as an ordinary inverted index.

- Second, to address Problem 2 we develop efficient and scalable algorithms for estimation of document diversity in large corpora. In particular, to solve the diversity computational problem, we propose two novel algorithms which make use of random sampling and Min-wise independent hashing paradigms. More specifically, we propose two fast approximation algorithms for computing the average pair-wise Jaccard similarity of $n$ sets with probabilistic accuracy guarantees, coined as *SampleDJ* and *TrackDJ*. Jaccard similarity is one of the most popular measures in Information Retrieval due to its simplicity and high applicability [MRS08], and provides intuitive and interesting results in our example studies. While the proposed approach is generic and can be applied to many dimensions of diversity, in the context of this thesis we are particularly interested in its capability to estimate the number of clusters in an inverted index according to the topical dimension and document access policies.

- Finally, we address Problem 3 and show that sensitivity of a particular document, and thus adequate *privacy settings* including both textual and multimedia objects, can be estimated using machine supervised learning techniques. This estimation can directly support users in their content sharing decisions and help to make adaptable anonymisation, in systems that make use of the trade-off between privacy protection and retrieval efficiency. Furthermore, we use automatically determined document sensitivity information to develop *privacy-oriented search* and *privacy-based diversification* algorithms. *Privacy-oriented search* allows users to systematically search for private content and obtain an overview of relevant sensitive resources about themselves published by third parties at an early stage. *Privacy-based diversification* allows search engine to retrieve a "mixture" of private and public content to minimize the risk of user dissatisfaction in cases where queries are ambiguous with respect to the privacy aspect of the information need.

## 1.3 Thesis Structure

The remainder of the thesis is organized as follows:

In Chapter 2 we discuss selected general background techniques and algorithms that build a basis to achieve the goals conducted in this thesis. In particular, we focus on selected techniques from the areas of Machine Learning, Information Retrieval, similarity and diversity approximation as well as index security and privacy protection.

Following that, in Chapter 3 we discuss Problem 1 of confidential document indexing and sharing. This chapter is structured as follows: First, in Section 3.1 we briefly present the contributions of this chapter. Then, Section 3.2 describes the collaboration scenarios we want to support and presents the characteristics of the *ideal* indexing scheme for confidentiality-aware document sharing environments. Following that in Section 3.3 we describe a scenario for confidentiality-aware top-k retrieval in a collaborative environment. Section 3.4 introduces the security model. In Section 3.5 we present our approach - *ZERBER* - an *r*-confidential inverted index. Then, we discuss the problem of enabling top-k retrieval from an *r*-confidential inverted index in Section 3.6 and present a design refinement to support top-k retrieval - *ZERBER+R* - in Section 3.7. Following that, we discuss practical aspects of using the resulting system in Section 3.8. In Section 3.9 we evaluate the performance of the system with real-world data and queries. Finally, we discuss the evaluation results and the overall contributions of this chapter in Section 3.10.

Then in Chapter 4 we discuss Problem 2 of diversity degree estimation in large datasets. This estimation can help to optimize index partitioning with respect to topical dimension and access control policies and thus to improve the retrieval efficiency. First, in Section 4.1 we briefly present the contributions of this chapter. Second, Section 4.2 introduces the diversity index used throughout the chapter. Third, in Section 4.3, we describe algorithms for efficiently computing the diversity index values, and prove accuracy and efficiency properties. We verify our theoretical findings using both real and synthetic data in Section 4.4. Finally, as application examples, we show results of various corpora studies conducted on various corpora in Section 4.5.

Next, in Chapter 5 we tackle Problem 3 of the sensitivity degree estimation for textual and multimedia documents. First, in Section 5.1 we briefly present the contributions of this chapter. Second, in Section 5.2 we describe our data collection and labeling method which enabled us to obtain a large dataset for the privacy-based image classification. Third, we analyze visual and textual image features in Section 5.3. Forth, in Section 5.4 we describe the classification approach, and present ranking and diversification techniques which provide an overview of the available search results taking into account their privacy as estimated by the classifier. Finally, we evaluate our techniques for privacy-based classification, ranking and diversification in Section 5.5.

Finally, we discuss the contributions of this thesis and point out directions for future research in Chapter 6.

# 2 General Background

In this chapter we briefly introduce general background techniques and algorithms that build a basis to achieve the goals conducted in this thesis. In particular, we focus on selected techniques from the areas of Information Retrieval (IR), sampling and similarity approximation, Machine Learning (ML) as well as index security and privacy protection.

First, we present relevant Information Retrieval techniques including indexing, ranking and diversification of search results. Second, we discuss relevant similarity and diversity approximation techniques for working with large-scale data, including Locality Sensitive Hashing, Min-wise independent hashing, sampling and diversity indexes. Following that, we discuss the selected aspects of supervised Machine Learning, such as image classification, feature selection methods and evaluation metrics. Finally, we discuss relevant index security and privacy protection aspects including index security and confidentiality, data encryption, anonymization, document sensitivity and access policies.

## 2.1 Relevant Information Retrieval Techniques

In this section we provide an overview of Information Retrieval techniques related to Problem 1, namely how to build an index that enables privacy-preserving document indexing and search. In particular, this includes indexing documents using inverted indexes and ranking of search results. Furthermore, we discuss techniques of search result diversification related to Problem 3.

Information Retrieval techniques concentrate on obtaining information relevant to the user information need from an information source (a larger set of documents). The user information need is typically represented as a set of keywords and the result is a ranked list of relevant documents likely to satisfy that need. One of specific problems in the area of Information Retrieval is system efficiency dealing with obtaining relevant information in a minimum of time, which is typically addressed through indexing. Indexes are data structures containing information useful to efficiently locating of the relevant documents

given user query. Obtained documents are ranked according to their estimated relevance for the user information need. Additionally the list can be diversified to reflect aspects of information need reflected in the result set.

### 2.1.1 Inverted Indexes

Inverted Indexes are the standard choice for keyword (full-text) search of documents. An inverted index is a sequence of *posting lists*, each of which contains the identifiers (IDs) of all documents containing one particular term and a score reflecting the relevance of the document to the corresponding term. Figure 2.1 illustrates an inverted index with three posting lists and nine *posting list elements* (*elements* for short).

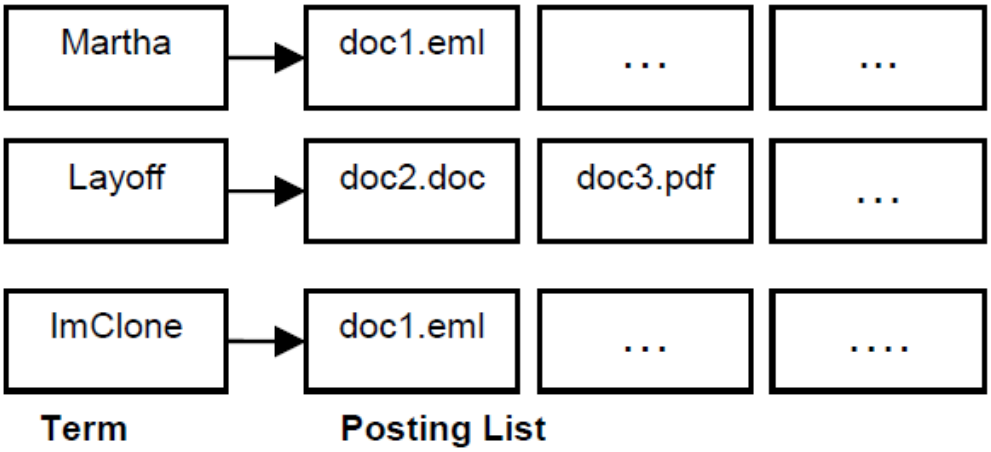



**Figure 2.1** Inverted Index with 9 Elements

In order to obtain a list of documents containing a specific term (e.g ”Martha”) it is sufficient to locate corresponding posting list. This can be done efficiently using data structures such as tree-structure (O(log(n))) or a Hashtable (O(1)) [MRS08]. The objects in a tree are sorted such that, in binary trees when touching a particular node, the larger valued node are contained to the left, and the lower valued to the right. Searching in the tree turns into descending to the corresponding node, or noticing its absence. Search trees, be B-Tree an example, are able to balance themselves and (since the pointer to the documents are contained in leaves - nodes in the lowest level) the number of comparison operations to obtain the search node is bounded by their height. Hash-table is a fixed array of lists of objects where the index (position) of the element in the array was calculated using a specific hash function involving the value of that object. Objects with the same hash value are chained in the list. In order to find an object in a hash-table, its hash value needs to be computed first. This value corresponds to the index in the table where the list containing the object is located. Finally, this list has to be scanned. The complexity of the latter procedure can be reduced when the hashtable is of sufficient length and the lists contain a minimum number of elements. Search Trees are more complex in execution compared to hash-tables, but

the former are far more efficient for updates and deletes and are typically used in dynamic contexts.

*Although inverted index is a very efficient data structure for document retrieval, it represents the summary of the collection and can leak sensitive information about containing documents. We propose a solution of this problem in Chapter 3.*

### 2.1.2 Document Ranking

After obtaining the relevant posting lists from an inverted index, the containing posting elements needs to be examined. In case of the multi-term query the corresponding posting lists are intersected and the matching documents in this intersection are ranked using one of the state-of-the-art functions. Typically the documents that are more similar to the query are ranked higher. There are different ways for computing those similarity, one of the most simplest ones is Jaccard coefficient, where the similarity is defined as intersection of terms in two documents (document and query in our case) divided by their union. The query and the document can also be represented as vectors in multidimensional vector space (having terms as dimensions). The smaller the distance between those vectors the more similar is the document and the query. Since query is typically much shorter than a document, not the absolute distance is considered, but the angle between those vectors. Unfortunately, the exact scores of the posting elements can be term specific sensitive and leak information even if the terms in the posting lists are encrypted or the posting lists are mixed.

*We discuss this problem and propose a solution in Chapter 3.*

### 2.1.3 Search Result Diversification

The result list of documents from a search engine can be very lengthy and typically a large portions of the documents within the list can be very similar to each other or even redundant. At the same time only a few results can find space on the users' screen following, instead of ordering the results by relevance, modern search engines can extract a representative subset of it, providing the overview over the whole result set. As a simplified method, the result set documents are clustered according to their position in the term vector space and only a documents from each cluster center is shown in the result list.

In order to perform search result diversification, first of all the most important topics - information nuggets - are identified for each document. Next, a subset of documents is extracted were information nugget diversity (novelty) is traded against the relevance of the documents.

The diversification quality can be measured in terms of (new) *Information Gain*, obtained by viewing a particular document subset. A trade-off between document relevance and its novelty is typically controlled by the $\alpha$ parameter. More formally, the following objective function is maximized:

$$G(D,q) = \alpha * (\sum_{d \in D} relq, d) + (1-\alpha) * (\sum_{a \in S(q)} weight(s,q) * cov(s,D)),$$

where $G(D)$ is the information gain obtained by a given set of documents the first summand represents the relevance of the subset and the second the topic coverage in this subset. More specifically, $rel(q,D)$ is the relevance score of the document *d* for the query *q*, $weight(s,q)$ is the importance of the topic $s$ for the query $q$ and, finally, $cov(s,D)$ is the topic $s$ coverage by the subset.

*In Chapter 5 we consider the problem of providing an overview of the available search results, while taking into account their privacy by developing specific diversification techniques for this task.*

### 2.1.4 Evaluation Measures for Search Result Diversification

The relevance of a search result is typically evaluated using the standard nDCG measure [JK02] that accumulates the gain of each individual search result throughout the result set. $\alpha$-nDCG [CKC+08] is a standard evaluation measure to balance relevance of search results with their novelty. $\alpha$-nDCG views a search result (i.e. an image or a document) as a set of mutually independent binary information nuggets. Each nugget represents the presence of a different aspect of the user's information need, such as a fact, or topicality of the document [CKC+08].

Compared to the standard nDCG measure, the computation of the gain $G[k]$ of a search result at rank $k$ in $\alpha$-nDCG is extended with a parameter $\alpha \in (0,1]$ to trade off relevance and novelty of a search result:

$$G[k] = \sum_{n_i \in N} J(d_k, n_i) \times (1-\alpha)^{\sum_{j=1}^{k-1} J(d_j, n_i)}, \tag{2.1}$$

where $N = \{n_1, \ldots, n_m\}$ is the space of the possible information nuggets, $J(d_k, n_i)$ is a binary user judgment of containment of nugget $n_i$ in the result $d_k$, and $\sum_{j=1}^{k-1} J(d_j, n_i)$ is the number of higher ranked results judged to contain nugget $n_i$. The resulting gain $G[k]$ is discounted by its position $k$, and accumulated over the positions $[1,k]$ to obtain the discounted cumulative gain $\mathrm{DCG}[k]$:

$$\mathrm{DCG}[k] = \sum_{j=1}^{k} \frac{G[j]}{\log_2(1+j)}. \tag{2.2}$$

Finally, the $\mathrm{DCG}[k]$ is normalized by the ideal discounted cumulative gain vector $\mathrm{DCG}'[k]$: $\mathrm{nDCG}[k] = \frac{\mathrm{DCG}[k]}{\mathrm{DCG}'[k]}$.

*In Chapter 5 we consider the problem of adaptation of $\alpha$-nDCG to the privacy-based diversification.*

## 2.2 Similarity and Diversity Approximation Methods

In this section we provide an overview of techniques related to Problem 2, namely how to provide data analytics for estimation of the number of structural elements in datasets that scale to the size of the Web. In particular, this includes methods for similarity and diversity approximation, including hashing-based methods, sampling and diversity indexes. We also discuss the relation to the work on diversity in search and query result diversification.

### 2.2.1 Hashing-Based Methods

**Locality Sensitive Hashing (LSH)** [IM98, DIIM04] is a family of approximation algorithms used for indexing high dimensional data points. Usually, in order to preserve the original properties, the value for each dimension would need to be stored for later recovery, which requires a large storage space on one hand and large times for comparing two data points dimension-wise on the other hand. Instead high dimensionality can be reduces by arithmetic operations summarizing (projection) the values of particular dimensions, resulting in a low-dimensional space. LSH has the property that two objects with a smaller distance in the resulting space are more likely to have a hash collision. The distance/similarity is measured by the $L_p$ norm (e.g. Euclidean or Hamming distance of vectors). There are various LSH implementations with different properties proposed in the literature. Approximation of cosine similarities of vectors can, for example, make use of hash functions with the LSH property [Cha02b].

**Min-wise independent hashing** is a technique originally proposed for finding near duplicate documents [Bro00a]. It can be considered as a variation of LSH and shows properties of Jaccard coefficient. This technique was initially used in the Web to determine web page duplicates in a result set, but can also be applied to measure the similarity of two document sets. Whilst there exist a number of variations of the algorithm, in this thesis we employ a particular implementation - with many hash functions. A hash function is a mapping function of an arbitrary original data into a value with a fix size. The algorithm runs as follows: Having two sets of objects it computes hash values for each object and compares the minimal values from both set. Having done it for a set of $k$ hash functions, the probability of collisions (where both minimal values equal each other) divided by $k$ equals the Jaccard coefficient. We use this property in Chapter 4 to reduce the number of pair-wise comparisons while estimating the Jaccard coefficient.

*Different from the existing applications, we are the first to use Min-wise independent hashing as a building block for efficient diversity index estimation. In particular, we use Min-wise independent hashing in our algorithm for approximating Jaccard coefficient in Chapter 4.*

### 2.2.2 Diversity Indexes

Diversity indexes were proposed in ecology [Sim49, Kre89] decades ago; they assume that objects are classified into groups. For example, Simpson's diversity index [Sim49], a well-known diversity index in ecology is defined as follows: $D = \sum_{i=1}^{Z} \pi_i^2$, where $\pi_i$ is the fraction of individuals belonging to group $i$ and each individual in the population belongs to one of $Z$ groups. Note that Simpson's index can just be employed in the special case where objects belong to one of a discrete set of categories. Diversity indexes were also proposed in various areas focusing on different applications. For example, Stirling [Sti07, RM10] proposed a general diversity index trying to measure diversities in different areas in science, technology and society. The general Stirling index is defined as $D = \sum_{ij(i \neq j)} (d_{ij})^{\alpha} (p_i p_j)^{\beta}$, where $d_{ij}$ is the distance between object $i$ and $j$; $p_i, p_j$ are the relative occurrences of object $i$ and $j$ in the whole dataset. The diversity measure used in this work for text collections can be interpreted as a special case of the Stirling index.

*None of the mentioned works deals with the efficiency problem of computing diversity measures for whole data collections, which is the main focus of Chapter 4.*

### 2.2.3 Sampling

Sampling is a technique that finds a wide range of applications, such as auditing of large databases and query optimization in databases [Olk93]. The basic idea is, that by selecting a large enough set of samples from a population, the basic properties of a distribution in this population can be approximated. Although sampling techniques deliver inexact (or error bounded) results, their execution usually requires only a fraction of time of an exact computation algorithm. A prominent example is Monte-Carlo sampling for estimation the integral of a curve, by considering few randomly selected intervals [Sob74]. A more theoretical example related to our work is the application of random sampling to estimate the average value (or sum) of a set of numbers, for which it is known to be difficult to bound the relative error without having prior knowledge about the input data [DKLR00]. More research on sampling can be found in the survey of Babcock et al. [BBD+02] and in [Olk93].

*As discussed in Chapter 4, we are the first to use random sampling as an underlying technique to solve the problem of efficiently computing diversity indexes.*

### 2.2.4 Diversity in Search and Query Result Diversification

There is a plethora of work on diversifying query or search results and adaptation of evaluation schemes in this context - see e.g. [CG98, CKC+08, GS09, AGHI09, CK06, WZ09]. Search engines, recommender systems and databases have the need to return diverse results to users in order to include answers for different user needs in the result set. This is especially important if the query is ambiguous. Several of these papers perform diversification of search results as a post-processing or re-ranking step of document retrieval. These

methods first retrieve the relevant search results, and then filter or re-order result lists to achieve diversification. Vee et al. [VSS+08] use the sum of similarities of all object pairs to measure diversities of relational records, while Ziegler et al. [ZMKL05] apply a similar measure for diversifying items in a recommender system. Gollapudi and Sharma [GS09] make use of inter-object similarity in their axiomatic search result diversification approach. In their recent work [MSN11] Minack et al. deal with efficiently diversifying search results for large data streams; note that obtaining a diverse *top-k list* from a large dataset corresponds to an entirely different problem setting than the one studied in this work.

*In general, work on search result diversification is about diversifying small top-k sets of query results. In contrast the problem of efficient computation of diversity indexes for whole data collections discussed in Chapter 4 has not been addressed sufficiently in the literature.*

## 2.3 Index Security and Privacy Protection

In this section we provide an overview of security and privacy protection techniques related to Problem 1, namely how to build an index that enables privacy-preserving indexing and search. In particular, this includes related work on index security and confidentiality, as well as data encryption and anonymization. Following that, we discuss existing work related to document sensitivity and access control policies related to Problem 3, namely how to support users in estimating sensitivity degree of the documents and enable systems to automatically determine adequate privacy settings.

## 2.4 Image Classification Using Machine Learning

In this section we provide an overview of Machine Learning techniques related to Problem 3, namely how to support users in estimating sensitivity degree of documents, and in particular images, to (semi-)automatically determine adequate privacy settings. In particular, techniques related to this problem include aspects of image classification, feature selection and quality evaluation.

Machine Learning deals with the development and evaluation of computer systems that are not explicitly programmed to achieve a particular goal, but can learn from given data examples. The algorithms that learn functions using labeled training data are typically referred to as supervised ML algorithms, whereas the algorithms that use predefined functions to directly learn from unlabeled data are commonly known as unsupervised ML algorithms. The commonly used methods in the respective areas include: 1) Classification, where a known set of labels can be assigned to a given data object using supervised ML techniques; and 2) Clustering, where the system leverages the underlying structure of a given dataset to automatically identify groups of similar data objects without labeled training data. Also a mixture of the both paradigms exists. One of the most important aspects

for the effectiveness of a ML algorithm is the selection of features that represent data objects in the learning process. These features enable ML algorithms to compare the objects, to learn specific feature patterns and to assign the objects represented by the features to categories.

*In Chapter 5 we analyse how a supervised Machine Learning classification method can be adapted in the context of privacy-oriented content analytics to determine sensitivity degree of images.*

### 2.4.1 Image Classification with Support Vector Machines

Classification is a well-studied area of supervised Machine Learning with a variety of probabilistic and discriminative models [Cha02a]. Given a number of class labels and sufficient amount of example objects for each class, a classification algorithm can learn the specific feature patterns for each class and create a model, able to assign a right label to an unseen object based on its features. Specifically, a Support Vector Machine (SVM) binary classification model can efficiently construct a hyperplane $\overrightarrow{\omega} * \overrightarrow{x} + b = 0$ that separates the set of positive training examples from a set of negative examples in a feature space, with a maximum margin. SVMs have been shown to perform very well for various classification tasks [MRS08]. In this thesis we use a popular SVMlight software package [Joa99] that provides various kinds of parameterizations and variations of classification techniques (e.g., binary classification, SVM regression and ranking, transductive SVMs, etc.).

In the context of image classification, there are three basic types of features, namely:

- Ratio / numeric (continuous numbers, such as brightness, or colorfulness);
- Ordinal (quantities with natural ordering, such as classes in an ontology or scales of a rating); and
- Nominal, i.e. categories such as tags (i.e. descriptive terms assigned by the users) and colors.

Typically, for a classification task, each nominal value is represented as a separate dimension in a multidimensional vector space; the numeric values represent the value of the feature for the given dimension and ordinal data can be used for both[1].

To give an example, assume there are two classes of photos, public and private in our case. We also define two features (a) the presence or absence of human faces in the image, and (b) the average color saturation of all pixels in the image. We will further take 10 images of each class, extract their features as 2-dimensional vectors and position the items in the 2-dimensional feature space as shown in the Figure 2.2. In this figure, the axes represent both feature dimensions, green crosses correspond to private images (positive examples) and red dashes to the public images (negative examples). The task of the

[1]For engineering reasons, it is common practice to convert numeric values into ordinal by grouping the intervals (e.g. in an histogram) or to consider ordinal data as categorical/nominal.

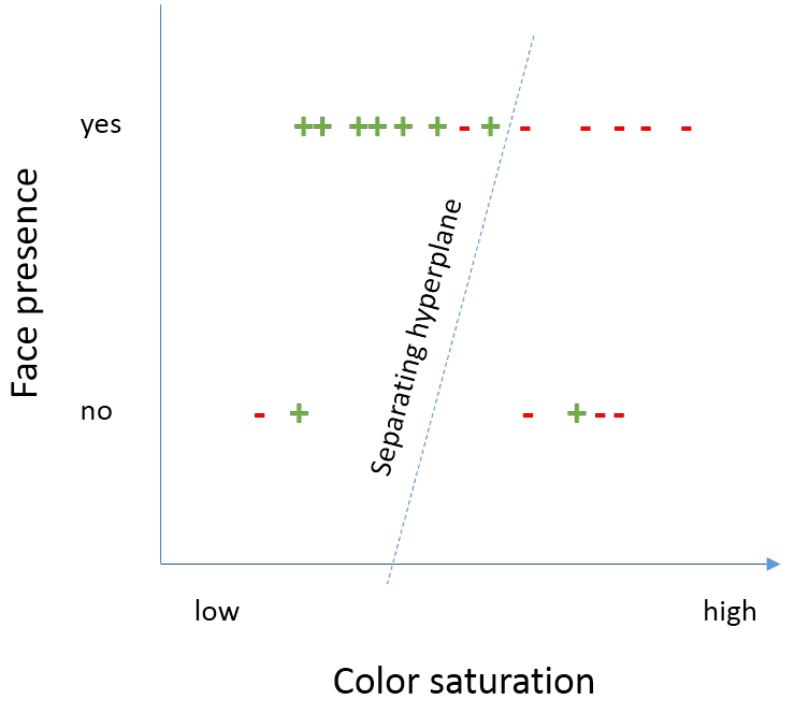



**Figure 2.2** Separating private (positive) and public (negative) image examples in a 2 dimensional feature vector space using face presence (on the $Y$-Axis) and color saturation (on the $X$-Axis) of the images as features.

SVM algorithm is to compute a hyperplane (a line in the 2-dimensional case), which can separate the positive from the negative examples, while maximising the margin between both classes. In this example we can observe that private photos are mostly located in the left top corner - the area representing photos containing faces and having lower color saturation. Intuitively, lower color saturation can correspond to the low cost cameras, privately used. In contrast, the color saturation of the public images (mostly located at the top right in Figure 2.2) is generally higher in this example.

As all supervised ML algorithms, SVM operates in two phases: 1) the training phase, where the algorithm learns and validates a function from the labeled data instances, and 2) the classification phase, in which unlabeled data instances are classified using this function. During the training phase, the task of the SVM algorithm is to find a hyperplane separating the two groups of labeled images in the vector space, such that the margins between those classes (more specifically between the data points nearest to the hyperplane) and the separating hyperplane are maximized. This training requires solving a quadratic optimization problem whose empirical performance is somewhere between linear and quadratic in the number of training images [Bur98]. However, in real life, the positive and negative training data are not always linearly separable. To deal with such tasks, SVM provides non-linear kernels as well as a so-called kernel trick to map the input in a more dimensional space where the problem becomes linearly separable again. In addition, SVM introduces slack

variables to relax the constraints. After the training has been completed, for a new previously unseen object (an image in our case), the SVM has to decide whenever the image belongs to the "positive" side or the "negative" side of the separating hyperplane. This is efficiently done through considering the sign of the scalar product of the both vectors - the hyperplane and the image.

SVMs have been shown to perform very well for various classification tasks (see, e.g., [DC00]). *In Chapter 5 we analyse the applicability and efficiency of the SVMs for private/public image classification tasks.*

### 2.4.2 Feature Selection using Mutual Information

The effectiveness of a ML classification algorithm like SVM depends, to a large extent, on the selection of discriminative features that can be used to separate images belonging to different classes. In this context, we rely on the Mutual Information (MI) [MRS08]. Given two variables, the Mutual Information measures the dependency of these variables on each other. Using the MI measure we can compute, how much a joint distribution of objects within categories deviates from a hypothetical distribution, in which objects and categories are independent of each other. The least dependent objects in each of the groups can be seen as a representative for this group. As an example, consider two sets of documents, one about winter sports, another one about summer sports (Table 2.1).

| Winter Sports | | | Summer Sports | | |
|---|---|---|---|---|---|
| $Doc_1$ | $Doc_2$ | $Doc_3$ | $Doc_4$ | $Doc_5$ | $Doc_6$ |
| sun | sun | ice | sun | sun | ball |
| ski | bob | hockey | speed | rocket | player |
| snow | ice | player | bicycle | ball | hot |
| cold | speed | cold | warm | grass | goalkeeper |
| speed | | goalkeeper | | player | |

**Table 2.1** Example: Typical Terms in the Documents about Winter Sports and Summer Sports

As Table 2.1 illustrates, whilst the terms "sun" and "player" will appear in the both datasets, the terms "snow", and "warm" are more common in the winter and summer set, respectively. A special case of MI is a TF-IDF measure, which, given a set of documents, prefers the terms that are frequent in a particular document and infrequent in all others. In this work we employ the MI measure to calculate the most representative textual and visual features, for either "public", or "private" class of objects (images in our case).

*In Chapter 5 we analyse effectiveness of selected textual and visual features in the context of private / public image classification tasks using Mutual Information.*

### 2.4.3 Quality Evaluation for Machine Learning Algorithms

After a Machine Learning model has been trained it is important to evaluate the model and assess its quality. In a perfect case, example instances provided to train the model form a representative sample for the particular classification problem. This sample also needs to be large enough, such that the main feature characteristics can be captured over the noise. In the real world a model often can be undertrained - meaning that the number of the training instances is not sufficient to capture the classification problem. Another extreme is overfitting, when the model learns the training examples rather then the classification problem itself. In both extremes such models are not generalizable, meaning that the model will only work on the training data rather than for the previously unknown instances.

#### Accuracy and Precision

Typically, to evaluate the quality of classifier prediction, the sizes of the four main classes produced by the model are captured:

- *True Positive (TP)* - positive objects that are correctly classified by the model.
- *False Positive (FP)* - negative objects that are wrongly classified by the model as positive.
- *True Negative (TN)* - negative objects that are correctly classified by the model.
- *False Negative (FN)* - positive objects that are wrongly classified by the model as negative.

Different classification models can perform better for different tasks and the quality measures can make them comparable. In the following we describe some of the established quality measures used in this work.

The most simple way to access the quality of a model on a test set is to measure its accuracy. The accuracy is defined as the proportion of correctly classified instances in the test set:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Although this measure is simple, it fails in case the test set is imbalanced (the number of positive and negative instances is not equal). For example, if the number of sensitive documents in a company database is very small, a model which classifies every document as non-sensitive will be considered as accurate.

Precision is another measure, which takes into account the prediction quality, only for the examples classified as positive by the model. It measures the portion of correctly classified instances among all instances classified as positive by the model:

$$precision = \frac{TP}{TP + FP}.$$

Recall measures the portion of correctly classified positive instances to all positive instances in the test set:

$$recall = \frac{TP}{TP + FN}.$$

Usually, there is a trade-off between precision and recall. $F$-measure combines both as a weighted average (or harmonic mean):

$$F1 = 2 * \frac{precision * recall}{precision + recall}.$$

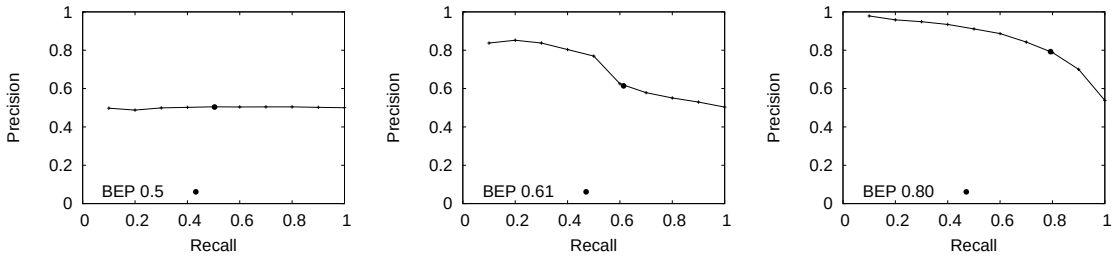
### Precision-Recall Curves

The precision and recall measures represent averaged values and do not reflect all properties of the model performance. Therefore, in this thesis we use precision-recall curves to measure the performance of generated models. A precision-recall curve shows the precision (on the $Y$-Axis) over recall levels (on the $X$-Axis). More precisely, in order to construct a precision-recall curve, in the first step a dataset consistent of 50% positive and 50% negative examples is classified and instances are sorted by their classification values in descending order. In the second step, the head of this sequence is separated such that it contains exactly a certain percentage (e.g. 10%) of the positive instances (which corresponds to 10% recall). Finally, precision value in the resulting subset is calculated as the first point of the curve and this procedure is continued for further recall levels (20%, 30% and so on). The higher the curve, the better is the performance of the examined classifier. The precision-recall graph gives insights in how much precision it is possible to gain in case only a certain amount of recall is required. Typically, the head of the distribution achieves better precision due to the fact that higher classification values correspond to the higher confidence of the classification model. Figure 2.3 depicts three examples of precision-recall curves illustrating different classification properties of the underlying models.

### Cross Validation

One of the possible issues in classification is overfitting, which occurs in cases where, instead of capturing a classification problem, the training dataset is "overlearned". In such cases the classifier performs very well on the training examples, but fails for the unseen instances. In the public/private classification application it would mean that classifier learns feature distribution of concrete photos instead of typical features reflecting the public/private context of the photos.

A general way to asses the generalizability of a particular classification model is to randomly partition a large set of labeled instances into two complimentary subsets and perform training on one subset and testing on the other. This can be done in multiple rounds using different partitions. The most common is the 10-fold cross validation, where the model is trained on the 90% of the data and tested with the remaining 10% in 10 rounds.




**(a)** Low classification ability

**(b)** Partial classification ability

**(c)** High classification ability

**Figure 2.3** Examples of precision-recall curves. The first curve 2.3a fluctuates around the value of 0.5, indicating that the classification ability of the model is very low and is close to random. The curve 2.3b depicts a limited classification ability. Around 50% of the dataset was classified with around 80% precision, followed by a sharp decrease. Finally, Figure 2.3c shows a typical classification curve which starts high, lowering only slowly.

The results of the individual rounds are averaged and the average precision can be taken as a measure of quality.

*In Chapter 5 we apply the evaluation techniques discussed above in the context of private/public image classification tasks.*

### 2.4.4 Related Work on Image Classification

Analysis of visual and textual image (meta-)data is applied to tackle a variety of problems, such as determining attractiveness [SPS09] or quality [YHBO10] of photos, identification of landmarks [ACN+09], search result diversification [vLGOvZ09], and others. Figueiredo et al. [FBP+09] analyze the quality of textual features available in Web 2.0 systems and their usefulness for classification.

*In comparison to these works, in Chapter 5 we apply classification techniques in a novel context to automatically determine image privacy and study textual and visual features in the context of privacy-oriented classification and search.*

### 2.4.5 Index Security and Confidentiality

Index security has been addressed by many techniques designed for the outsourcing threat model, where the goal is to secure the index from tampering by the untrusted storage server. These techniques store the index in plain text, and so do not address confidentiality. For example, Merkle hash trees let one verify the authenticity of any tree node entry by trusting the signed hash value stored at the root node. Authenticated dictionaries [BG05, GTS01] support secure lookup operations for dictionary data structures. Encryption is a standard

technique for storing data confidentially [Bla93, GTS01, KRS+03]. [BCF01, MS03] provide a framework for policy-based protection of XML data by encryption. Other techniques include suppressing and/or generalizing released data into less specific forms, so that they no longer uniquely represent individuals [FWY05, Iye02]; k-anonymity is one popular form of generalization (e.g., [BA05, LDR06, MKGV07]). Unfortunately, it is not possible to directly apply these techniques to secure an inverted index. Even if posting list entries are encrypted, they can leak critical statistical data.

The research most relevant to our problem is $\mu - Serv$, a system developed at IBM to index distributed access-controlled documents [BBAV09]. $\mu - Serv$ has a centralized index based on a Bloom filter; it responds to a keyword search by returning a list of sites that have at least $x\%$ probability of having documents containing one of the query keywords, where $x$ is a preset parameter. Users then repeat their query at each suggested site. The lack of precision in results from the central index represents a trade-off between search efficiency and confidentiality preservation. This approach lengthens the querying process and wastes cycles at sites that do not contain query-relevant entries. For example, if $x = 5\%$, the user must query 20 times as many sites to get the relevant results. Further, $\mu - Serv$ does not support centralized ranking; the user must get ranked search results from individual sites and combine them.

*In our approach ZERBER proposed in Chapter 3, the centralized indexes direct users to documents that definitely satisfy the user's query, and provides a level of confidentiality equivalent to $\mu - Serv$ at much lower query cost: because ZERBER provides exact search results, users can rank their search results locally and visit only the top-k document server sites to obtain document snippets.*

While many other researchers have addressed aspects of data confidentiality, none of their schemes are intended for an environment with many dynamic collaboration groups. For example, researchers have suggested ways to search encrypted text or tables stored on a remote untrusted server (e.g., [BCOP04, cCM05, HILM02, SWP00]). In a situation with many collaboration groups with dynamically changing membership, these approaches are not easy to use or manage. Document owners and/or project group managers must generate and distribute keying material for all group members, so that they can encrypt keywords and decrypt search results. If a key is lost, stolen, or even published, the index entries encrypted with it are compromised. When a key is compromised or a member leaves a group, the key must be revoked and all the content associated with that key must be re-encrypted and re-indexed. Modern group key management schemes, such as logical key trees [CLK04] and broadcast encryption, reduce the costs associated with giving keys to members, but still require content re-encryption. Some approaches also require that the entire index for a particular collection of documents be regenerated by the collection owner every time an entry is added to or deleted from the index.

*Our approach ZERBER proposed in Chapter 3 does not use keys. Distributed indexes such as Distributed Hash Tables (DHTs) are popular for P2P networks [BMT+05]. ZERBER distributes complete instances of an encrypted index to multiple servers for security reasons, while in DHTs each peer typically stores only a fraction of the index. An applica-*

*tion of* r*-confidential indexing to a DHT-based infrastructure was also considered in our research [ZPD14].*

### 2.4.6 Data Encryption

Encryption is a most natural technique to secure data. It is a transformation of the input data (plain text) into encrypted form (chiffre) using a key, such that original contents can not be directly recovered without the knowledge of the key. More formally:

$$plaintext = dec(enc(plaintext, key), key),$$

where $enc$ and $dec$ are encoding and decoding methods, respectively.

Both, plain text and chiffre are basically byte arrays and so is also the key. There is a large number of encryption techniques varying in terms of strength (the more effort is needed to recover the plain text, the stronger is the technique) and efficiency. Generally, the key is not always needed and strength of the technique can be concentrated in the method alone, but common assumption is that it is improbable that the method can be kept secret for a long time. Usually, the method can be made public and security should rely alone in the knowledge of the key.

*Although, strong encryption techniques and secure key exchange algorithm have been well developed for texts, they can not always hold their security properties on (semi-)structured data such as inverted indexes (we will discuss this issue in Chapter 3) and database tables.*

### 2.4.7 Data Anonymization

Companies and institutions, where a large amount of data is produced, often do not have enough expertise or resources to process, mine and analyse this data. Similarly, centralized inverted indexes (that can be considered as a subset of the data, required to navigate through the whole corpora) for big datasets can be typically located on external servers that are optimized for the Information Retrieval tasks. (This is also the case for the popular search engines). However, outsourcing of datasets containing sensitive information is not possible in such cases. The IT area of information outsourcing is dealing with the problem of data anonymization, transforming the data into a form, where desired analytics are possible, but neither relations to concrete persons can be identified, nor other sensitive information is contained in its clear form.

A naive solution for data anonymization in these scenarios would be encryption of the sensitive attributes and person names in the outsourced dataset. For structured data, such as a set of database records with person specific data, even encryption of the plain names of the persons is not always sufficient to prevent their re-identification. This problem is tackled in the area of information outsourcing. Each tuple in a relational database has (to some extend) recognizable signature in terms of its attribute combination and their values.

Encryption of those values does not not always help as sometimes even a presence of an attribute value leaks information about the person. Furthermore, encryption can make the data unusable for further analytics.

To deal with this problem, the notions of *k-anonymity* was introduced [BA05, LDR06, MKGV07]. The idea is to make each record in the database indistinguishable from at least k-other records in the same set using *suppression* - selective removal of attributes and *generalization* - binning ranges for attributes (e.g. age, location), and generalization of user categories (e.g. occupation of users, type of diseases). Such anonymized dataset keeps general properties and blurring the details that are not important for the analytics at the same time. Each of the clusters obtained this way is called an equivalence class.

*l-diversity* additionally strengths the k-anonymity defining that at least $l$ values of a sensitive attribute in each equivalence class of should not be identical, more specifically, ensuring that the most common value does not appear too often and less common values are not appearing too infrequently in the batch. *t-closeness* additionally generalizes this notion by ensuring that the distribution of the values in each equivalence class does not vary more than a threshold $t$.

*In Chapter 3 we propose a posting list merging techniques, where each merged list can be seen as an equivalence class.*

### 2.4.8 Document Sensitivity and Access Policies

**Document sensitivity** There are several works studying user privacy attitudes in social networks, privacy decisions upon sharing multimedia resources, and possible risks. In their early work [GA05], Gross and Aquisti studied privacy settings in a large set of Facebook users, and identify privacy implications and possible risks. Lange [Lan08] studied user behavior with respect to revealing personal information in video sharing. In [SH10] photo-sharing habits of young people in Germany have been analyzed. All of these papers point out lack of user awareness regarding the exposure of the aggregated contextual information arising from users' resource sharing habits.

*In Chapter 5, we provide a way to automatically identify privacy-relevant content of images to increase user awareness and support privacy-related decisions.*

**Access policies** There is a plethora of work dealing with the problem of establishing suitable access policies and mechanisms in social web environments. Caminati and Ferrari [CF08], for example, proposed collaborative privacy policies as well as techniques for enforcing these policies using cryptographic protocols and certificates. Felt and Evans [FE08] suggested to limit access to parts of the social graph and to certain user attributes. Squicciarini et al. [SSP09] introduced privacy mechanisms in social web environments where the resources might be owned by several users. In [AGS09], the authors discussed the problem of defining fine-grained access control policies based on tags and linked data. The user can, for instance, create a policy to specify that photos annotated with specific tags like "party" can only be accessed by the friends specified in the user's Friend-of-a-Friend

(FOAF) profile. In this work, we do not focus on access mechanisms or policies. Instead, we concentrate on the *automatic* identification of private resources (more specifically, photos) and on privacy-oriented search. Vyas et al. [VSCY09] utilized social annotations (i.e. tags) to predict privacy preferences of individual users and automatically derive personalized policies for shared content. These policies are derived based on a semantic analysis of tags, similarity of users in groups, and a *manually* defined privacy profile of the user. Ahern et al. [AEG+07] studied the effectiveness of tags as well as location information for predicting privacy settings of photos. To this extent, tags are *manually* classified into several categories such as Person, Location, Place, Object, Event, and Activity.

*In comparison to the existing works, our approach presented in Chapter 5 does not require manual steps, and, in addition, makes use of the image content, which allows us to utilize visual features to determine adequate privacy settings even for resources that do not have any associated tags.*

# 3

# Privacy Preserving Content Indexing and Retrieval

In this chapter we target Problem 1 of supporting efficient indexing and top-k retrieval for sensitive unstructured documents shared within collaboration groups. These groups can reside within a large enterprise, or may even span multiple enterprises, so there is no central authority that all members trust with the content of their documents. Nevertheless, members are willing to trust the enterprise's authentication facilities. Such environments are common in large companies, large universities, and large government groups.

## 3.1 Contributions

The contributions presented in this chapter address Problem 1 and cover multiple aspects of privacy-preserving indexing and retrieval, as follows:

i First, we propose *r*-confidentiality as a measure of the degree of information that can leak from an index about inaccessible documents, given an adversary's background knowledge of the corpus or language statistics.

ii Second, we propose *ZERBER* [1], an *r*-confidential global inverted index for sensitive documents. *ZERBER* relies on a centralized set of *largely untrusted* index servers that hold posting list elements encrypted with a **k** out of **n** secret sharing scheme [Sha79], which provides complete resistance against inappropriate information disclosure regarding pre-existing documents even if **k**-1 index servers are compromised. To provide tunable resistance to statistical attacks, *ZERBER* employs a novel term merging scheme that has minimal impact on index lookup costs. *ZERBER* guarantees freshness of shared documents at low cost, makes economical use of network bandwidth, requires no key management, and answers most of the queries almost as fast as an ordinary inverted index.

[1] A mythical three-headed watchdog.

iii Third, for effective and efficient retrieval of the top-k documents most relevant to a given query from an *r*-confidential global inverted index, we further enhance our initial *ZERBER* approach and present *ZERBER+R* , a novel ranking model which minimizes information leakage by top-k retrieval from a confidential outsourced inverted index by supporting sorted indexes which do not exhibit additional information to a potential adversary. We propose a novel relevance score transformation function, *RSTF* for short, a heuristic which hides term specific distribution of relevance score values, making scores of different terms indistinguishable. This heuristic enables inclusion of (transformed) relevance scores in the posting elements on an untrusted server to allow the server answering top-k queries efficiently.

iv Finally, we demonstrate retrieval effectiveness and efficiency of *ZERBER* and *ZERBER+R* on two real-world datasets.

## 3.2 The Ideal Solution

The content of the shared documents evolves over time, as does the group membership. Groups come and go relatively quickly, as projects start and finish. As each person can only accomplish a certain amount of work, in practice she will belong to a limited number of collaboration groups.

Given a keyword query, the *ideal* indexing scheme's answer will be identical to that of a trusted centralized ordinary inverted index that incorporates an access control list check on the ranked document list just before returning it to the user. In general there is a trade-off between retrieval effectiveness of the index and confidentiality it can provide. On the one hand in order to effectively answer a query, an index server requires possibly complete ranking information enclosed in its posting elements. On the other hand this information gives undesirable insights into the content of the indexed documents. The ideal system should enable the user to control such trade-off through some tunable parameter. The ideal indexing scheme will answer queries and handle updates as fast as an ordinary inverted index, and with no greater network bandwidth or storage usage. Changes in group membership will be immediately reflected in the query answers of the ideal indexing scheme. The ideal indexing scheme will impose no management burden on group members, beyond the requirement that the group coordinator maintain a list of the identities of the people in the group, the group members know how to authenticate to those identities, and the group members have trusted desktop or local web servers where they can upload their sensitive content into appropriate access-controlled directories and have a daemon automatically ensure that the corresponding index updates are carried out quickly. If the server or servers containing the ideal scheme's index are compromised, no information about the content of the documents should be revealed. No user or superuser on a non-trusted machine should be able to obtain any information about the content of sensitive indexed documents that they are not authorized to access.

The ideal indexing scheme will be unattainable in practice, but we can quantify the degree to which any proposed approach falls short. We can also provide schemes that provide tunable trade-offs between the confidentiality guarantees provided by the index and its query and update processing overhead.

## 3.3 Scenario

PCC (Production Control Company) creates adaptive solutions for production process controlling in manufactures. These solutions include special soft and hardware which is adapted according to the needs of every specific customer. A lot of electronic documents such as project and scientific documentation, stuff management, e-mail correspondence, presentations, collaborative documents and others have to be shared among the partners in the specific projects of PCC.

John is a leader of several projects within PCC. Each project corresponds to a customer manufacture. In order to always obtain up-to-date documents for his projects and share appropriate information with team members of a specific project, John requires a privacy-preserving centralized sharing and search facility. Because of the very sensitive nature of shared data, the advisory board of PCC decides to use *ZERBER* as the indexing system. *ZERBER* supports selectively sharing of access-controlled documents and delivers precise search results and at the same time preserves the $r$-confidentiality of the shared data.

As John has access to a huge number of documents, he is not interested in obtaining all the documents containing query terms, but rather a few documents most relevant to the query. An ordinary search engine can pre-select such documents using relevance score values attached to each posting element. However, these scores are calculated based on sensitive statistical information such as e.g. term frequency. A term frequency distribution is sufficient to characterize the subject matter of a lengthy document, and the likely content of a short email, giving adversary insights in the content of the indexed documents. To avoid this problem *ZERBER* stores encrypted relevance values in the posting elements and returns references to all the documents matching a query, such that only an authorized user can decrypt and rank posting elements on the client side.

Due to the nature of his job John has to travel a lot and uses *ZERBER* search interface with help of a PDA and GPRS internet connection. As this kind of connection is very slow the data volume transmitted over it needs to be minimized. To reduce data transfer the server needs some means to identify the best documents that match Johns' query and return only the best top-k search results.

In the following, we show how to construct a server-side index which supports top-k retrieval while preserving the given confidentiality guarantees.

## 3.4 Threat Models

In this section we first discuss possible attacks on the inverted index. Then, we discuss additional attacks that can become possible if the index structure provides additional information to support top-k retrieval.

### 3.4.1 Attacks on the Inverted Index

To give a sense of the set of potential dangers, consider the following three goals of a potential attack on an index.

A1. *Reconstruct the exact content or the term frequencies of an inaccessible document.*

Exact reconstruction is clearly undesirable. The *term frequency* distribution, i.e., the number of occurrences of each term in the document, is sufficient to characterize the subject matter of a lengthy document, and the likely content of a short email.

A2. Determine the aggregate term/document frequencies for the set of inaccessible documents at a participating site.

The *document frequency* is the number of documents at a site that contain a particular term. In an ordinary inverted index, the length of a term's posting list is its (global) document frequency. These frequency distributions will often suffice to characterize the nature of a project that the site's owner is participating in. For example, one might be able to tell that the CEO is considering a buyout offer from a particular suitor, or tell what solution approach the smartest project group in a course has adopted. Document frequencies can also tell an industrial spy which compounds are used in the development of a new chemical process [BC05].

A3. Determine whether a particular term appears in a particular inaccessible document at a particular site, or at any indexed site.

For example, curious employees may want to know whether Mildred Hesselhofer of IBM is a finalist for the CEO job at HP. Rare terms like "Hesselhofer" especially need this protection.

The conflicting interests of enterprise project groups make it hard for them to trust a centralized corporate index server. Even with a distributed index, an attacker Alice will already have some background knowledge about the possible contents of a document collection. We will restrict Alice's ability to increase this knowledge, even if she takes over an index server and can examine the contents of that server. More formally, we will **bound the ability of an adversary to make probabilistic claims about the contents of a document collection**. From her background knowledge $B$ and the parts of the index structure $I$ that

she can access, Alice will know a-priory that a term $t$ is contained in document $d$ with a probability P($t$ is in $d$). For example, for a set of emails, $B$ should include P("Subject" is in $d$) = 1, for all $d$ and $I$. We cannot control the probability estimate $P(X|B)$ about fact $X$ that Alice can make based on $B$, but we can limit her ability to refine that estimate when she computes $P(X|I,B)$. In the remainder of the chapter, we will consider only facts $X$ of the form "term $t$ is in document $d$" and "term $t$ is not in document $d$".

**Definition 3.1 ($r$-Confidential Indexing Scheme)** *An indexing scheme is* r-*confidential iff*

$$\frac{P(X|B,I)}{P(X|B)} \leq r. \tag{3.1}$$

Here, $r$ is the factor of maximal probability amplification for term $t$ in $d$ given $I$. In other words, P($t$ is/is not in $d \mid B$) $\leq \alpha$ implies P($t$ is/is not in $d \mid B$, $I$) $\leq r * \alpha$. The indexing scheme offers maximal protection when $P(X|B) = P(X|I,B)$, i.e., $I$ does not provide any additional knowledge about $X$. $r$-Confidentiality focuses on document content confidentiality. In addition, secure communication channels such as https should be used to provide confidentially for the content of queries and updates. If no one should be able to tell that a particular user sent a request to an index server, we recommend the use of MIX networks and other standard techniques from network security that foil traffic analysis attacks. An attacker could compromise a non-index site so that it, for example, gives query results or actual documents to unauthorized parties. Such attacks should be guarded against using standard techniques, and we do not consider them further, as our goal is to secure the *index*.

To execute a keyword query, the user first authenticates herself to an index server and supplies the query terms to the server as well as $k$ - the desired number of the top-k documents. The index server determines users' access rights, identifies posting list containing query terms and returns the highest ranked elements from the requested list.

### 3.4.2 Additional Attacks on the Index for Top-k Retrieval

In order to enable the server to identify the top-k elements relevance scores of each posting element must be visible to the index server. This additional knowledge can compromise confidentiality of the index. Specifically, we want to bound the ability of an adversary to perform the following attacks:

A4. *Identify terms represented by the posting elements by analyzing relevance score values stored in the index. Posting List ID Posting Elements.*

An adversary Alice could use relevance score distribution statistics to extract specific features like score ranges, or score distribution patterns for each particular term. Alice could compare extracted features with the relevance score distribution in the posting lists to find correlations. If the terms are encrypted Alice could use these statistics to break the encryption. In a merged index, in case a simple term frequency based scoring function is used,

she could claim that "frequent terms are more probably located in the head of the merged posting list" and even undo the posting list merging.

A5. *Determine query terms of other users by observing queries and query results.*

In case of a merged ordered posting list, the number of requests required for obtaining top-k elements for a rare or a frequent term may differ. Alice can also know the $k$-value which is requested by the client application. As document frequency is term specific, Alice could guess the term by observing the number of follow-up requests required to fill the top-k results.

To guard against these attacks *ZERBER+R* proposes novel techniques making relevance scores and number of follow-up requests for different terms indistinguishable for the server while preserving retrieval accuracy of server-side top-k processing.

## 3.5 *ZERBER* Design

*ZERBER* is an *r*-confidential inverted index that incorporates secret splitting and term merging. After a quick overview of the reasons for these choices, we discuss each feature in detail.

As mentioned earlier, encryption is the classic way to protect information on an untrusted server, but has significant drawbacks with respect to key compromise, revocation, and manageability. We avoid the need to distribute *any* keys to group members by applying a *k* out of *n* secret sharing scheme [Sha79] to posting list entries. Each entry is divided into *n* shares, such that any *k* of the shares can be used to reconstruct the entry, and one share is given to each of the *n* index servers. Each non-compromised index server authenticates the user and ensures she belongs to the right group, before giving her an element in response to her query. Even if an index is designed and implemented perfectly, the platform it resides upon is still vulnerable to compromise; one can bribe the sysadmin, measure radiation, take over root, etc. If one server is compromised, its secret shares do not provide enough information to decrypt any element-*k* secret shares from *k* different servers are required. Thus, at least *k* servers must authorize a user before she can decrypt any posting list element.

Each index server should be owned and managed by a different part of the enterprise. With this setup, no person in the enterprise can decrypt an index entry she is not authorized to see, unless she can find colluders from *k*-1 other factions, or compromise that many index servers. In other words, at least *k* index servers must be compromised to decrypt an index entry that should remain inaccessible. This minimizes the amount of trust that project groups must place in the *n* "centralized" index servers.

The *n* server boxes can provide extra resilience to attack by running *only* the index and no other services; providing only a narrow interface to the outside world (i.e., only insert, delete, and look up posting list elements); and using different hardware, operating systems, and systems software versions. Without this diversity, a successful attack on the underlying platform of one server may succeed against all *n* servers.

**Algorithm 1:** Compute *k*-out-of-*n* Secret Shares

**Input**: posting element $a_0$, prime *p*
**Output**: Encrypt a posting element (run by the document owner)
**begin**
1. Select random coefficients $a_1, \ldots, a_{k-1} \in Z_p$
2. Create a polynomial of degree *k*-1, e.g.,
   $f(x) = a_{k-1}x^{k-1} + \ldots + a_0$ *mod p*, where $a_0$ is the secret
3. Take the $x_i$-coordinate of each of the *n* servers, $x_i \in Z_p$
4. For each $x_i$ compute $y_i = f(x_i)$
5. Send the resulting $y_i$-coordinate to server *i*

### 3.5.1 Encryption of Posting Elements

Each posting list element in *ZERBER* is encrypted using Shamir's *k* out of *n* secret sharing scheme [Sha79]. According to this scheme, all the operations described later in this section are carried out in the finite field $Z_p$. The secret splitting algorithm starts by choosing a large prime number p, such that any element (secret) to be shared is in $Z_p$. In addition, each server *i* is assigned a unique random value $x_i$ in $Z_p$. We call this the *x*-coordinate of the server. These numbers *p* and $x_i$ are made public, so all users know them.

To index a document, its owner first parses the document and computes its elements. For each element $a_0$ (viewed as an integer), she generates a pseudo-random polynomial f of degree *k*-1 of the form $f(x) = a_{k-1}x^{k-1} + ... + a_0$ mod $p$, with coefficients $a_i$ (except $a_0$) randomly picked from field $Z_p$. The number of random bits required for each coefficient is the number of bits in $a_0$. The secret share given to the ith server is $f(x_i)$. *k* such shares are enough to reconstruct the polynomial. The encryption procedure is outlined in Algorithm 1.

The owner repeats this process to split all the elements for the document across the *n* servers. She also tells the servers who can access this document. Algorithm1 has a computational complexity of $O(nN)$, where *n* is the number of servers and *N* is the number of distinct terms in the document. For example, creation of the secret shares for one server for a document with 5,000 distinct terms requires only 33 msec on the platform described in Section 3.9.4.

To decrypt an element, a user must obtain *k* of its secret shares and determine the coefficients of the polynomial *f* by solving a system of *k* linear equations. This is presented in Algorithm 2.

The $k$ linear equations can be solved in $O(k^3)$ time with Gaussian elimination methods, which is affordable given that *k* is quite small in practice. For instance, on the platform described in Section 3.9.4, we can decrypt 700 elements in 1 msec on average.

Shamir's secret sharing scheme allows dynamic extension of the number n of servers without recalculating the existing secret shares, by just selecting additional points on the polynomial curve. Moreover, if an adversary learns some of the shares, proactive sharing

**Algorithm 2:** Compute Secret from *k* Shares

**Output**: Decrypt a Posting Element (performed by the querying user)

**begin**

1. Gather at least *k* shares of the element from any of the *n* servers
2. Recover $a_0$ by solving the following system of *k* linear equations: $y_i = a_{k-1}x_i^{k-1} + \ldots + a_0 \; mod \; p, i \in [1, k]$

techniques can be used to prevent the adversary from getting k shares [HJKY95]. With this technique, the shares are updated so that those she already knows become useless.
Just splitting the elements across *n* servers does not secure the documents against all the threats discussed above. For example, the number of elements in a term's posting list is its document frequency. Exploiting this, the adversary can learn about the presence (and even the exact number) of documents containing sensitive terms. The server cannot obfuscate the mapping of terms to posting lists on its own, as an adversary who takes over a server will be able to learn the mapping. Document owners could encrypt all terms before building posting elements for them; but then the other group members must know the encryption/decryption function, and we have already said that we want to free users from key management and from re-encrypting documents as groups shrink. Fortunately, there is a better way to increase security.

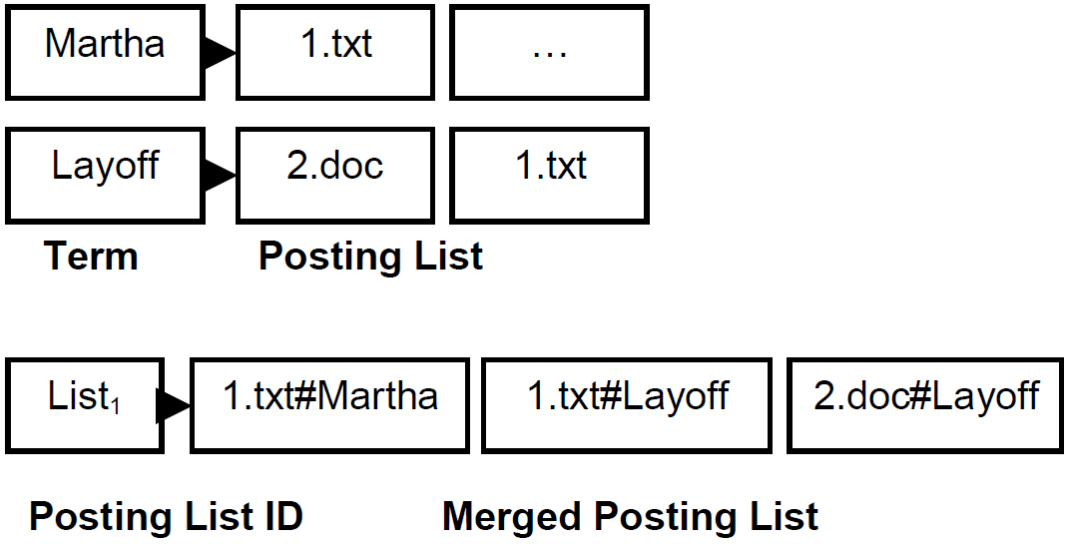



**Figure 3.1** Merged and Unmerged *Unencrypted* Posting Lists

### 3.5.2 *ZERBER* Merged Posting Lists

To prevent the adversary on a compromised index server from learning a term $t$'s document frequencies, we combine $t$'s posting list with the posting lists for several other terms, as shown in Figure 3.1 for an unencrypted posting list (term frequencies are not shown). An additional encoding is stored with each element to identify the term for that element. This encoding is encrypted along with the document ID and term frequency TF, using the *k* out of n secret sharing scheme described in Section 3.5.1. An unencrypted element hence contains three fields:

$$secret = [document_ID, term_ID, TF]$$

With this scheme, the adversary on a compromised index server only sees the combined posting list length of the merged terms and cannot determine an individual term's document frequencies. Further, the elements for a document are encrypted separately, so the adversary cannot determine which elements correspond to the same document. We formally analyze this below. Suppose the posting elements of terms $t_1, \ldots, t_n$ have been merged into one list. Upon examining an element in the merged list, an adversary can deduce only the following:

- Some document contains one of the terms $t_1, \ldots, t_n$.
- That same document can be read by users $u_1, \ldots, u_n$.

Although the adversary cannot determine the exact term in an element, she can make probabilistic claims about it using her background knowledge, e.g., general language statistics.

The probability $p_t$ of occurrence of a term *t* in the document corpus *D* is represented by its normalized document frequency:

$$P_t = n_d(t) / \sum_{t_i \in D} n_d(t_i), \tag{3.2}$$

where $n_d(t)$ is the number of documents in *D* containing term *t*. The probability of the posting element containing a particular term $t_u$, given that it is one of the terms in the set $S = t_{1,\ldots,t_n}$, is the ratio of the normalized frequency of that term to the sum of the normalized frequencies of all the terms in the set $S$:

$$p_{t_u} / \sum_{t_i \in D} p_{t_i}. \tag{3.3}$$

For a scheme to be *r*-confidential, this probability should not exceed *r* times the probability of tu occurring in a document according to the adversary's background knowledge:

$$\forall t_u \in S : (p_{t_u} / \sum_{t_i \in D} p_{t_i}) \leq r * p_{t_u}. \tag{3.4}$$

Intuitively, *r* measures the additional information the adversary can extract from the index, beyond her background knowledge. Thus, for the merging scheme to be *r*-confidential, we must have:

$$\sum_{t_i \in D} p_{t_i} \geq 1/r. \tag{3.5}$$

The *r*-confidentiality definition also encompasses the ability of the adversary to make claims about the absence of a term tu in a document. Given an element for a merged set of terms *S*, the probability that it is not an element for term $t \in S$ is $1 - (p_{t_u} / \sum_{t_i \in S} P_{t_i})$. This probability is smaller than the original probability $1 - p_{t_u}$ in the document corpus. Hence our objective is to find a merging strategy that satisfies Formula 3.5 and minimizes the expected query cost. In Section 3.5.3 we present several merging strategies that trade off between the degree of confidentiality *r* and the index size. Note that in the remainder of the chapter, *all posting lists are merged*.

### 3.5.3 Merging Heuristics

This section explores posting list merging heuristics that limit query processing costs while preserving *r*-confidentiality.

Suppose that all the posting lists are merged into $M$ lists $L_1, \ldots, L_M$. The total workload cost $Q$ for a set of queries is:

$$Q = \cong \sum_{L_i \in M} [length(L_i) \times (\sum_{j \in L_i} q_j)], \tag{3.6}$$

where $q_j$ is the query frequency of term j, and length($L_j$) is the number of elements in the merged posting list $L_i$. An efficient posting list merging heuristic must satisfy the *r*-constraint and minimize the expected workload cost. In other words, the optimization problem is to choose *M* lists such that *Q* is minimal and the *r*-confidentiality constraint on each list is satisfied. This problem can be shown to be NP-complete by reduction from the minimum sum of squares [CK97]. Thus we look for merging heuristics that are good in practice.

We first consider a uniform term probability distribution. It can be shown that the r (confidentiality) value in this case is equal to the number of merged posting lists. For example, if all terms are merged into one posting list, then *r* = 1 and no information about the keywords' document frequencies can be extracted from the index, beyond the adversary's background knowledge *B*. With two posting lists, *r* = 2 and we have half as much confidentiality.

In general, an index *I* with *M* posting lists increases the P(DocumentFreq = DF | *B*,*I*) probability by a factor of *M*.

The *document frequency* distribution in real documents is usually Zipfian, as in Figure 3.13. This suggests two strategies for merging. Given an $r$-value, we can merge terms

into posting lists so as to minimize $Q$ while maintaining $r$-confidentiality. Or, given a maximum value for $M$ (the number of merged posting lists), we can try to maximize $r$. We consider three such approaches.

During merging, we create a publicly available *mapping table* that maps a term to the ID of its posting list. The three algorithms differ in the way this table is initialized. All the algorithms base merging decisions on keywords' document frequencies. Though basing merging decisions on query term frequencies is more effective at reducing the total workload cost [MKGV07], use of query frequencies would violate our confidentiality goals.

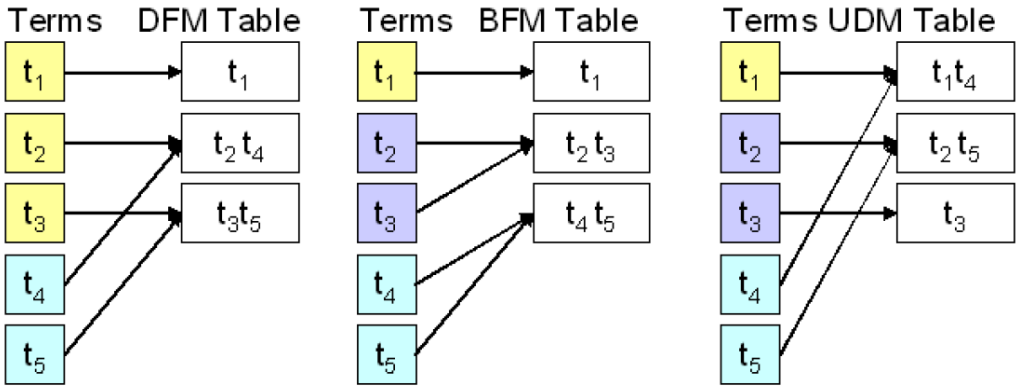



**Figure 3.2** Mapping Table Construction for Posting List Merging

### Depth First Merging (DFM)

DFM assigns the most frequent terms to separate posting lists, using a predetermined value of $M$ (the number of merged posting lists) as the table size (see Figure 3.2). This exploits the fact that frequently occurring terms are also queried more often. DFM fills the cells of the table from top to bottom with terms sorted by document frequency in rounds until the $r$-condition in each cell is satisfied. Algorithm 3 gives the DFM procedure.

### Breadth First Merging (BFM)

The Breadth First Merging heuristic (Algorithm 4) sorts terms on document frequency, then assigns successive terms to the first posting list until the r-condition is met. Then BFM moves to the second posting list, and so on until all terms are assigned to a list. BFM does not require us to predetermine $M$.

### Uniform Distribution Merging (UDM)

UDM is a variation on DFM in which terms are assigned to lists in rounds as in Algorithm 3, but without considering the resulting accumulated probability value. Once all terms are assigned to posting lists, we calculate the resulting confidentiality value as:

**Algorithm 3:** Depth First Merging

**Output**: Depth First Merge of Posting Lists (*terms* [], *M*, *r*)
**begin**
1. Calculate probability $p_t$ for each term *t* in terms, the array of all terms
2. Sort terms into descending order, based on $p_t$
3. Set the number of posting lists to *M*, and mark all of them as unfilled
4. **while** some term is not yet assigned to a posting list
5. go to the next posting list that is not marked as filled
6. **if** sum of the $p_t$ of terms assigned to this list exceeds 1/*r*
7. **then** mark the posting list as filled and go to the next list
8. **else** assign term *t* to this posting list

**Algorithm 4:** Breadth First Merging

**Output**: Breadth First Merge of Posting Lists (**terms** [], **r**)
**begin**
1. Calculate probability $p_t$ of each term *t* in terms (the array of all terms)
2. Sort *terms* into descending order, based on $p_t$
3. **while** more terms need to be assigned to a posting list
4. create a new empty posting list 5. **while** more terms need to be assigned and the sum of the $p_t$ of terms assigned to this posting list is less than 1/*r* 6. assign the next term to this posting list 7. **if** the r-condition is not satisfied for the last posting list
// there are not enough terms left to reach a good r-value for this list
8. **then** delete the last posting list and randomly distribute its terms among the other posting lists.

$$1/r = min_{L \in M}(\sum_{u \in L} P_{t_u}), \tag{3.7}$$

where *M* is the number of posting lists in the mapping table. DFM and UDM allow us to create an index with a predetermined number of posting lists, and compute the final confidentiality value after merging. BFM allows us to specify the confidentiality value, but the resulting number of posting lists is unknown until the merging is finished. We compare the query workload efficiency achieved by different merging heuristics in Section 3.9.6.

### Additional Hash-Based Merging

An adversary can inspect the mapping table and see whether a term is not included in any indexed site. Also, if a rare term is subsequently added to the mapping table, an adversary who has taken over a server can see which site requested the term's inclusion. To avoid this, we use hash-based merging for rare terms that do not significantly change the total

probability mass for a specific posting list. We consider a term rare if its original probability was below a certain cut-off threshold.

Hash-based merging works by assigning rare terms to posting lists using a public hash function, so that rare terms never appear in the mapping table. Therefore by inspecting the mapping table an adversary cannot find out whether a rare term appears at any indexed site or not. As the index does not contain any empty posting lists after its start-up period, an adversary cannot use emptiness of a posting list to check whether terms appear at any indexed site. Hash-based merging is also used to distribute the new terms randomly over the index.

## 3.6 Enabling Top-k Retrieval from an $r$-Confidential Index

In order to allow for efficient top-k retrieval and to support index updates in a collaborative environment, relevance score needs to be included in each posting element in a way the index server can access. In this section we provide a definition for scoring function, and discuss its factors relevant to confidential ranking.

### 3.6.1 Problem Statement for Top-k Retrieval from an r-Confidential Index

*ZERBER+R* targets the problem of providing confidential top-k retrieval from an outsourced inverted index. In the ideal case the server can order posting elements regarding to their relevance without violating the confidentiality of the index.

In an ***r***-confidential index the probability of a particular element in the posting list of being related to a specific term is $r$-bounded. A *scoring function* will order the terms according to their relevance and may increase the probability of a specific term being within particular positions or intervals of the posting list, given the adversary has knowledge of specific TF distribution as well as the public scoring function.

The ideal indexing scheme restricts an adversaries' ability to increase her available knowledge, even if she takes over an index server and can examine the ranking information of posting elements and the stream of incoming queries and updates. Assuming posting elements contain relevance scores and they are accessible by the index server, then the elements can be ordered (sorted) by the server in order to retrieve top-k results for a given query. In the following, we refer to a merged index whose posting elements contain relevance score values as an *ordered index*. An *ordered posting list* is a merged posting list in an *ordered index*.

The *ordered index* offers maximal protection in case relevance score values does not reveal any additional knowledge about the index content.The ideal ordered index will be unattainable in practice, but we can identify the factors which have impact on the confidentiality of an *ordered index* and quantify the degree of their impact.

### 3.6.2 Relevance Score Function for Top-k Retrieval

Powerful top-k server-side ranking techniques were introduced in the literature [Sin01]. The Vector Space Model is the most widely used model in IR for determining document relevance within a collection. In this model, a document ***d*** is represented as a vector, where each term is assigned a specific weight indicating the importance of the term in representing the semantics of the document. Two factors are of importance in the weight assignment: the normalized term frequency, which is the number of term occurrences in the document divided by the document length, and the inverse document frequency ( ***IDF***) which represents the query term selectivity. At the query time the relevance score ($rscore$ hereafter) of a document ***d*** for a query ***Q*** is computed using one of the standard techniques. As an example the computation using TFxIDF technique is performed as follows:

$$rscore(Q,d) = \sum_{q \in Q} \left( \frac{TF_q}{|d|} - IDF_q \right) = \sum_{q \in Q} \left( \frac{TF_q}{|d|} - log \frac{\sum_{t_i \in D} n_d(t_i)}{n_d(q)} \right), \tag{3.8}$$

where: $TF_q$ is the number of occurrences of the query term ***q*** in ***d***, $|d|$ is the document length measured in terms and $nd(t)$ is the number of documents containing term $t$. The term frequency based weighting factor is responsible for the correct ordering of documents with respect to a single query term. Normalization applied to the term frequency in Equation 3.8 prevents longer documents from being highly ranked just because of their length. IDF is responsible for making relevance scores of different terms comparable in case of multi-term queries. Unfortunately, IDF calculation requires knowledge of collection statistics, such as total number of documents as well as the number of documents containing particular query term. As the global index contains posting elements with different access rights, revealing such global IDF in the relevance score leaks critical statistical data about inaccessible documents [BC05]. Therefore, in *ZERBER+R* we focus on confidential top-k query processing for single-term queries. In this case IDF factor is constant and relevance score calculation can be calculated as:

$$rscore(q,d) = \frac{TF_q}{|d|}. \tag{3.9}$$

Results of a single-term query can be accurately ranked based only on the information contained in a single document using Equation 3.9. Processing of multi-term queries can then be performed by executing a number of single-term queries.

### 3.6.3 Outsourcing Relevance Scores for Top-k Retrieval

A naive approach to provide confidential ranking in an outsourced index would be to arrange posting elements in the posting list on the client side before outsourcing them. For instance, an inserting client could ensure that the most relevant top-k posting elements are contained in the head of the posting list. However, this approach is not suitable for the collaboration groups scenario, as the index contains posting elements with different access rights and therefore cannot be rearranged by a single user. Moreover, the whole process would need to be repeated whenever the document collection changes, rendering it impractical in case of frequent index updates.

To allow for top-k retrieval in an ordinary inverted index, relevance scores of posting elements are made available to the server. However, the scores increase the amount of information available to the server and thus decrease confidentiality provided by the outsourced inverted index, like it will be shown in the Section 3.6.4.

Scoring information is typically term specific and can allow adversary to reverse-engineer the terms.

In case of probabilistic-based index protection (e.g. [BBAV09]) an index contains a controlled amount of false positive elements. Adding relevance scores to the posting elements in this case would require generation of the realistic relevance scores for the false positive elements to prevent statistical attacks.

Sorting posting elements by their relevance score in a *ZERBER* 's merged posting list (Figure 3.3) may amplify the probability of a particular posting element to be related to a specific term, violating the *r*-confidentiality guarantees provided by the index. In the worst case it may allow an adversary to overcome the merging i.e. to find out which of the merged terms corresponds to a particular posting element. Consider a merged posting list containing terms "and" and "imClone". By sorting posting elements according to their term frequency, "extended version" would more probably appear in the tail of the list, as its term frequency is lower for all (or nearly all) documents.

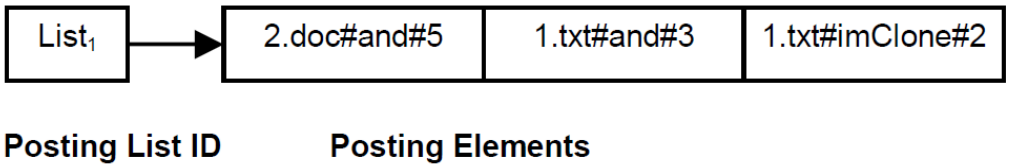
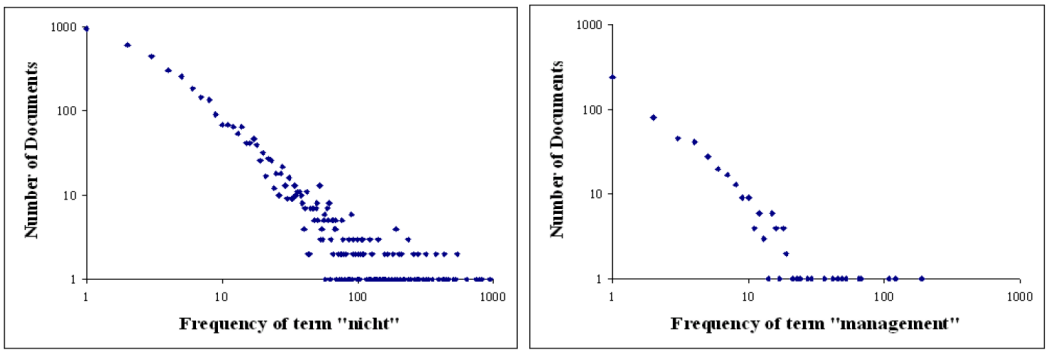



**Figure 3.3** Merged Posting Lists sorted by Term Frequency

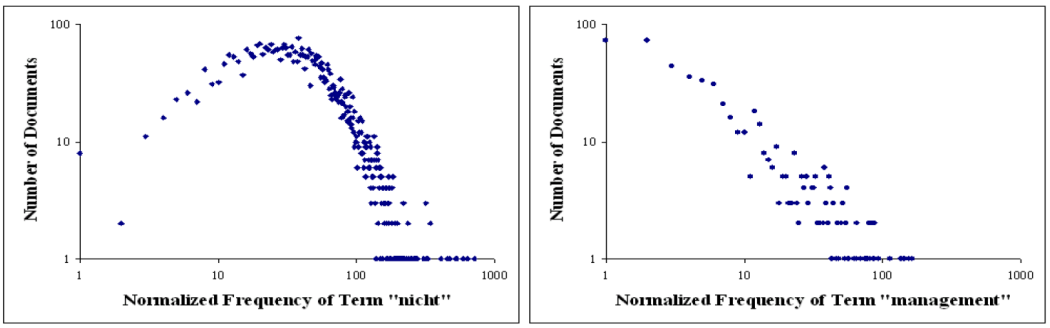
### 3.6.4 TF Distribution

Confidential document ranking is a challenging task. On the one hand, encrypting relevance scores to protect them on an untrusted index server does not allow the server to perform any ranking. On the other hand, if accessible to the index server, the term fre-




**(a)** **(b)**

**Figure 3.4** Log-Log Plot of Term Frequency Distributions




**(a)** **(b)**

**Figure 3.5** Log-Log Plot of Normalized Term Frequency Distributions

quency information required for the computation of relevance scores is term specific, and therefore can reveal the actual term in an index even if the term itself is encrypted, hence breaking confidentiality guarantees provided by the index.

Term frequency distribution among the documents in a collection follows a power law distribution (as shown by a log-log plot). Figure 3.4 shows the term frequency distributions for a frequent term (in German language) “nicht” and a less frequent term “management” in the test collection described in Section 3.9.1.

Terms can be differentiated by slope and value range of their TF distribution. State-of-the-art IR techniques mostly use term frequency normalized by document length [Sin01] in order to avoid that large documents are ranked higher simply because they contain more terms.

Normalized TF represents the frequency fraction of a term in a document, which depends on the document topic, assigning higher ranks to the specific terms. Normalized TF distributions, as extracted from our study on our test collections (Figure 3.5 illustrates an example for the aforementioned terms), are not power law but still term specific. An attacker knowing these typical term distribution patterns could derive the indexed terms from

the TF distribution found in the inverted index.

Since most state-of-the-art IR techniques use normalized TF for ranking [Sin01] our aim is to store the relevance scores accessible to the untrusted server while hiding the term specificity of the normalized TF distribution, therefore making posting elements representing different terms indistinguishable in a merged posting list. At the same time we need to preserve the relevance order of posting elements in a merged list to allow for top-k retrieval.

### 3.6.5 Confidentiality Preservation for Top-k Retrieval

Server side top-k processing requires the server to access relevance scores of each posting element. However, as explained before, the distribution of posting elements within a merged posting list according to their plain relevance scores can reveal the corresponding term of the posting element (as shown on Figure 3.3). Since the scoring function presented in Equation 3.9 is monotonic, it is possible to make transformations to such function without affecting the ranking process as long as the ordering within posting elements representing each particular term is preserved. This section introduces the requirements for building a *relevance score transformation function* ( ***RSTF*** hereafter), which makes relevance score distributions of different terms indistinguishable. Specifically, it uniformly distributes posting elements representing different terms within the merged posting list while the order of posting elements related to each single term remains unaffected. This transformation function assigns a *transformed relevance score* ( ***TRS*** for short) to each posting element, therefore replacing the original relevance score. To preserve confidentiality of the index a posting element related to any term $t$ in an ordered posting list should have equal probability to obtain a given ***TRS*** score.

***RSTF*** has to fulfil the following properties:

- ***RSTF*** maps the relevance scores of different terms to a range ***R***, which will be the same for all ***RSTF***s.
- ***RSTF*** uniformly distributes the TRS values over ***R***.
- ***RSTF*** preserves the order of the relevance score values.

In the Section 3.7 we propose a heuristic for the construction a ***RSTF*** for arbitrary term independent of its score distribution.

## 3.7 Design Refinement for Top-k Retrieval in *ZERBER+R*

Processing in *ZERBER+R* can be split in two phases: an offline pre-computing phase performed once at the time of index initialization and an online insertion and query phase. In the pre-computing phase, *ZERBER+R* initializes and publishes the ***RSTF*** for each term in the training document set, such that in the online insertion phase this function can be used by an inserting client. To index a document, its owner extracts the documents' terms, builds their elements, encrypts them, calculates ***TRS*** values, and sends encrypted posting

elements to the server along with the IDs of the merged posting list that the new element belongs to, the documents' group and the ***TRS*** value. The index server authenticates the user, checks his group membership and accepts the update if appropriate. Finally, the server inserts posting elements into the specified merged posting list. Upon query the server has access to the ***TRS*** values and can identify the top-k most relevant query answers.

### 3.7.1 *RSTF* Construction

We define the target ***RSTF*** range to map the relevance score input values (calculated using Equation 3.9) as $R = [a_1, a_2]$.

In order to explain our approach, we first consider the case where the input values are already uniformly distributed over some range $[b_1, b_2]\%$. In this case ***RSTF*** is a straight line with slope $(b_1 - b_2)/(a_1 - a_2)$ and offset -$b1$, and is a projection of $[b_1, b_2]$ on ***R***.

For example, Figure 3.6 shows how $f(x) = 2.5 * x - 1.25$ maps a range $[0.5, 0.9]$ to $[0, 1]$. The slope of the projection function is responsible for the scale of the input range in the output range. Given an input range, the greater the slope of the projection function, the wider is the output range. Unfortunately, the relevance score distribution in the document set is not uniform and a linear projection does not change the distribution of the input relevance score values. To uniformly distribute the input values over the output range, the slope of the ***RSTF*** at each particular point has to reflect the probability density of the relevance score values at this point, i.e. it should create a wider output range in the more crowded areas. Thus the probability density function of the relevance score values is a derivative of the ***RSTF*** and respectively the ***RSTF*** is an integral over the probability density function of the relevance score values.

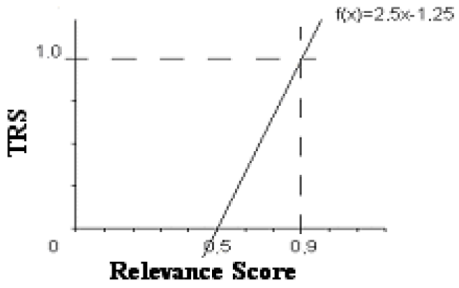



**Figure 3.6** Relevance Score Transformation: Linear Projection

The values of the integral over a probability density function are always contained within the common range $[0, 1]$. Moreover, an integral is monotonically increasing, which preserves the ordering of the input values. Thus an ***RSTF*** created in this way would possess the first and third required properties discussed above.

In the following we model the probability density of the relevance score distributions and compute a ***RSTF*** that approximates the relevance score distribution to a uniform distribution.

### 3.7.2 Modelling Relevance Score Distribution for Top-k Retrieval in *ZERBER+R*

To model the relevance score distribution, *ZERBER+R* requires as input a training set of documents. This set must be a representative sample of the corpus, such that the distribution of the relevance scores will hold for the whole corpus as well. From the training set *ZERBER+R* extracts the relevance scores for each term-document pair. Terms found later, which were not contained in the training set are assumed to be rare and can therefore be assigned a random ***TRS***. We base our model on the central limit theorem [Ric01] stating that the sampling distribution of the sample mean is approximately normal, even if the distribution of the population from which the sample is taken is not normal.

We consider each relevance score value from the training set to be a sample mean of the relevance score. The continuous probability density function of the *normal distribution* is the Gaussian function. We model the probability distribution of the relevance score values around each sample mean as a Gaussian curve. We take the sum of the Gaussian curves over all samples as an approximation of the relevance score distribution for a given term over the whole document corpus. Thereby we make use of the fact that a probability density function can be arbitrarily closely approximated by a weighted sum of Gaussian curves [AS72].

A Gaussian function can be defined by two parameters, location and *scale*: the mean ("average", $\mu$) and variance (standard deviation squared) $\sigma^2$, respectively. The probability density of a term $t$ over the whole document corpus given $N$ training points is calculated as:

$$f_{\vec{\mu},\sigma} = \frac{1}{N}\sum_{i=0}^{N}\left(\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-\mu_i)^2}{2\sigma^2}}\right), \tag{3.10}$$

where $\mu_i$ is the $i^th$ value from the training set and $\sigma$ is the scale of the Gaussian function.

A more frequent term results in several training values. In this case the sum of the Gaussian bells, one for each input value, will reflect the probability distribution over the input interval. The probability of unseen values being in particular region is reflected through the density of training points in that region.

Figure 3.7 shows the sum of the probability density functions over five input values. The $X$-Axis shows the relevance score of a training value. The $Y$-Axis shows the probability density. Solid lines represent probability density of each training value. The dashed line represents the probability density accumulated using several training values.

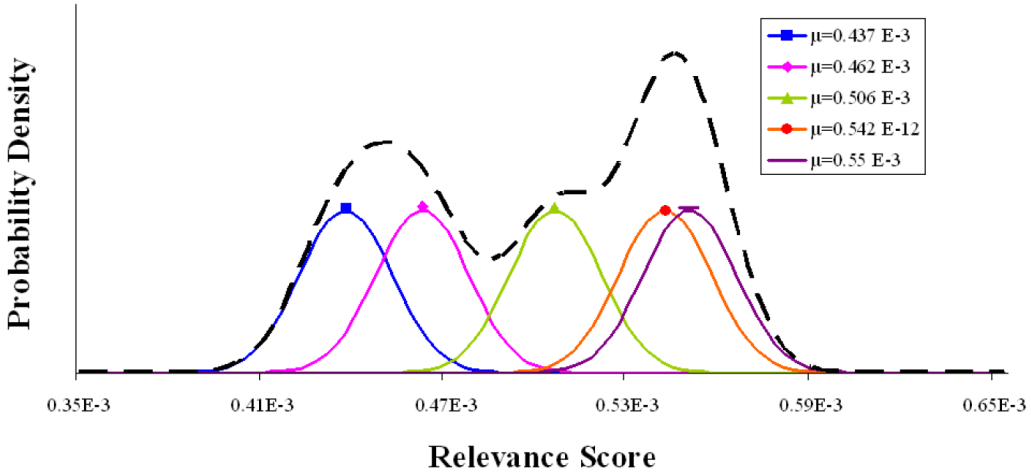



**Figure 3.7** Probability Distribution from 5 Training Values

### 3.7.3 Calculating the *RSTF*

***RSTF***(x) is an integral of the probability distribution of the input training values within the range $[-\infty, x]$.

Given *N* training points ***RSTF***(x) can be calculated as:

$$f_{\vec{\mu},\sigma} = \sum_{i=0}^{N} \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{(u-\mu_i)^2}{2\sigma^2}} du, \tag{3.11}$$

where x is a relevance score to be transformed, $\mu_i$ is the $i^th$ value from the training set, $N$ is a number of the values in the training set and $\sigma$ is the scale of the Gaussian function.

Therefore, ***RSTF***(x) represents a projection of the relevance score distribution that approximates the relevance score distribution into a uniform distribution over the range $[0, 1]$. An integral of the Gaussian function within the range $[-\infty, x]$ can be estimated with the standard error function:

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(u-\mu)^2}{2\sigma^2}} du \approx \frac{1}{1+e^{-\sigma(x+\mu)}}. \tag{3.12}$$

Thus the ***RSTF*** can be calculated as:

$$RSTF_{\vec{\mu},\sigma}(x) \approx \frac{1}{N} \sum_{i=0}^{N} \left[ frac1(1+e^{-\sigma(x+\mu_i)}) \right] \tag{3.13}$$

Using training set from our datasets (see Section 3.9.1), Figure 3.8 illustrates an example RSTF function for the German term "Vergütung" (reimbursement). The $X$-Axis shows the input relevance score, the $Y$-Axis illustrates its output TRS value computed using Equation 3.13.

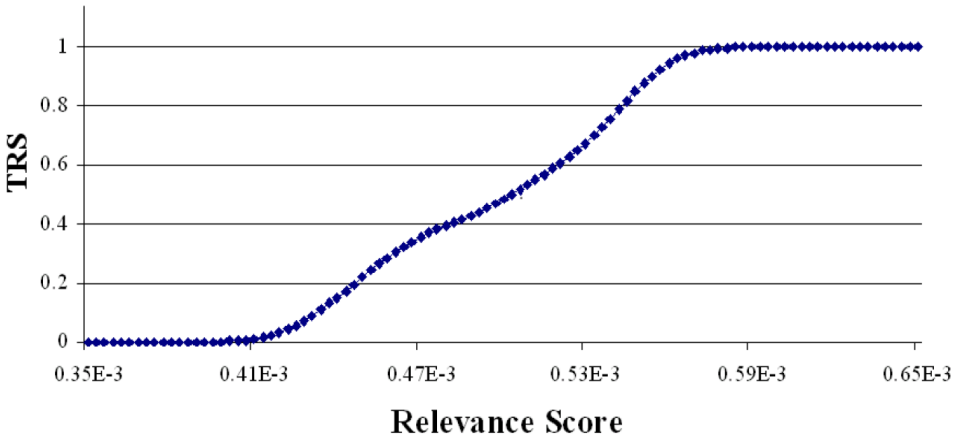



**Figure 3.8** An Example RSTF Function for the German Term "Vergütung"

### 3.7.4 $\sigma$ Parameter Selection

In order to ensure the uniformness of the ***TRS*** distribution on the whole corpus it is needed to ensure the correct prediction of the relevance score distribution during the ***RSTF*** computation. Correctness of the prediction depends on the representativeness of the training set as well as on the correct $\sigma$ selection.

The $\sigma$ parameter represents the scale of the Gaussian function reflecting the generality of the ***RSTF***. $\sigma$ is responsible for the learning/memorizing effect. Smaller $\sigma$ means a broader Gaussian bell and thus a more general prediction. Higher $\sigma$ value means a narrower bell, meaning a less general function which represents the particular training point (also known as overfitting).

To select an optimal $\sigma$ value we use the cross-validation technique to measure the uniformness of the ***TRS*** distribution in a control set. As a basic measurement for uniformness we compute the variance in the distribution of the ***TRS*** values of a particular term in the control set with respect to a uniform distribution, that is, how far the ***TRS*** distribution is from a uniform distribution.

Figure 3.9 presents the ***TRS*** variance in the control set dependent on the $\sigma$ value. The $X$-Axis shows the $\sigma$ value, the $Y$-Axis shows the variance within ***TRS*** values in the control set. At first, the ***TRS*** values are distributed more uniformly with an increasing $\sigma$. However, after reaching the minimum (an optimal $\sigma$), the overfitting effect appears and the uniformness is destroyed. An optimal $\sigma$ for a particular term is the infimum of the variance function. As Figure 3.9 depicts, a good selection of $\sigma$ provides a variance of smaller than 0.00002 (standard deviation of 0.0044, that is, 0.44% of the range $[0, 1]$).

The process of cross-validation is time consuming. Finding a method for directly determining an optimal $\sigma$ value is an interesting direction for future research.

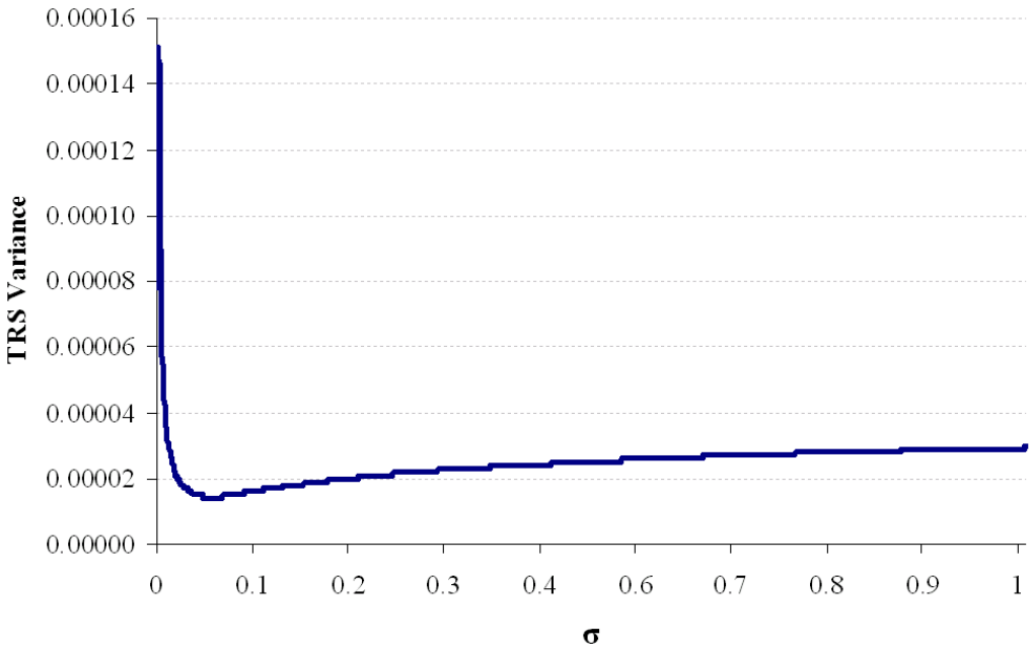



**Figure 3.9** TRS Variance Depending on the Selected $\sigma$ Parameter

## 3.8 Using *ZERBER* and *ZERBER+R*

In this section, we describe access control, indexing and querying in *ZERBER* and *ZERBER+R* .

### 3.8.1 Access Control

To answer user queries, each *ZERBER* index server enforces access control on its posting elements. Upon query, the server authenticates the user and determines the groups the user belongs to. For this purpose, each index server records which users belong to each group, and which posting elements are accessible to each group. We do not consider this information sensitive here, given that we can bound the ability of the adversary to extract document contents from the inverted index. (If this information is sensitive, it can be blinded and/or stored on a separate server.)

The structure of a *ZERBER* index server is shown in Figure 3.10. The architecture supports dynamic changes in group membership. To add or remove a user from a group, only the table containing the user-group metadata needs to be updated. (The metadata that controls who can update a group is not shown in Figure 3.10, and that process is outside the scope of this thesis.).

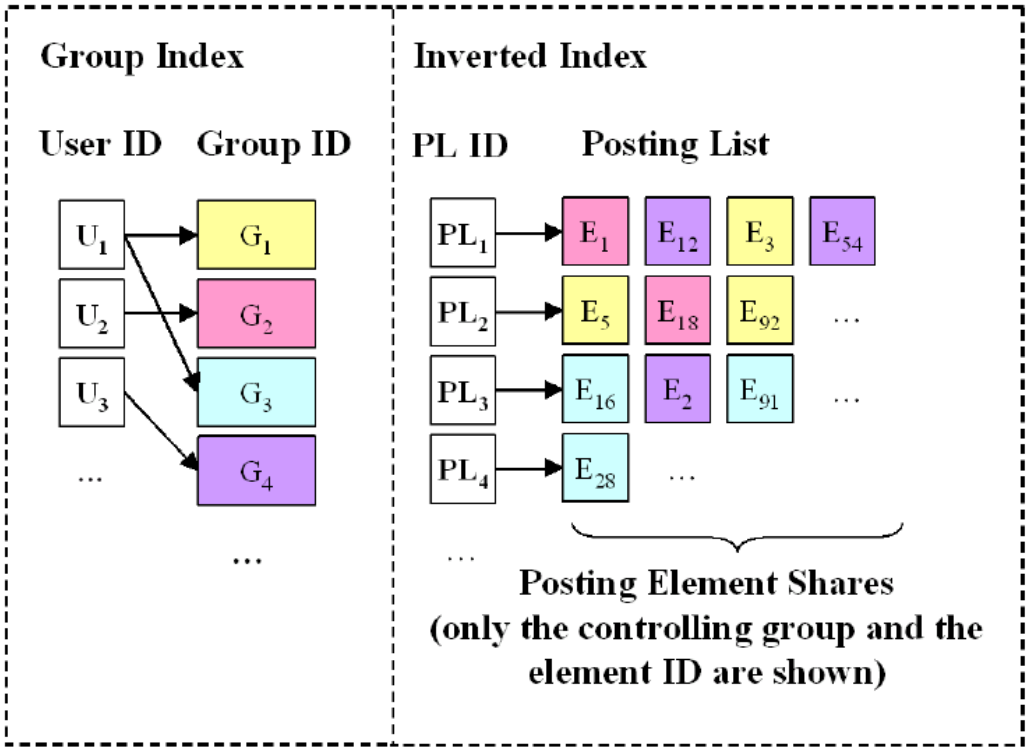



**Figure 3.10** *ZERBER* Index Server

### 3.8.2 Indexing a Document

To index a document, its owner extracts the document's terms, builds their elements, encrypts them with Algorithm 1, gives each one an ID that will be globally unique within its posting list, and sends an element share to each of the $n$ servers, along with the IDs of the merged posting list that the new element belongs to, the document's group, and the element ID. The index server authenticates the user, checks his group membership and accepts the update if appropriate. The element IDs help an index recover after failure, and tell users which shares to merge together.

Index updates in *ZERBER* can be performed in batches that insert or delete posting elements for multiple documents. Batching can reduce index freshness[2], but also reduces the average network and disk overhead per update (each append to a posting list incurs a random I/O). If Alice has compromised an index server, then batching also reduces the information she gets by watching updates. For example, if Bob inserts into the posting lists for Martha, P and Ralph, Q, Alice can guess that "Martha" and "Ralph" occur in the same document (if P and Q are unlikely to occur with the other terms). Inserting elements from several documents in one batch makes it hard for Alice to guess which terms co-occur. Bob can also pool his updates with other people's, or send his through a MIX network, to give

[2]Delaying index updates for some of the terms in an updated document does not affect *document freshness*: only the most recent copy of the document on a site will ever be retrieved.

himself anonymity and improve index freshness. If the user trusts that no index servers are compromised, then the indexes can be updated whenever a shared document changes, rather than in batches.

Batch size, frequency, and other batch parameters can be tuned by each document owner to trade off security and index freshness, based on the elapsed time since the last batch, term frequency, vocabulary changes, or query volume [BNST05]. *ZERBER* runs a client program at the document owner that tracks local changes and performs only the necessary updates at the central indexes.

### 3.8.3 Query Answering using *ZERBER*

To execute a keyword query, the user first authenticates herself to *k* or more index servers. The index servers rely on an enterprisewide authentication service, such as one normally finds in today's large enterprises; Kerberos or any other approach to authentication in distributed systems can be adopted here. The index servers determine her groups by consulting the group table. This can be done in $O(N)$ time, where *N* is the number of groups. The user tells the servers which posting lists she wants (she does not divulge which *terms* she is querying), and each server returns a share of each posting element that she is allowed to access: $PL\_ID$, $[\{\{g\_id_1, e(doc_1, term_1, tf_1)\}, \ldots, \{g\_id_j, e(doc_j, term_j, tf_j)\}]$, where PL_ID is a posting list ID; *e* returns a secret share; and $g\_id_i$, $doc_i$, $term_i$, and $tf_i$ are the global element ID, document ID, term ID, and term frequency of the $i^th$ element, respectively. The document ID must identify both the machine on which the document is hosted and the document within that machine. Based on the global posting element ID, the client determines the corresponding element shares from the different servers, decrypts the element with Algorithm 2, then filters out false positives, i.e., elements for terms not queried. The client then ranks the results using any modern document ranking technique [Sin01].

*ZERBER* uses client-side ranking with personalized collection statistics obtained from the set of all documents accessible to the user. We use a modification of Fagin's Threshold Algorithm [CK97] that lets one obtain the top-k ranked results in time

$$O(PLLength^{\frac{QT-1}{PLLength}} * K^{\frac{1}{QT}})$$

, where *PLLength* is the length of the posting list and QT is the number of terms in the query. Search engine results usually include a document ID and also a small portion of the document content surrounding the query term. Such context information cannot be stored on the index servers due to security and space concerns. *ZERBER* clients request snippets from the peers hosting the top-k documents before presenting the search results to the user. Finally, the user chooses among the top-k documents and clicks on those that are to be fetched from the hosting peers. Algorithm 3.1 summarizes query processing in *ZERBER* .

**Listing 3.1Query Processing in *ZERBER***

```
begin func main( )
Server Servers[]:= getAvailableZerberServers();
RankedPostingElements[]:=
User.search(auth_token, query, Servers);
//Display top-k elements
for i:=1 to K begin
url:= toUrl(RankedPostingElements[i]);
snippet:= getSnippet(url);
print snippet+url;
end for
end func
begin class User
//Process user query on the client side
begin func search(query, Servers[])
//PL_IDs: merged posting list IDs to retrieve
PL_IDs[]:=mapQueryTerms(query);
for i:=1 to Servers.length begin
serverAnswers[i]:= Servers[i].
getPostingLists(PL_IDs);
end for
PlainList[]:=decodeShamirsScheme(serverAnswers);
//Remove false positive posting elements
PostingElements[]:=
filterElements(PlainList, query);
return rank(PostingElements);
end func
end class
begin class Server
//Retrieve the posting lists on the server side
begin func getPostingLists(auth_token, PL_IDs)
userID:=authenticateUser(auth_token);
if userID=false, then
return error;
end if
//Select group indexes accessible to the user
GroupIDs[]:=DB.execute
("SELECT groupID FROM groups WHERE userID = "+userID);
//Retrieve accessible parts of requested posting lists
PostingLists[]:=
DB.loadPostingLists(GroupIDs, PL_IDs);
return PostingLists;
end func
end class
```

### 3.8.4 Query Answering using *ZERBER+R*

To execute a keyword query, the user first authenticates herself to an index server and tells the server which posting list she wants to query as well as the k-value. The index server determines users' access rights and returns a number of highly ranked elements from the requested list. The client decrypts posting elements and filters out elements for terms not queried. If the client did not obtain the desired number of elements belonging to the queried term it sends a follow-up request to the server. Dependent on the response size, the number of follow-up requests for rare and frequent terms can differ, leaking information to the adversary. In the following we discuss heuristics reducing the number of follow-up requests.

Suppose that all the posting lists are merged into *M* lists $L_1, \ldots, L_M$.

The total workload cost ***Q*** for a set of queries for retrieving the top- ***k*** elements can be calculated as:

$$Q \cong \sum_{L_i \in M} \left[ N(L_i) \times \left( \sum_{j \in L_i} q_j \right) \right], \tag{3.14}$$

where $q_j$ is the query frequency of term j, and $N(L_j)$ is the number of elements to be retrieved from the merged posting list $L_i$.

Posting elements in each merged list are sorted based on their ***TRS***. Following the uniform distribution, the first position $\mathbf{pos}_1$ of the term ***t*** in the list can be approximated as:

$$pos_1(t) \leq \frac{1}{p_t} = \frac{\sum_{t_i \in L} n_d(t_i)}{n_d(t)}, \tag{3.15}$$

where $\mathbf{p}_t$ is the probability of term ***t*** and $n_d(t)$ is the document frequency of the term ***t*** in a merged list ***L***. In order to obtain the top- ***k*** elements of the term ***t*** the total number ***N*** of elements to be retrieved from the list ***L*** is:

$$N(L) = k \cdot pos_1(t) = \frac{k}{p_t} = k \left( \frac{\sum_{t_i \in L} n_d(t_i)}{n_d(t)} \right), \tag{3.16}$$

where $n_d(t)$ is the document frequency of the term ***t*** in a merged list ***L***. In order to retrieve the top-k elements from the merged list without knowing which particular term is queried, the number of retrieved elements should be sufficiently large to include the top-k elements for all merged terms in the posting list ***L***.

However, this is impractical in case a posting list contains very rare terms. For instance, for a list with terms for which $n_d(t) = 1$ (term $t$ is contained only in one document), the whole posting list will be returned in response to the query. Thus we need a heuristic to determine the query response size that reduces the total workload cost. This heuristic

should minimize the used bandwidth while including the top-k answers in the first response to most of the received queries.

From query load logs described in Section 3.9.1 we know that the most frequent queries constitute nearly the whole query workload (Figure 3.12). Thus to reduce the total workload cost, the query answering heuristic should provide high efficiency for the most frequent queries. Due to confidentiality concerns we can not use query frequency directly. However, document frequencies and query frequencies are correlated, though some frequent terms are rarely queried (e.g., "although") [MHW06]. To provide efficient query answering for the most frequent queries while reducing the bandwidth, *ZERBER+R* puts a bound b on the initial response size, such that only sufficiently frequent terms with probability $p_t * b >= 1$ are necessarily returned within the first query response. We discuss the choice of the initial response size in Section 3.9.7.

In case a rare term $t$ with probability $p_t * b < 1$ is requested the user sending the query might need to issue several follow-up requests to obtain the top-k results. This can give an adversary the possibility to infer a rare term has been requested, therefore allowing her to distinguish between two merged terms in case one is frequent and the other rare. In order to prevent this situation, *ZERBER+R* makes use of the BFM index (using the Breadth First Merging of posting lists) described in Section 3.5.3, which ensures that the terms merged in a posting list have similar frequency distributions. Therefore even if retrieving top-k results from a merged list containing rare terms could require a number of follow-up requests, this number will be similar for all terms contained in the list, avoiding that an adversary is able to infer any additional information. An adversary could also try to estimate the position of a queried term in the posting list based on the querying behaviour of the user. *ZERBER+R* reduces the information leakage in this case by progressively increasing response size for follow-up requests. Assuming that the user can process at least the amount of data she already obtained *ZERBER+R* doubles response size for each follow-up request until the user is satisfied with the result or obtains the whole list.

## 3.9 Evaluation

In this section, we discuss *ZERBER* 's and *ZERBER+R* security guarantees and evaluate their storage requirements, query performance, and network bandwidth usage compared with an ordinary inverted index, using a real-world web search query log. We also evaluate the effectiveness of the proposed merging heuristics.

### 3.9.1 Experimental Setup

This section provides details about our documents and workload. We used two datasets, from the Stud IP Learning Management System and Open Directory Project crawl data. We used a web search engine query log as the workload, computed as follows. The time to scan a posting list is the sum of the seek time (to position the disk head at the start of the posting list) and the transfer time (the time to read the posting list). The total seek time for

a given query workload is a constant, independent of the merging heuristic. The transfer time for a posting list is proportional to its length. Equation 3.6 is the sum of the posting list lengths, weighted by their query frequencies. Thus the total transfer time (and hence the total workload cost, since the seek time is constant) is proportional to Equation 3.6, which we use as the workload cost in the experiments that follow. All experiments ran on a 2-processor 2.0 GHz Intel CPU T2500 with 2 GB RAM.

**Algorithm 5:** HierarchyConstruction

**input** : tables $T$, ontology $o$
**output**: hierarchy of concepts $H$

**begin**
establish child relations, set depth
$C \longleftarrow$ `BottomUp`$(T, o)$
establish parent relations
$H \longleftarrow$ `TopDown`$(C)$

### Stud IP Data

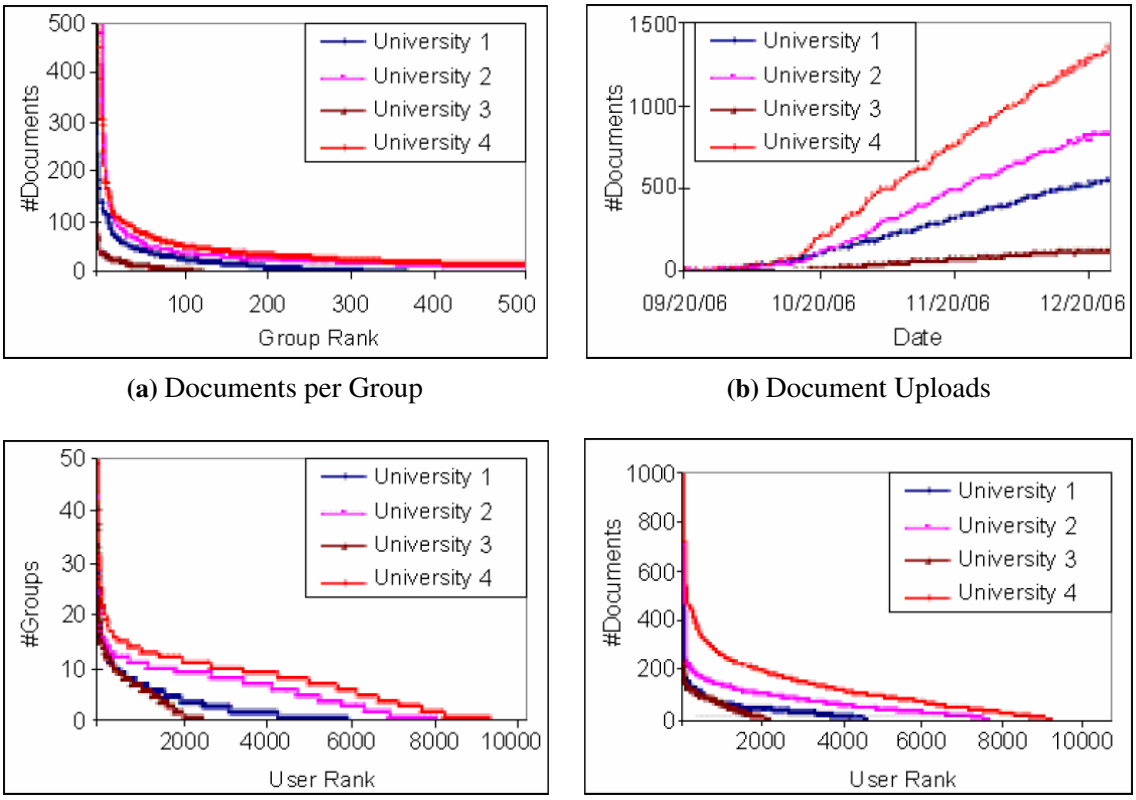
The Stud IP Learning Management System [Stu] allows sharing of access-controlled materials within groups of students and teachers. We had access to the Stud IP documents at four universities. Figure 3.11 provides insights into these datasets. For example, the installation at "University 1" has over 3,300 courses and 6,000 registered students. Most users belong to at most 20 groups and can access fewer than 200 documents. The amount of material stored for each course increases uniformly during the semester (Figure 3.11). A mid-semester snapshot used for our experiments contained 8,500 documents with 570,000 terms. At the time of writing, Stud IP did not provide full-text search capabilities and thus we did not have an associated query log.

### ODP Data

We used a collection from the Open Directory Project (a human edited directory of the Web) crawled in 2005, with 237,000 documents and 987,700 distinct terms. The crawler's strategy was to find pages on a variety of topics [KCN06], such that 100 topics were randomly selected; we used the set of documents on one topic as the set of documents of one group.

### Web Search Engine Query Log

Our query log has 7 million queries and 135,000 distinct query terms. Figure 3.12 shows the correlation of the query frequency and the corresponding cumulative query workload (computed using Equation 3.6).




**(a)** Documents per Group

**(b)** Document Uploads

**(c)** Users per Group

**(d)** Documents Accessible per User

**Figure 3.11** Statistical Profile of the Stud IP Dataset

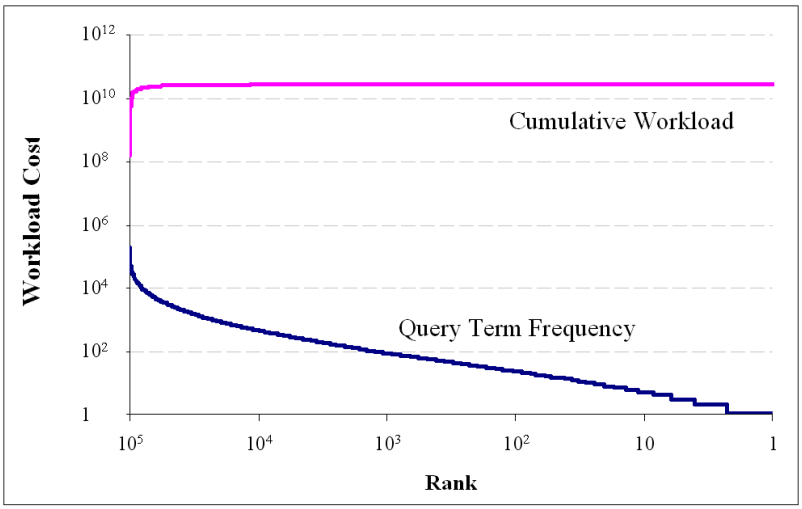



**Figure 3.12** Cumulative Query Workload Cost for a Search Engine Query Log

The log-scale $X$-Axis shows the query terms in decreasing order of frequency. The most frequent queries constitute nearly the whole query workload. Thus to reduce the total workload cost, the merging heuristic should provide high efficiency for the most frequent queries. As explained earlier, confidentiality concerns require us to base merging decisions on document frequencies rather than query frequencies. These are correlated, though some frequent terms are rarely queried (e.g., "although"). The $X$-Axis in Figure 3.12 lists the terms ordered from most to least popular.

### 3.9.2 Security Guarantees

If an adversary Alice compromises an index server, she can attempt to amplify her knowledge in many ways. For example, she can issue arbitrary queries and updates and scrutinize the responses. With *ZERBER* , she cannot learn anything this way that violates the principle of *r*-confidentiality. If Alice takes over a server, she can learn who sends each new query/update to that server; to prevent this, one would need to extend *ZERBER* to include only *opaque* user IDs in requests and in the user-group mapping.

Alice can see which posting lists each user queries at her compromised server, and see the (opaque) answers. We expect that in practice, the posting list query frequencies she learns will be consistent with her background knowledge. In other words, she will not be able to use this information to violate *r*-confidentiality by improving her guesses about which terms are in each document.

On her compromised server, Alice can see which posting lists are affected by each new update. By monitoring the sequence of updates, Alice can guess that a set of new posting elements refers to the same document. This lets Alice make correlation attacks. For example, suppose that an email contains terms from posting lists $p_1$ and $p_2$ and that the only two terms from $p_1$ and $p_2$ that are likely to co-occur are $t_1$ and $t_2$, respectively. Then Alice

can guess that the email contains these two terms, even though the posting elements are encrypted. Thus Alice may be able to violate $r$-confidentiality for newly created documents, though in general she cannot be sure of the exact contents of an element. However, Alice cannot violate $r$-confidentiality for documents committed before she compromised the server, as she cannot tell which preexisting posting elements refer to the same document.

Alice can collude with others to jointly take over multiple index servers, and pool the resulting knowledge. If the colluders take over fewer than $k$ servers, they will not be able to violate $r$-confidentiality for documents committed before the attack.

In *ZERBER+R* , Alice can also examine the unencrypted ***TRS*** values attached to the posting elements in an index but *ZERBER+R* should ensure that she can not learn anything in order to preserve our concept of ***r***-confidentiality. To achieve maximal security effectiveness the ***RSTF*** needs to distribute the relevance scores of each term from the real dataset equally well as it is achieved in the training set values. Otherwise, term specific distribution patterns would be introduced allowing Alice binding posting elements within specific posting list areas to their corresponding terms with higher probability. In case the document training set is a representative sample of the corpus and if value is selected properly, all terms will have equal probability to obtain a given ***TRS*** value, such that using ***TRS*** does not introduce any additional attack possibilities. Alice can also observe user queries and query responses. As discussed in Section 3.8.4 it is impractical to include top-k results into the initial query response for all possible terms[3]. Thus for rare terms which top-k elements are not contained in the initial response Alice could observe an increased number of follow-up requests and conclude that a rare term has been requested. However, as a *ZERBER+R* BFM index contains terms of similar probability inside of a posting list, the number of requests observed by Alice will not differ for the terms contained in one merged list, such that ***r***-confidentiality of the index will not be affected. Alice could also try to estimate a position of such a rare (unknown) term based on the number of returned posting elements. Thereby growing response size for follow-up requests will reduce the total number of such requests and introduce an increasing degree of uncertainty in Alices' claims regarding the position of the (unknown) rare queried term.

### 3.9.3 Storage Overhead

The number of posting elements that *ZERBER* maintains per index server is the same as in any conventional inverted index. However, *ZERBER* posting elements include additional fields to identify the term in the merged set and the global element ID, which increases element size by about 50%. Encryption under Shamir's $k$-out-of-$n$ scheme does not change the element size. Hence, each *ZERBER* index server uses about 50% more space than an ordinary inverted index. Since *ZERBER* replicates the index on n servers, the total index space required is $1.5n$ times more than for an ordinary inverted index.

[3] In this work we focus on a fixed result set size in the initial response to a query. However, we leave for further work optimizations where this size could vary depending on the frequency of the terms of each merged posting list.

Each document server maintains an inverted index (also useful for local search) of its local shared documents, to support efficient updates. This index includes the global ID of each element.

To allow for top-k processing an ordinary inverted index typically contains relevance score information attached to each posting element. *ZERBER+R* attaches a transformed relevance score ***TRS*** to each posting element, which is sufficient for effective posting element ranking on the server side. Thus it does not introduce any storage overhead compared with an ordinary inverted index.

### 3.9.4 Network Bandwidth

*Insertion and deletion*. To index a document, the owner sends its elements to $n$ servers, so *ZERBER* uses $1.5n$ times more network bandwidth for this operation than an ordinary inverted index does.

Deletion from an ordinary inverted index can be implemented by sending the ID of the document to be deleted to the index server. *ZERBER* elements (and hence the document ID field) are encrypted, so the server cannot determine which posting elements have the same document ID. To delete a document, its owner must delete each element separately. The document deletion network cost is thus the same as its insertion cost.

*Query processing*. *ZERBER* query processing is performed in two steps: (1) the client sends the query to $k$ index servers and retrieves the IDs of matching documents, and (2) the client requests the snippets for the top-k documents from their owners. Step (1) queries $k$ index servers, requiring $k$ times as much bandwidth as a traditional lookup. Moreover, a traditional inverted index might return only top-k search results, but *ZERBER* must return all of the elements accessible to the user. On the other hand, *ZERBER* uses no additional bandwidth to retrieve lower-ranked search results, while traditional inverted indexes do revisit the server for each page of results.

For our calculations, we assume the following intranet setup: users connect over a 55 Mb/s wireless LAN, while servers use 100 Mb/s LAN connections. We use 2-out-of-3 secret sharing. The document snippets arrive in XML format.

We use a real-world query workload and the Open Directory Project (ODP) data described in Section 3.9.1. For our experiments we assume the worst case: the user has access to all 100 document collections in the ODP data. In this workload, about 2700 elements are returned from the ODP index per query term on average. Assuming that each posting element is encoded using 64 bits, this is approximately 170 Kb (21.5 KB) per query term response. The queries in the workload contain on average 2.45 terms, which allows for execution of up to 35 queries/second per user and about 200 queries/second answered by each server on average. We expect that the number of queries answered by a server can be increased in an enterprise setting as users typically belong to a smaller number of groups (see Section 3.11). On average, each snippet contains about 250 B including XML formatting, which yields 2.5 KB for the top-10 snippets. Thus average total response size for the top-10 results is 24 KB.

In contrast, an initial response of *ZERBER+R* contains $b$ elements and the total response size including follow-up requests required to obtain top- $k$ elements is calculated using Equation 3.19. We use initial response size $b$, $k$=10. In this workload, about 85 posting elements are returned from the ODP index per query term on average. Assuming that each posting element is encoded using 64 bits, this is approximately 5.3 Kb (0.7 KB) per query term response. The queries in the workload contain 2.4 terms on average, which allows a server for the execution of about 750 queries per second. On average, each snippet contains about 250 B including XML formatting, which yields 2.5 KB for the top-10 snippets. Thus average total response size for the top-10 results is 3.5 KB.

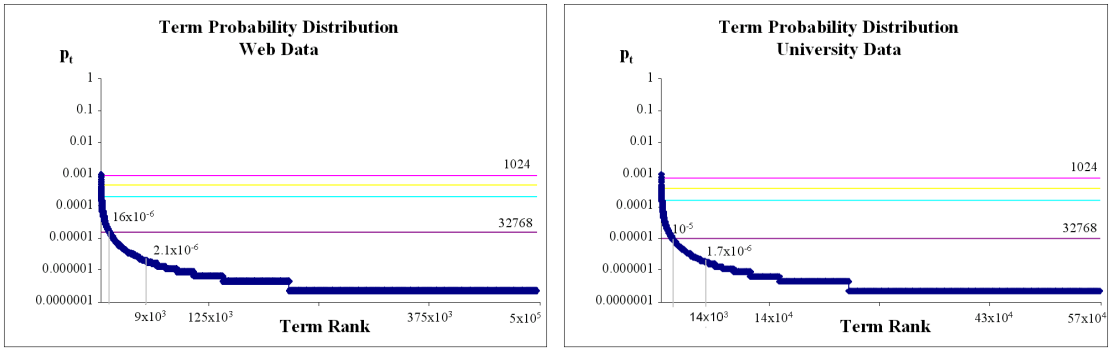
In comparison, Google's response for the top-10 results is about 15 KB, including the snippets as well as information used for presentation purposes (HTML, CSS, etc.), which is 1.6 times less than the *ZERBER* response size. Altavista returns 37 KB and Yahoo returns 59 KB of top-10 results, which are comparable to or bigger than *ZERBER* . However, *ZERBER* 's element shares are almost random, so standard HTML compression is ineffective. The compressed responses of Google, Altavista and Yahoo are 3, 2.4 and 1.6 times smaller than *ZERBER* responses, respectively. (Of course, a *ZERBER* response contains all answers.) The compressed responses of Google, Altavista and Yahoo are comparable to *ZERBER+R* responses. Further optimization can be achieved by adding search result checksums and caching them on the client, as defined in HTTP 1.0.

### 3.9.5 Selection of the Confidentiality Level *r*

As described in Section 3.5.3, *ZERBER* 's merging heuristics support a trade-off between query efficiency and confidentiality level *r*. Whether a particular value for *r* is appropriate depends on the (typically collection specific) document frequency distribution. In this section, we examine the choices for *r* for the test datasets.

We learned the document frequency distribution from the first 30% of the documents in the two datasets. In ODP this subcollection contained 70,000 documents and 500,000 terms. As document frequencies follow a Zipfian distribution, this should be sufficient to learn the most frequent terms in the collection. (Terms introduced later should be less frequent, and we assigned them uniformly to the existing posting lists.) Since the central indexes contain documents from many collections on a variety of topics, and we did not remove stop words, the most frequent terms are not specific to any particular collection and the adversary should already know their occurrence probabilities from her background knowledge. Under this assumption, the most frequent terms are least in need of the protection that comes from merging, and our goal is to make it impossible for the adversary to distinguish elements containing the rarer terms from elements containing more frequent terms.

We used the term occurrence probability distribution ($p_t$ in Equation 3.2) to help us set target values for r. Figure 3.13 shows $p_t$ for the Stud IP and ODP datasets. The $X$-Axis shows the terms in descending order of frequency. The horizontal lines show the $1/r$ values for 1,024, 2,048, 4,096, and 32,768 posting lists. Both subfigures show that the term




**(a)** Web data **(b)** University data

**Figure 3.13** *r*-Parameter Selection

probability distribution is Zipfian, with the top few percent of terms far more frequent than others. $10^{-6}$ is the smallest value of pt among the 10% most frequent terms. When we merge posting lists, we would like the aggregate term probability of every merged list to be at least this big.

From Figure 3.13, we concluded that our test data sets should have at most 32K merged lists, because with 65K or more lists, the merging heuristics would not be able to attain our target r-*value* of $10^{-6}$. With 32K merged lists, every term with original probability $p_t < 16.09 * 10^{-6}$ will reside in a posting list with aggregate term probability exceeding that of any but the 1.83% most frequent terms. In Figure 3.16a, these protected terms reside to the right of the intersection of the 32,768 line with the term probability distribution curve, projected onto the $X$-Axis. Each term to the left of this point will have a posting list of its own under BFM and DFM, and each term to the right will be merged with at least one other term. Methods of choosing a target value for r that adapt to the characteristics of the document frequency distribution are an interesting direction for future work.

**Table 3.1** *r*-Parameter Value for 3 Merging Heuristics

| # of Posting Lists | 1/r for BFM, DFM | 1/r for UDM |
|---|---|---|
| 1,024 | 9.30*10$_{-4}$ | 7.86*10$_{-4}$ |
| 2,048 | 4.45*10$_{-4}$ | 3.57*10$_{-4}$ |
| 4,096 | 2.07*10$_{-4}$ | 1.58*10$_{-4}$ |
| 32,786 | 16.09*10$_{-6}$ | 9.60*110$_{-6}$ |

We used the DFM and UDM algorithms to create 1K, 2K, 4K, and 32K posting lists, and then computed the resulting *r* values using Equation 3.7 for the minimal sum of probabilities accumulated in a merged posting list. We tweaked the input value of *r* given to the BFM algorithm so that it would also produce the same number of lists. Table 3.1 presents the resulting r values for the web dataset. For a given number of posting lists, BFM and DFM produce the same r value. Table 3.1 shows that UDM offers less confidentiality on

average (see also Figure 3.15).

Figure 3.14 shows the correlation between $r$ and the number $M$ of merged posting lists for ODP and BFM/DFM. As $M$ increases, the confidentiality level decreases according to the Zipfian term probability distribution in the underlying data (see also Figure 3.13).

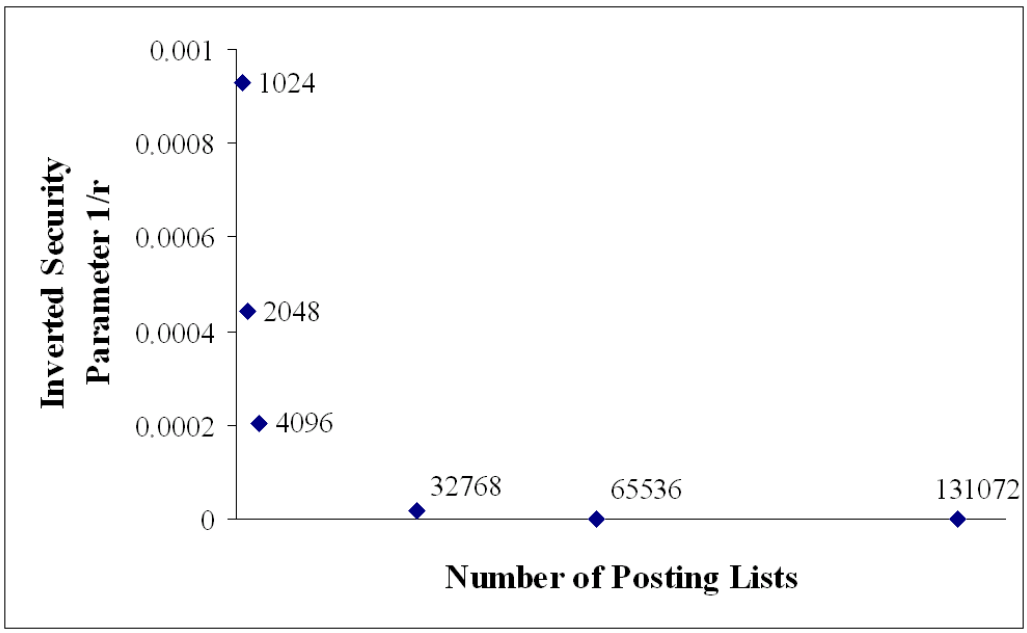



**Figure 3.14** Correlation Between $r$ and $M$ for ODP & BFM/DFM

### 3.9.6 Comparison of Merging Heuristics

In this section, we analyze the security and the query efficiency provided by the different merging heuristics with the ODP data. Figure 3.15 shows the amplification $r$ of the original term occurrence probability with different merging heuristics, for ODP data. To improve visibility, we show only the top 1000 terms in the 1,024 index. UDM's curve deviates from the DFM curve and exceeds its $r$-value in several places. However, UDM is comparable to DFM on average, and has the advantage of giving higher confidentiality to very common terms. DFM and BFM give the top 1.83% of terms their own individual posting lists, but UDM merges even these most popular terms.

We conducted extensive simulations to evaluate the effects of the merging heuristics on query efficiency. For each dataset, we merged the posting lists using BFM, DFM, and UDM for 1,024, 2,048, 4,096 and 32,768 merged posting lists.

Figure 3.16 shows the average ratio of the total workload cost *QRatio(t)* due to term $t$ in its merged posting list $L$, versus the workload cost attributable to t with unmerged posting lists. The curves in the figures correspond to terms with document frequency $DF$ of 1, 1000, and 3500. The $X$-Axis gives the number of posting lists in the index, and the $Y$-Axis

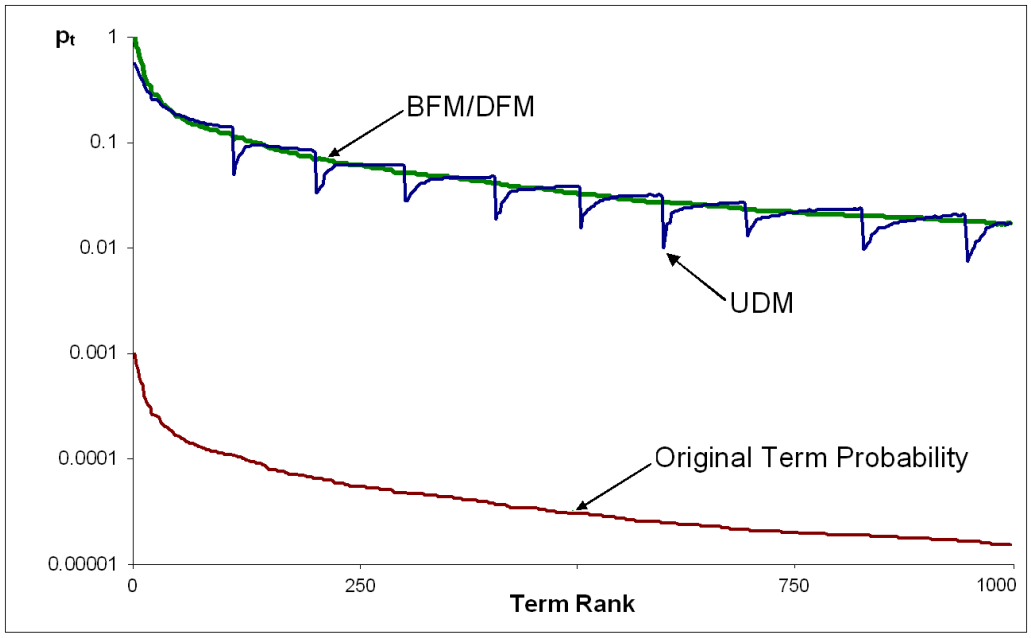



**Figure 3.15** Term Probability Amplification with 1,024 Posting Lists and Different Merging Heuristics

shows the workload cost ratio, calculated as:

$$QRatio(t) = \frac{\sum_{U \in L} DF_u \cdot \sum_{u \in L} qf_u}{DF_t \cdot qf_t}, \tag{3.17}$$

where $qf_x$ is the query frequency of term $x$.

Figure 3.16 shows that as expected, merging mostly affects the costs of queries with rarer terms. Overall, increasing $M$ significantly improves the cost ratios for terms with low and medium DF. In the 32,768 index with BFM/DFM, queries over terms with high and medium DF are nearly unaffected by merging. Queries over high-DF terms perform well already with only 4K lists. UDM query performance for high- and medium-DF terms is comparable to that of BFM/DFM for 32K posting lists; However, UDM slows down queries over low-DF terms more than the other schemes do.

Comparing Figures 3.13 and 3.16, we also see that there is a trade-off between the ability of the index to hide the occurrence of the most frequent terms, and the query processing overhead.

We calculated the efficiency in query answering $QRatio_{eff}$ introduced by different merging heuristics as the ratio between the number of posting elements that correspond to the query term $t$ and the total number of posting elements in its merged list $L$:

$$QRatio_{eff}(t) = \frac{DF_t}{\sum_{u \in L} DF_u}. \tag{3.18}$$

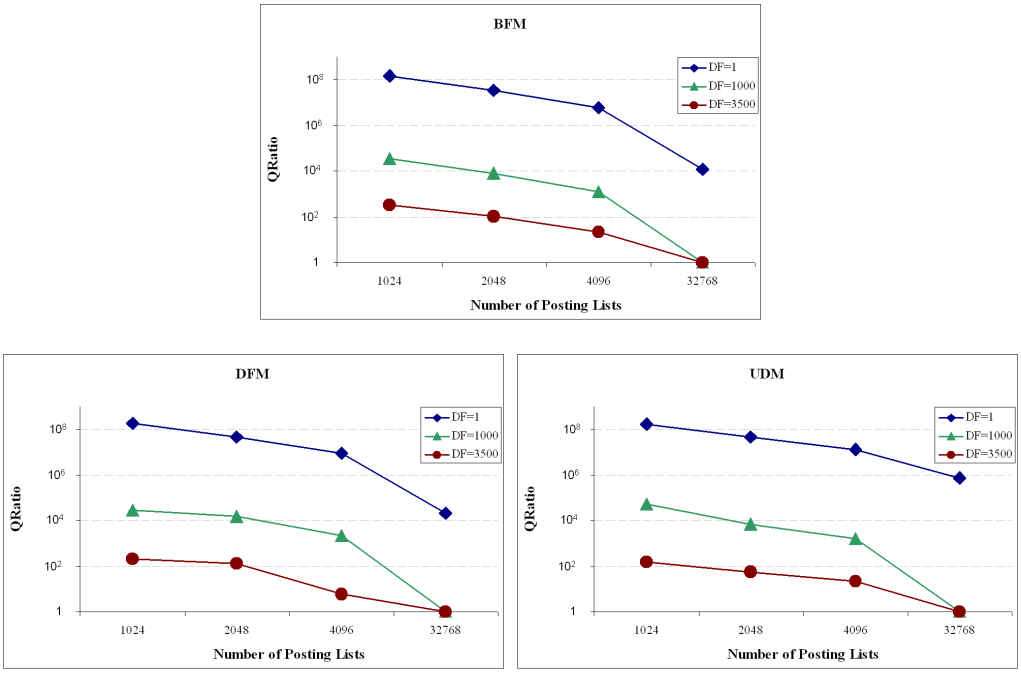



**Figure 3.16** Ratios of Workload Cost for BFM, DFM and UDM

Figure 3.17 plots the efficiency in query answering $QRatio_{eff}$ for indexes with 32K merged posting lists. In this figure, the $Y$-Axis shows $QRatio_{eff}$ and the $X$-Axis represents the query terms in the workload (in %), ordered by $QRatio_{eff}$ .

The best query efficiency distribution among the merging heuristics is attained using the DFM/BFM index with 32K lists. In that index, the longest running 70% of the queries in the workload have an efficiency value $QRatio_{eff} > 0.96$ and the next 10longest-running queries have $QRatio_{eff} = 0.75$ on average. The shortest running 20% of the queries have average $QRatio_{eff} = 0.2$.

Figure 3.18 plots the response size from the DFM index with 32K lists. The $X$-Axis shows the posting lists ordered by the number of elements they contain, and the $Y$-Axis shows the total number of posting elements in the posting lists, computed as the sum of document frequencies of the terms in a merged posting list. Figure 12 shows that only 40% of the posting lists have a response size exceeding 100 posting elements.

The largest response obtained from the ODP test collection using a DFM-32,768 index contains 10K posting elements. On the platform described in Section 3.9.1, 700 posting elements are decrypted in 1 msec on average. Thus only 14.3 msec are needed to decrypt the search results from one server for this response.

The simulation results described above have shown that BFM and DFM heuristics offer very reasonable query performance. Our experiments show that the query workload cost ratio for long and middle-size posting lists in the central indexes can be kept comparable to a conventional inverted index while providing security guarantees for 98% of the terms

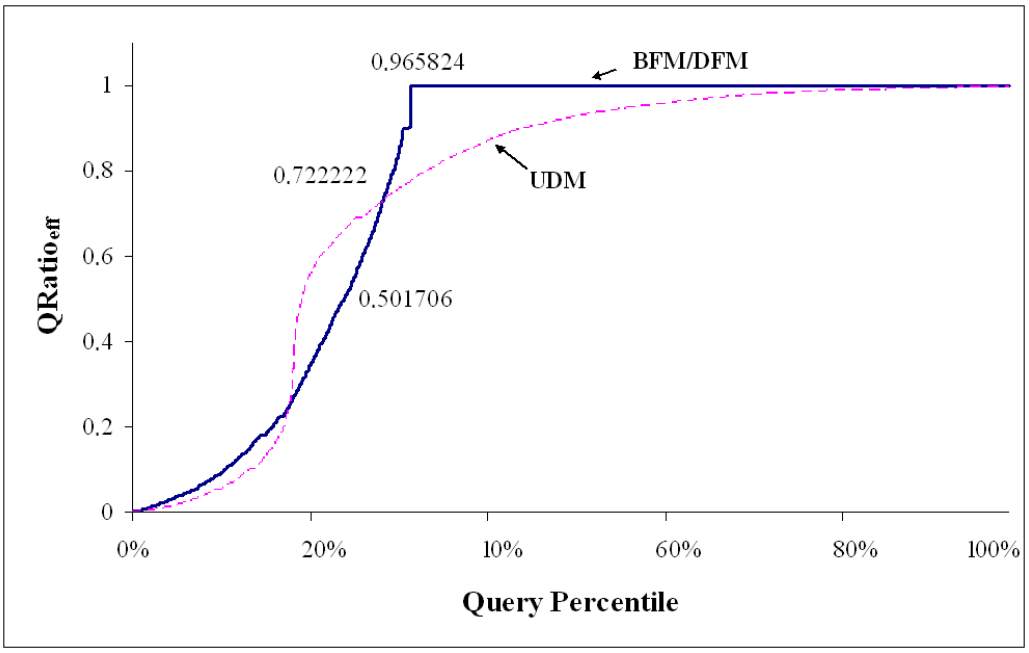



**Figure 3.17** Efficiency in Query Answering for Merged Posting Lists

in the dataset (with 32K posting lists). The remaining 2% of the terms are common words, which are usually not collection specific (e.g., "remaining") and do not require protection in the global index containing documents from a number of collections.

This performance can be achieved without learning query statistics, which is important for query confidentiality. BFM is more straightforward to implement given an $r$-parameter value, as it does not require pre-estimation of the mapping table size.

One of our research questions was the preferable heuristic for posting list merging. In general, there were no significant differences between the BFM and DFM heuristics. In contrast, UDM requires increased bandwidth by queries over low-*DF* terms and slows down their processing at user peers.

### 3.9.7 Selection of the Initial Response Size for Top-k Retrieval

As discussed in Section 3.8.4 *ZERBER+R* increases its response size progressively depending on the number of follow-up requests to a query. The initial response size should be selected in a way to minimize the number of follow-up requests and the bandwidth overhead for the majority of the queries in the workload.

We denote the number of posting elements in the first response as initial response size ***b*** and the accumulated number of posting elements in a sequence containing ***n*** follow-up requests as a total response size ***TR***es. Given the number of follow-up requests, the total response size can be calculated as:

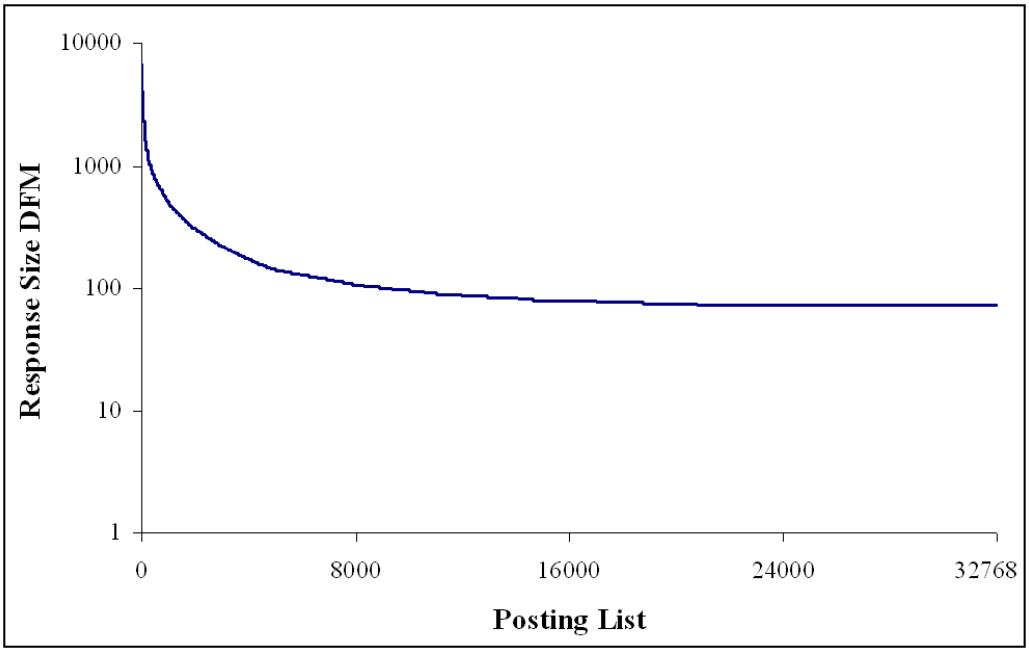
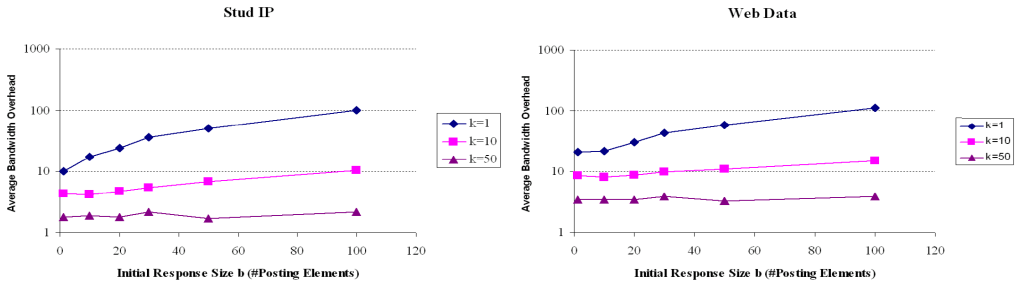



**Figure 3.18** Response Size for the DFM Index with 32K Lists

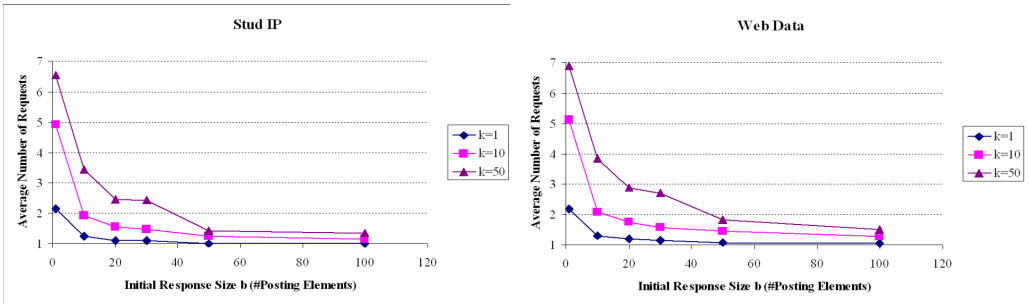
$$TRes = b \cdot \sum_{i=0}^{n} 2^2. \tag{3.19}$$

Figure 3.19 presents an average bandwidth overhead ***AvBO*** over the set of queries ***Q*** in the query workload calculated as the ratio between the total response size ***TR***es of *ZERBER+R* required in order to obtain the top-k elements and ***k*** elements returned by an ordinary inverted index in response to a top- ***k*** query:




**(a)** **(b)**

**Figure 3.19** Average Bandwidth Overhead of *ZERBER+R*




**(a)** **(b)**

**Figure 3.20** Average Number of Requests for *ZERBER+R*

$$AvBO = \frac{\sum_{q \in Q} \left( \frac{TRes(q)}{k} \right)}{|Q|}. \tag{3.20}$$

The $X$-Axis of Figure 3.19 show the number of posting elements in the initial query response. The $Y$-Axis shows an average bandwidth overhead in both test collections calculated using Equation 3.20 for ***k***=1, 10 and 50.

Figure 3.19 shows that the minimal bandwidth overhead for a top- ***k*** query in *ZERBER+R* can be achieved with $b$=$k$, i.e. by returning around ***k*** elements. Further enlargement of the initial response size leads to an increased bandwidth overhead.

Figure 3.20 shows an average number of requests required in order to obtain the top- ***k*** elements dependent on the number of elements in the initial response for ***k***=1, 10 and 50 in both test collections. The $X$-Axis shows the number of elements in the initial response. The $Y$-Axis shows an average number of requests required to obtain the top-k elements for the queries in the workload.

Figure 3.20 also illustrates that with an initial response size of approximately 10 elements most of the query terms return the top- 10 results within 2 requests (returning 30 posting elements in total). In order to further reduce the number of requests, the initial response size needs to be significantly increased. However this is not desirable because of the significant increase in the bandwidth overhead. Thus for our further experiments we selected ***k*** as the preferable initial response size for a top- ***k*** query.

### 3.9.8 Query Performance for Top-k Retrieval

We calculated the efficiency in query answering $QRatio_{eff}$ introduced by different sizes of the initial response as the ratio between ***k*** and the total *ZERBER+R* response size:

$$QRatio_{eff} = frackTRes. \tag{3.21}$$

Figure 3.21 plots the efficiency in query answering $QRatio_{eff}$ for the top- ***k*** request

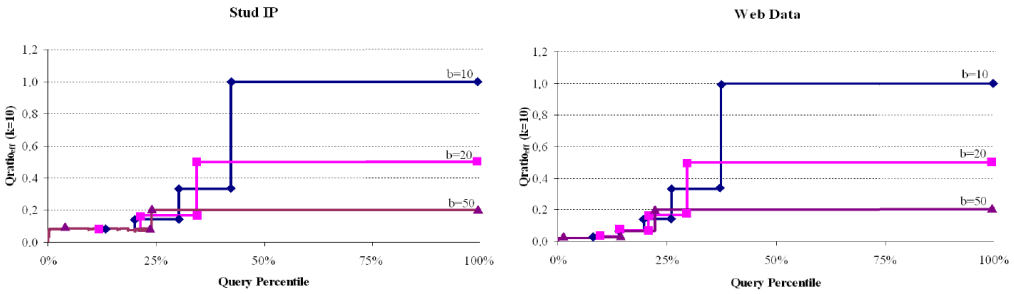



**Figure 3.21** Efficiency in Query Answering for *ZERBER+R*

with ***k***=10 and the initial response size b=10, 20 and 50 elements in the both test collections. In this figure, the $Y$-Axis shows $QRatio_{eff}$ and the $X$-Axis represents the query terms in the workload (in %), ordered by $QRatio_{eff}$.

The best query efficiency distribution for the top-10 request in the both test collections is attained using the initial response size ***b***=10. In this case around 60% of the longest running queries in the workload have an efficiency value $QRatio_{eff}$=1and the next 20% longest-running queries have $QRatio_{eff}$=0.2 on average. The shortest running 20% of the queries have average $QRatio_{eff}$=0.1. Increasing the initial response size to 20 elements leads to the significant reduction of QRatioeff for all longest running queries. In this case around 70% of the longest running queries in the workload have an efficiency value $QRatio_{eff}$=0.5 and the shortest running 30% of the queries have average $QRatio_{eff}$=0.1.

The simulation results described above have shown that initial response size of ***b***=10 offers very reasonable query performance for a top-10 query. Our experiments show that using *ZERBER+R* the query workload cost ratio can be kept comparable to a conventional inverted index for 60% of the queries, while preserving the *r*-confidentiality of the index.

## 3.10 Discussion

In this chapter we addressed Problem 1 of privacy-preserving sharing of distributed documents. In this situation users need an indexing facility where they can quickly locate and select top-k relevant documents they are allowed to access, without (1) leaking information about the remaining documents, (2) imposing a large management burden as users, groups, and documents evolve, or (3) requiring users to trust a central authority.

To address these problems, we proposed a tunable *r*-confidentiality measure, as the degree of information from inaccessible documents an index can leak, given an adversary compromises the index and possesses some background knowledge on the corpus and/or language statistics. We presented *ZERBER* , an *r*-confidential global inverted index for sensitive documents. *ZERBER* relies on a centralized set of *largely untrusted* index servers and offers resistance against inappropriate information disclosure even if *k*-1 index servers are compromised. To provide tunable resistance to statistical attacks, *ZERBER* employs a

novel term merging scheme that has minimal impact on index lookup costs. Our experiments show that *ZERBER* makes economical use of network bandwidth, requires minimal key management, and answers queries almost as fast as an ordinary inverted index.

Then, we further enhanced *ZERBER* and presented *ZERBER+R* , a ranking model that enables top-k retrieval from an *r*-confidential outsourced inverted index. *ZERBER+R* creates a relevance score transformation function, which makes relevance scores of different terms indistinguishable, in a way that even if they are known to an adversary they do not reveal any information about the indexed data. This enables the server to provide the top-k results most relevant to a user query. Our experiments on two real-world datasets show that *ZERBER+R* makes economical usage of bandwidth and offers Information Retrieval properties comparable with an ordinary inverted index while preserving the confidentiality of the indexed data.

So far we did not focus on the access control and the impact of the number of working groups on the index efficiency. Index partitioning, through clustering according to access control policies, has a potential to further improve efficiency. Many clustering techniques require the number of clusters as a parameter. In Chapter 4 we will discuss how the number of clusters in a dataset can be estimated exploiting a diversity index as well as computation of such diversity index for large datasets.

# 4 Scalable Content Diversity Analysis

Generally, inverted indexes have been shown very efficient for locating relevant documents in large corpora such as the Web. However, this is only true under the assumption that the searcher has access to all available documents this index is pointing to. In the case of access control, where the user only can access particular posting elements in a large posting lists, full scans of those list are required. These unnecessary scans can be avoided through index partitioning according to the given access control policies. This model has two extremes. On the one side, there is a single index where *for each posting element* at query time the access policy needs to be evaluated in order to include the element into the result set. On the other side, there are a number of indexes, one for each access control policy. Both examples have their advantages and drawbacks. In the first case the required storage space and the maintenance overhead are rather small, the processing overhead at the query time, however, rather large. Oppositely, in the second case the processing at query time is minimized, whilst the storage dramatically increases, as the number of partial indexes can be as big as the number of working groups. Since some of the working groups may overlap in terms of users and documents, these partial indexes will overlap as well and require additional maintenance and synchronization. Obviously, an optimum will be somewhere in between. Index clustering could be a natural solution of the described index partitioning problem in the access control scenarios with a large number of overlapping user groups. Many clustering techniques such as k-means [Jai10], k-medoids or the expectation-maximization algorithm for clustering require the expected number of clusters as a parameter that need to be estimated efficiently. In this chapter we present novel techniques for efficient diversity analysis for large document corpora. We are able to show, that the diversity score has direct relation to the number of clusters in the dataset. This score has a potential to estimate parameters for index partitioning also with respect to the access control policies.

Diversity has been studied in different disciplines and contexts for decades. The diversity of a population can reveal certain cultural properties of a country [Lie69, Fea03], e.g. with respect to religion, ethnic groups, or political orientations. In the area of ecology, biodiversity is used as a measure of the health of biological systems [LM03]. Recently,

diversity also drew the attention of scientists in the database and Information Retrieval communities. For instance, Vee et al. [VSS+08] use the sum of similarities of all object pairs to measure diversities of relational records, while Ziegler et al. [ZMKL05] employ a similar measure for diversifying items in a recommender system. In the area of search result diversification [GS09, ZMKL05], inter-object similarity is used to obtain a subset of the most dissimilar results, providing an overview over the result space. However, due to the quadratic computational complexity, the mentioned approaches are applied on relatively small sets, limited to the number of life forms in a biosystem, or top-k objects selected in the context of search result diversification.

## 4.1 Contributions

In this chapter, we address Problem 2 and focus on analyzing diversity on large-scale document collections and in other text corpora *as a whole*. In order to solve the diversity computational problem, we propose two novel algorithms which make use of random sampling and Min-wise independent hashing paradigms. More specifically, we propose two fast approximation algorithms for computing the average pair-wise Jaccard similarity of $n$ sets with probabilistic accuracy guarantees, coined as *SampleDJ* and *TrackDJ*. We discovered that specific properties of the Jaccard coefficient as underlying measure for the pair-wise similarities required in diversity indexes can make the computation feasible, even for huge amounts of data. Although there exist a variety of alternative metrics, Jaccard is still one of the most popular measures in IR due to its simplicity and high applicability [MRS08, BYRN99], and provides intuitive and interesting results in our example studies.

SampleDJ, which is based on random sampling, solves the problem in $o\left(\frac{1}{\mathrm{RDJ}^2}\right)$ time independent of the dataset size, where $\mathrm{RDJ}$ is the diversity index value to be estimated. TrackDJ, which is based on Min-wise independent hashing [BCFM00, Bro00b, Bro97, Bro00a] solves the problem in $O(n)$ time regardless of the input data distribution. Experiments on real-world as well as synthetic data confirm our analytical results. Furthermore, we show the applicability of our methods in example studies on various data collections.

## 4.2 Diversity Index Definition

There exists a plethora of methods for computing pair-wise similarities (e.g. cosine similarity, Jaccard coefficient, Okapi BM25, inverted distances, etc.). For diversity computation in this chapter, we employ the Jaccard coefficient, which is one of the most popular measures in IR due to its simplicity and high applicability [MRS08, BYRN99]. We will see that specific properties of this coefficient can make the computation of diversity values feasible even for large datasets.

Given two term sets $O_i$ and $O_j$ (e.g. sets of tags for two photos in Flickr), their Jaccard similarity $\mathrm{JS}(O_i, O_j)$ is defined as follows:

$$\mathrm{JS}(O_i, O_j) = \frac{|O_i \cap O_j|}{|O_i \cup O_j|}.$$

We define our topic diversity index as *Refined Diversity Jaccard-Index*, which is in fact the average Jaccard similarity of all object pairs.

**Definition 4.1 (Refined DJ-Index)** *Given a collection of objects $O_1, \ldots, O_n$, where each object is a set of elements (e.g. terms), the refined DJ-Index measuring the diversity of the collection is defined as follows:*

$$\mathrm{RDJ} = \frac{2}{n(n-1)} \sum_{i<j} \mathrm{JS}(O_i, O_j),$$

*where $1 \leq i < j \leq n$.*

We use the expression *Refined* DJ-Index, or *RDJ index* for short, to emphasize that self-similar pairs are not included in the diversity computation. It is easy to see that the RDJ-index can be considered as a special case of the Stirling index described in Section 2.2, where $\alpha$ and $\beta$ are set to 1, and both $p_i$ and $p_j$ are $1/n$ if self-similar pairs are included. Note that smaller RDJ values mean larger diversities. For better visualization in plots we do not use 1-RDJ to measure diversity because RDJ values are usually quite small, and 1-RDJ is often close to 1.

**Computational Problem:** There are a number of well-known indexes measuring diversity in ecology such as Simpson's diversity [Sim49] and the Shannon index [Kre89]. For applying the concept of diversity to *text data*, existing works use the sum of all document pair similarities or variations based on pair-wise distances as diversity metrics [RBS10, GS09, ZMKL05, VSS+08]. These diversity indexes are based on the computation of pair-wise comparisons. Thus, a common problem is their computational complexity: $O(n^2)$ comparisons are necessary for sets of $n$ objects. While this is still feasible in scenarios with small amounts of data as for top-k candidates in query result diversification, it becomes prohibitive when computing topic diversity for large datasets.

## 4.3 Diversity Index Computation

Having defined the RDJ-index for measuring diversity, the goal of this section is to compute this statistic for a given text corpus.

**Problem 1 (RDJ-Index Computation)** *Given a collection of objects $O_1, \ldots, O_n$, where each object is a set of elements (e.g. terms), the objective is to compute the RDJ-index value efficiently.*

### 4.3.1 The Naive Method: All-Pair

The straightforward solution for computing RDJ directly according to Definition 4.1 is 1) to compare all object pairs and compute the size of intersection and union of each pair, and then 2) to sum up the similarities of all pairs and to divide by the number of object pairs. We call this algorithm *All-Pair*. Clearly, All-Pair takes $O(n^2)$ time, and is inefficient for data collections with a large number of objects. Although some heuristics may improve the efficiency of All-Pair, the time complexity will still be $O(n^2)$ since in the worst case all pairs will have to be compared to obtain the exact value. Note that All-Pair can be easily parallelized with $O(n^2/m)$ comparisons where $m$ is the number of machines, which, however, is still not feasible for large datasets. In the following, we study approximate but more efficient techniques to estimate the value of the RDJ diversity index.

### 4.3.2 Diversity Index Approximation

To increase the time efficiency of the RJD computation, we consider approximation algorithms that are usually faster but provide estimated results, which are subject to errors. Our goal is to bound these errors within a tolerable range. In the following, we define error measures for the estimation quality of approximation algorithms.

**Definition 4.2 (Absolute Error)** *Let $\hat{D}$ be an estimate of a statistic $D$, the absolute error is defined as follows:*

$$\left|\hat{D} - D\right|.$$

**Definition 4.3 (Relative Error)** *Let $\hat{D}$ be an estimate of a statistic $D$, the relative error is defined as:*

$$\frac{\left|\hat{D} - D\right|}{D}.$$

Like many other indexes, diversity indexes are mainly used for comparisons. A single index value alone usually cannot reveal much insight to users, and the relative error is more appropriate than the absolute error for measuring the accuracy of index value estimates. For the absolute error, it becomes hard for users to specify a meaningful accuracy requirement in form of an error bound unless they approximately know the value of the indexes to be estimated. Therefore, for both approximation algorithms described in the next subsections, we will focus our theoretical analysis on the relative error.

**Algorithm 6:** SampleDJ: An approximation algorithm estimating diversity indexes

**Input**: A collection of objects $O_1, \ldots, O_n$, each object is a set of term IDs (assuming the term IDs in each object are sorted); relative error bound $\epsilon$, accuracy confidence $1 - \delta$, the number of consecutive trials $W$ (e.g. 10000) before checking the accuracy and tolerable overall running time $T$.

**Output**: Estimates of RDJ-Index of the given collection

**begin**

1. Load the dataset into memory;
2. $r_1 = 0$; $r_2 = \lceil \log_2 \frac{1}{\delta} \rceil$ ;

**for** $i = 1, \ldots, r_2$ **do**

3. jsSum[i] = 0;

**repeat**

**for** $i = 1, \ldots, r_2$ **do**

**for** $j = 1, \ldots, W$ **do**

4. Generate 2 distinct integers, A & B, uniformly at random within the range [1, n];
5. jsSum[i] = jsSum[i] + JS($O_A$, $O_B$);

6. $r_1 = r_1 + W$;
7. $\widehat{\mathrm{RDJ}} = \frac{\mathrm{median(jsSum[1],...,jsSum}[r_2])}{r_1}$;
8. $\Delta = \sqrt{\frac{1}{r_1}}$;

**until** $\frac{\Delta}{|\widehat{\mathrm{RDJ}} - \Delta|} <= \epsilon$ *or Running time* $> T$;

**if** $\frac{\Delta}{|\widehat{\mathrm{RDJ}} - \Delta|} <= \epsilon$ **then**

9. Return $\widehat{\mathrm{RDJ}}$.

**else**

10. Return FAIL;

### 4.3.3 The SampleDJ Algorithm

A natural solution to reduce the computational cost is sampling. One approach is to take a random sample of the input dataset, i.e. to sample $r$ objects uniformly at random (without replacement) from a collection of $n$ objects, compute the sum of similarities of all object pairs in the sample, and scale the diversity index of the sample. Another approach is to take $r$ objects uniformly at random but with replacement (see also [Olk93] for a general discussion of sampling with replacement). In our context, we sample from all possible *pairs*, compute the similarities of those $r$ pairs and scale the result according to $r$ and $n$ as the final estimate. The advantage of this approach is that each pair is taken independently of other pairs, which makes the sampling analysis easier. We focus on this approach in our work and name it *SampleDJ*.

The details are shown in Algorithm 6. The input to the algorithm is a collection of objects with each object consisting of a set of term IDs. In the Line 2 the values of $r_1$ and $r_2$ are initialized depending on the desired accuracy confidence; here, $r_1$ is the number of trials in each of the $r_2$ experiments. The final result is computed as the median RDJ value obtained from the $r_2$ independent experiments ($r_2$ being determined by user requirement $\delta$); $r_1$ increases with each iteration of the algorithm while $r_2$ remains constant. In the Line 4 and 5 the similarity sum of all sampled pairs is computed. Line 6 keeps track of the number of trials. Line 7 computes the current RDJ estimate. Line 8 and the "until" statement serve to determine if the RDJ estimate is accurate enough.

Note that window size $W$ is independent of dataset sizes, and merely corresponds to regular accuracy checks in the algorithm; the number of windows is determined *automatically*, and changes according to the data properties. Therefore, no tuning of $W$ is required. In our experiments we used the same value for $W$ (10,000) for all of the datasets. For practical reasons, there is also a "tolerable running time" parameter $T$ because SampleDJ can take more running time than the straightforward method in the worst case, and this time is not predictable. For example, if all object pairs have zero similarity, SampleDJ would proceed forever without a stopping condition based on $T$. The algorithm stops when the estimated result is accurate enough or the running time exceeds the user-specified time limit. We name the former case as "fail" and the latter as "succeed". Based on our analysis described in the next paragraph, the accuracy is determined by the absolute error bound $\Delta$, which can be determined based on the number of trials $r_1$. Note that $\epsilon$ and $\delta$ are the user requirements for the accuracy of the diversity index computation, and $r_2$ is determined by user input $\delta$.

Since SampleDJ is an approximation algorithm, it is of high practical importance to have accuracy guarantees. The next statement specifies the time and space complexity of the SampleDJ algorithm.

**Claim 1 (Complexity of SampleDJ)** *If Algorithm 6 succeeds, the estimate* $\widehat{\mathrm{RDJ}}$ *is guaranteed to have a relative error at most* $\epsilon$ *with probability at least* $1-\delta$*. That is,*

$$\Pr\left[\frac{|\widehat{\mathrm{RDJ}} - \mathrm{RDJ}\,|}{\mathrm{RDJ}} > \epsilon\right] < \delta,$$

*where* RDJ *is the true value of* $\widehat{\mathrm{RDJ}}$*. The time and space costs to guarantee a success are* $O\left(\frac{U \cdot \log_2 1/\delta}{\epsilon^2\,\mathrm{RDJ}^2}\right)$ *and* $O(nU)$ *respectively, where* $U$ *is the maximum set size of all objects.*

We first prove the accuracy of SampleDJ. Let $p_1, \ldots, p_N$ be all possible object pairs, where $N = \frac{n(n-1)}{2}$. Let $X$ be a random variable indicating the Jaccard similarity of a pair drawn uniformly at random from $p_1, \ldots, p_N$. That is, $X = s_i$ $(i = 1, \ldots, N)$ with $\Pr[X = s_i] = \frac{1}{N}$, where $s_i$ is the Jaccard similarity of $p_i$. Then $E[X] = \frac{1}{N}\sum_{i=1}^{N} s_i = \mathrm{RDJ}$, and

$$\begin{aligned} \mathrm{VAR}[X] &= E\left[X^2\right] - (E[X])^2 = \frac{1}{N}\sum_{i=1}^{N} s_i^2 - \mathrm{RDJ}^2 \\ &\leq \mathrm{RDJ}(1 - \mathrm{RDJ}). \end{aligned}$$

If there are $r_1$ identically and independently distributed random variables following the same distribution as $X$, the mean of the $r_1$ variables is also expected to be equal to $\mathrm{RDJ}$ and the variance of the mean is at most $\frac{\mathrm{RDJ}(1-\mathrm{RDJ})}{r_1}$. According to Chebyshev's Inequality, we have

$$\Pr\left[\frac{\left|\widehat{\mathrm{RDJ}} - \mathrm{RDJ}\right|}{\mathrm{RDJ}} > \epsilon\right] < \frac{\mathrm{RDJ}(1 - \mathrm{RDJ})}{r_1 \epsilon^2\, \mathrm{RDJ}^2} \leq \frac{1}{4 r_1 \Delta^2},$$

where absolute error bound $\Delta = \epsilon \cdot \mathrm{RDJ}$. Let the probability bound $\frac{1}{4r_1\Delta^2} = \frac{1}{4}$, then $\Delta = \sqrt{\frac{1}{r_1}}$. By repeating the same experiment $\log_2 \frac{1}{\delta}$ times independently and taking the median, the upper bound of the probability $\Pr\left[\left|\widehat{\mathrm{RDJ}} - \mathrm{RDJ}\right| > \epsilon\, \mathrm{RDJ}\right]$ is raised to $\frac{1}{4}^{\frac{\log_2 \frac{1}{\delta}}{2}} = \delta$.

Next, we show the time and space complexities. To guarantee the relative error at most $\epsilon$ with probability $1 - \delta$, it is sufficient to guarantee the absolute error at most $\Delta$ since $\left|\widehat{\mathrm{RDJ}} - \Delta\right| \leq \mathrm{RDJ}$, $\frac{\Delta}{\left|\widehat{\mathrm{RDJ}} - \Delta\right|} \geq \frac{\Delta}{\mathrm{RDJ}} = \epsilon$. Therefore, setting $r_1 = \frac{1}{\Delta^2} = \frac{1}{\epsilon^2\, \mathrm{RDJ}^2}$ guarantees the relative error with probability $1 - \delta$ according to the accuracy analysis. Thus, the time complexity is $O\left(\frac{U \cdot \log_2 1/\delta}{\epsilon^2\, \mathrm{RDJ}^2}\right)$. As for the space complexity, the main costs come from storing the input, which is $O(nU)$.

Note that the running time of SampleDJ is independent of $n$, but dependent on $\mathrm{RDJ}$, the value to be estimated. Thus, SampleDJ can be very efficient for large datasets if $\mathrm{RDJ}$ is reasonably large. However, for smaller values of $\mathrm{RDJ}$, the running time can be higher. Note further that, similar to All-Pair, SampleDJ can be easily parallelized; this is because the pairwise similarity computations, which correspond to the main computational overhead, and part of the aggregation steps involved can be run independently on different machines.

### 4.3.4 TrackDJ: Estimating the RDJ-Index in Linear Time

Although SampleDJ can be very efficient in some cases, its performance is sensitive to the data distribution, and it can be very slow in the worst case. Apart from that, without prior knowledge of the input data, it is difficult to predict the running time. In this section we present TrackDJ, another approximation technique returning an accurate estimate for the DJ-Index in $O(n)$ time regardless of the input data distribution, where $n$ is the number of

objects (term sets) in the dataset. We first briefly sketch the underlying concept of Min-wise hashing and its application in our scenario.

### Prerequisites: Min-wise Independent Hashing

Broder et al. proposed a powerful technique called *Min-wise independent hashing* (Min-hash) [BCFM00, Bro00b, Bro97, Bro00a]. An interesting property of this technique is that the hashing collision probability of two objects is exactly equal to their Jaccard similarity. The basic idea of Min-hash mapping a term set is as follows: Min-hash picks a permutation $\pi$ uniformly at random from all possible permutations of the term vocabulary. Given a set of terms $O$, Min-hash applies $\pi$ on $O$ and takes the term that has the minimum rank after the permutation as the Min-hash value.

**Example 1 (Min-wise Independent Hashing)** *Given 3 objects $O_1(A, B, C)$, $O_2(B, C, D)$ and $O_3(C, E)$ where A, B, C, D, E are distinct terms. Assume TrackDJ uses two min-hash functions $h_1$ and $h_2$ where the two permutations are: from $(A, B, C, D, E)$ to $\pi_1 = (E, B, A, C, D)$ and $\pi_2 = (A, C, B, D, E)$. Under these permutations, the terms with minimum ranks are as follows:*

$$h_1(O_1) = B, h_1(O_2) = B, h_1(O_3) = E,$$
$$h_2(O_1) = A, h_2(O_2) = C, h_2(O_3) = C.$$

### The TrackDJ Algorithm

Given the Min-wise hashing property that more similar objects are more likely to have a hash collision, TrackDJ counts the number of collisions of all object pairs and estimates the diversity index. This is done by finding the self-join sizes[1] of all min-hash values. The self-join size of a set of items is in fact the similarity sum of all object pairs where the similarity function returns only 1 or 0 depending on whether the pair of objects match or not. In summary, the key idea of TrackDJ is that if there are more similar pairs, the self-join size of the min-hash values will be higher due to the Min-wise hashing property; instead of comparing all object pairs, the self-join size of a set of items can be computed in linear time.

Using the above Min-wise Hashing example, the self-join sizes of min-hash values under $h_1$ and $h_2$ are both 5. Accordingly, TrackDJ returns $\frac{5-3}{3(3-1)}$ as the RDJ-Index estimate where the 5 in the numerator is the average of the two self-join sizes, and the 3 in the denominator is the number of objects. In our example, the estimate is exactly the true RDJ-Index value. Although this may not always be the case in practice, the *expected* value of the estimate is equal to the true value, which we will prove below. Typically, TrackDJ needs to use a larger number of min-hash functions to reduce the variance.

[1] The Self-join size of a set of items, $F$, is defined as $F = \sum_i f_i^2$, where $f_i$ is the number of occurrences (frequency) of item $i$ in the dataset.

TrackDJ maps each object (e.g. a term set) to a min-hash value. The algorithm uses the fact that two objects have a higher probability to have a min-hash collision if they have a higher Jaccard similarity; the total number of collisions of all possible object pairs directly approximates the sum of their pair-wise similarities. Thus, by computing the self-join size of all min-hash values (i.e. the total number of collisions of all pairs), TrackDJ estimates the RDJ-Index value.

The detailed approximation method is shown in Algorithm 7. In addition to the input dataset and accuracy and confidence requirements $\epsilon$ and $1-\delta$, the user needs to specify the maximum term set size among the $n$ objects. Line 1 sets the number of independent trials and experiments. For each of the $L_2$ iterations there are $L_1$ trials. $L_1$ and $L_2$ are computed based on the accuracy and confidence requirements defined by the user. Note that for TrackDJ the number of required iterations are known in advance; thus no checks of error bounds are necessary at later stages. Details can be found in the algorithm analysis part below.

Line 2 and 3 initialize a buffer storing the $L_2$ self-join sizes. Line 4 and 5 initialize counters, which are used later to store the number of occurrences of each ID. In each of the $L_2$ iterations, there is a data stream with $L_1 \cdot n$ objects. Line 6 generates the $L_1 \cdot L_2$ min-hash functions. Line 7 increments the corresponding counter based on the min-hash value of object $O_i$. Line 8 and line 9 compute the accumulated self-join size so far. Note that by maintaining a linked list of non-zero counters, TrackDJ does not need to check all $D$ counters which can significantly improve the efficiency if $D$ is large. Line 10 resets non-zero counters of $\mathrm{IdCounts}[]$. Line 11 computes the median, and Line 12 returns the final result; it computes the average self-join sizes over $L_1$ iterations and removes the contribution from self-similar pairs.

### Analysis of TrackDJ

Similar to SampleDJ, TrackDJ is a randomized algorithm which approximates the RDJ-Index. In the following, we prove complexity and accuracy guarantees for TrackDJ.

**Claim 2** *The refined DJ-Index estimate from TrackDJ,* $\widehat{\mathrm{RDJ}}$*, is expected to be equal to the true value* RDJ*. The variance of* $\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}$ *is at most*

$$\mathrm{VAR}\left[\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}\right] \leq \frac{2(U-1)}{K},$$

*where $U$ is the maximum set size of all objects in the input object collection, and $K$ is the number of min-hash functions used.*

We first analyze the case with only one hash function; the case with $K$ hash functions corresponds to repeating the same experiment $K$ times independently, which we will discuss at the end of the proof.

**Algorithm 7:** TrackDJ: Estimating the RDJ-Index in Linear Time

**Input**: A collection of objects $O_1, \ldots, O_n$, each object is a set of IDs of terms ranging from 0 to $D-1$; a user specified relative error bound $\epsilon$ and confidence $1-\delta$; maximum set size $U$ among all objects

**Output**: an estimate of RDJ-Index of the given collection

**begin**

1. $L_1 = \left\lceil \frac{8(U-1)}{\epsilon^2} \right\rceil$; $L_2 = \lceil \log_2 \frac{1}{\delta} \rceil$;
2. Initialize counters $f[j]$ storing $L_2$ self-join sizes of min-hash values;

**for** $l = 1, \ldots, L_2$ **do**

3. $f[l] = 0$; //to store self-join sizes

4. Initialize counters $\mathrm{IdCounts}[]$ storing the frequency of each min-hash values;

**for** $l = 1, \ldots, D$ **do**

// ($D$ is the number of distinct terms in the dataset)

5. $\mathrm{IdCounts}[l] = 0$; //counts of min-hash values

**for** $k = 1, \ldots, L_1 \cdot L_2$ **do**

6. Pick a min-hash function $h_k[]$ uniformly at random from a min-hash function family;

**for** $j = 1, \ldots, L_2$ **do**

**for** $l = 0, \ldots, L_1$ **do**

**for** $i = 1, \ldots, n$ **do**

7. $\mathrm{IdCounts}[h_{(j-1)L_1+l}[O_i]] + +$ ;

8. $\mathrm{sj} = \sum_{\mathrm{ID}} \mathrm{IdCounts}[\mathrm{ID}]^2$; //efficiency can be improved, see the algorithm description

9. $f[j] = f[j] + \mathrm{sj}$;

10. Reset non-zero counters of $\mathrm{IdCounts}[]$;

11. Determine the median of $f[j], j = 1, \ldots, L_2$ and set it to $\mathrm{fm}$;

12. return $\frac{(\mathrm{fm}/L_1)-n}{n(n-1)}$;

Let $X_{ij}$ be a random variable indicating if a pair of objects $O_i$ and $O_j$ are mapped to the same min-hash value, i.e. $X_{ij} = 1$ if $h(O_i) = h(O_j)$ and 0 otherwise. Also, let $Y = \sum_{i \neq j} X_{ij}$. (Note that $X_{ij} = X_{ji}$.) Thus, $\widehat{\mathrm{RDJ}} = \frac{1}{n(n-1)} Y$. Let $s_{ij} = \mathrm{JS}(O_i, O_j)$, then $\Pr[X_{ij} = 1] = s_{ij}$, and $E[X_{ij}] = s_{ij}$. Thus,

$$E[\widehat{\mathrm{RDJ}}] = \frac{1}{n(n-1)} \sum_{i \neq j} E[X_{ij}] = \frac{1}{n(n-1)} \sum_{i \neq j} s_{ij} = \mathrm{RDJ}\,.$$

$$\begin{aligned}
\mathrm{VAR}[Y] &= E\left[\left(\sum_{i \neq j} X_{ij}\right)^2\right] - \left(E\left[\sum_{i \neq j} X_{ij}\right]\right)^2 \\
&= E\left[\sum_{i \neq j} X_{ij}^2 + \sum_{i \neq j \neq k} X_{ij}X_{ik} + \sum_{i \neq j \neq k \neq l} X_{ij}X_{kl}\right] - \left(\sum_{i \neq j} s_{ij}\right)^2 \\
&\leq \sum_{i \neq j} s_{ij}(1 - s_{ij}) + \sum_{i \neq j \neq k} s_{ij}(1 - s_{ik}) + \sum_{i \neq j \neq k \neq l} s_{ij}(1 - s_{kl})
\end{aligned}$$

Due to the definition of Jaccard similarity (sets overlap size divided by sets union size), if $s_{ij} > 0$ we have

$$s_{ij} \geq \frac{1}{2U - 1}.$$
$$\therefore (1 - s_{ij}) \leq \frac{2U - 2}{2U - 1} \leq 2(U - 1)s_{ij}.$$

$$\begin{aligned}
\therefore \mathrm{VAR}[Y] &\leq 2(U-1)\left(\sum_{i \neq j} s_{ij}^2 + \sum_{i \neq j \neq k} s_{ij}s_{ik} + \sum_{i \neq j \neq k \neq l} s_{ij}s_{kl}\right) \\
&= 2(U-1)\left(\sum_{i \neq j} s_{ij}\right)^2
\end{aligned}$$

$$\therefore \mathrm{VAR}\left[\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}\right] = \frac{\mathrm{VAR}[\widehat{\mathrm{RDJ}}]}{\left(\frac{1}{n(n-1)} \sum_{i \neq j} s_{ij}\right)^2} \leq 2(U - 1).$$

If TrackDJ uses $K$ min-hash functions independently, then

$\mathrm{VAR}\left[\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}\right] \leq \frac{2(U-1)}{K}$.

With this statement, we can derive the time and space complexities of TrackDJ.

**Claim 3 (Complexity of TrackDJ)** *To estimate the refined DJ-Index and guarantee a small relative error (at most $\epsilon$) with a high probability (at least $1 - \delta$), TrackDJ requires $O\left(\frac{\log(1/\delta)}{\epsilon^2} nU\right)$ time and $O(\log(nU)nU)$ memory bits, where $n$ is the number of objects in the input dataset and $U$ is the maximum set size among all objects.*

Knowing the variance from Claim 2, by Chebyshev's Inequality we can observe that $K$ in TrackDJ should be set to $O\left(\frac{U}{\epsilon^2}\right)$ to bound the relative error to $\epsilon$ with a constant probability (e.g. $\frac{1}{2}$). To increase the success probability, we can repeat the process described before $2\log(1/\delta)$ times independently and report the median of the outputs. In this way, the success probability can be boosted to $1-\left(\frac{1}{2}\right)^{\log(1/\delta)}$. Therefore, TrackDJ has a running time of $O\left(\frac{\log(1/\delta)}{\epsilon^2}nU\right)$. The space costs come from storing the input and ID counters are as follows: TrackDJ needs to store $O(nU)$ counters of all term IDs, which is equal to the space requirement of the input. Each frequency counter in the buffer needs $O(\log(nU)$ bits, resulting in the overall space complexity of $O(\log(nU)nU)$.

Note that TrackDJ can be easily parallelized with a complexity of $O(n/m)$. This can be done by partitioning the data into $m$ parts for $m$ machines, computing self-join sizes separately for the subsets using TrackDJ, and aggregating the result on a central machine accordingly.

## 4.4 Evaluation

We evaluated the algorithms described in this chapter using both synthetic and real world datasets[2]. In this section, we first elaborate on datasets used. We then compare the three algorithms, All-Pair, SampleDJ and TrackDJ, under different efficiency aspects including accuracy, time and space costs, and parameters. Note that there exist no other methods for efficiently computing the diversity of large document corpora we could compare with.

### 4.4.1 Data

#### Real Data

Our real-world datasets were obtained from Flickr[3] and DBLP. DBLP [Ley] is a Computer Science Bibliography database containing more than 1.2 million bibliographic records. The data records in DBLP are mostly contributed by human editors and are well-structured. From DBLP, we extracted 1,256,089 paper titles based on the publication year and venue. We only considered conference and journal papers and removed books and other article types. Each paper title was considered as one object for our diversity analysis. For reasons of completeness and cleanness we focused on DBLP paper titles to mine topic diversities of computer science papers.

Finally, we gathered tag assignments for 134 Mio *Flickr* photos uniformly over the time period from 01 Jan 2005 until 05 Sept 2010. From this set we selected a subset of 25 Mio photos where each of photo contained at least 3 English tags (though a dictionary check), and performed stemming.

[2] Source code and datasets are available at www.L3S.de/~deng/diversity/.

[3] www.flickr.com

#### Synthetic Data

We created additional synthetic datasets in order to study the performance of different algorithms on datasets with various RDJ-index values. To this end, we generated groups of objects with the property that 1) all objects within a group had the same number of terms and were pair-wise similar with the same similarity, and 2) objects in different groups had always similarity $0$. By adjusting the number of groups with different sizes (i.e. the number of objects in each group), we constructed multiple datasets with different RDJ-index values. More specifically, we constructed the datasets as follows: 1) We set $U$, the number of term IDs in each object to $10$. 2) In each group, we generated a set of common IDs shared by all objects in the group. All other IDs in the group were distinct; in this way, by controlling the common ID numbers, we set the pair-wise similarity of all pairs in each group to around 0.5. 3) By varying the number of groups $G$ and group size $Gs$, we created multiple synthetic datasets with $n = G \cdot Gs = 524,288$ objects each.

### 4.4.2 Implementation Details

To implement the straightforward *All Pair* algorithm, we used an array of size $n$ to store the input file, with each array element being, in turn, an array containing the term IDs of an object. To compute the RDJ-Index values, All-Pair iterates over all possible object pairs, and computes the Jaccard similarity of each pair by linearly scanning the two sets of sorted term IDs and counting the number of common terms and the total number of terms. To implement *SampleDJ*, we used the system-provided `lrand48()` function to generate random numbers within the range $1$ and $n$. We set $W$, the number of consecutive trials to 10000. For *TrackDJ*, we used linear hashing to implement Min-Wise independent hashing as suggested by Broder et al. [BCFM00]. We used the same settings for error bound and confidence as for SampleDJ.

### 4.4.3 Performance Experiments

In this set of experiments, we compared the performances of the 3 algorithms All-Pair, SampleDJ and TrackDJ on real-word and synthetic data.

#### Performance Metrics

The three metrics used to measure the performance of the algorithms were running time, space (memory) requirements, and accuracy. In terms of memory, the primary costs of All-Pair and SampleDJ were almost the same: namely, storing all term IDs, requiring e.g. about 33MB for the DBLP title IDs. TrackDJ requires more space for storing the frequency counters. Depending on the datasets, this cost can be as large as the input dataset in the worst case. For the DBLP dataset, it is less than 1/10 of the input data. Because

the space cost is relatively clear and corresponds directly to the input dataset size, we focused on running time and accuracy in our experiments. The preliminary experiments with the synthetic data on the effect different parameters were conducted on a machine with 2x Xeon5320 1.86GHz processors and 16GB of memory running CentOS 5.3. For experiments on the real world dataset, we used one separate core on a more powerful machine with 8x Quad-Core AMD Opteron 2.7GHz processors and 256GB of memory running the same OS. Programs were coded in C and compiled using gcc 4.40.

### Performance Results on Real-World Datasets

Experiments for the approximation algorithms were performed with an error bound $\epsilon$ of 0.1 and a confidence value of $1 - \delta = 0.95$. Table 4.1 shows the running time and error values (along with the exact RDJ values computed through All-Pair). Experimental results are consistent with our theoretical findings described in Section 4.3.

For the naive All-Pair solution, we observe that running times are rather small for dataset sizes of 1,000 and 10,000 but, due to its quadratic behavior, quickly become infeasible for larger sets.

Our SampleDJ approach shows the best running time behavior of the three tested algorithms. Running times stay in the order of magnitude of minutes, and are almost constant with respect to the input size. The shorter running times for relatively small data sizes are an artifact of the hardware, and can be explained by L1 and L2 caching as for small datasets a larger part of possible object pairs can be kept in the caches; for larger sizes we observed a constant running time (approx. 5 min for the Flickr dataset), consistent with our theoretical findings in Section 4.3.3. Also consistent with our theoretical findings, TrackDJ shows linear behavior and outperforms All-Pair for datasets of size $n$ larger than 1 million. Note that for the approximation algorithms TrackDJ and SampleDJ the actual error is clearly below the defined error bound (in most cases less than 0.001 for $\epsilon = 0.1$) For the given datasets, SampleDJ shows much better performance than TrackDJ; however, in the next paragraph we will see that TrackDJ can be more efficient for data distributions with very high diversity.

### Effect of Data Characteristics and Error Bound

We have seen that SampleDJ always provides accurate estimates within short running time. This is because the RDJ value is still relatively large. However, Table 4.2a shows SampleDJ degrades indeed in $O(\frac{1}{RDJ^2})$ rate for varying $RDJ$ values on our synthetic data as shown in our theoretical analysis. In comparison, All-Pair and TrackDJ are not sensitive to the RDJ values; TrackDJ outperforms SampleDJ if RDJ values are small.

In another set of experiments, we used the first 1000 lines of the DBLP data to test the effect of the relative error bound $\epsilon$ on running time and accuracies of SampleDJ and TrackDJ. We set $\delta$ to a fixed vale of $0.001$ and varied $\epsilon$. Table 4.2b confirms that the running time of both SampleDJ and TrackDJ are indeed $O(\frac{1}{\epsilon^2})$. Ideally, the errors of both

**Table 4.1** Running times, RDJ value and error for All-Pair, SampleDJ and TrackDJ for Flickr and DBLP

**(a)** Flickr

| **Dataset Size** | **All-Pair** | | **SampleDJ** | | **TrackDJ** | |
|---|---|---|---|---|---|---|
| n | RDJ | Time (seconds) | Error (%) | Time (seconds) | Error (%) | Time (seconds) |
| 1,000 | 0.002060 | 0.08 | 0.017 | 34.30 ( 0.57 min ) | 0.000 | 39.33 ( 0.66 min ) |
| 10,000 | 0.001992 | 8.82 | 0.028 | 40.05 ( 0.67 min ) | 0.013 | 410 ( 6.84 min ) |
| 100,000 | 0.001992 | 912.74 ( 15.21 min ) | 0.019 | 90.14 ( 1.50 min ) | 0.043 | 5,253 ( 1.46 h ) |
| 1,000,000 | 0.001993 | 97,215.13 ( 27 h ) | 0.080 | 223 ( 3.72 min ) | 0.041 | 51,730 ( 14.37 h ) |
| **Dataset Size** | **All-Pair** | | **SampleDJ** | | **TrackDJ** | |
| n | | Time (seconds) | RDJ | Time (seconds) | RDJ | Time (seconds) |
| 10,000,000 | . | *113 days** | 0.001998 | 350 ( 5.84 min ) | 0.001997 | 790,016 ( 9.14 days ) |
| 20,000,000 | . | *450 days** | 0.002203 | 246 ( 4.10 min ) | 0.002206 | 1,613,566.20 ( 18.68 days ) |
| *estimated value | | | | | | |

**(b)** DBLP

| **Dataset Size** | **All-Pair** | | **SampleDJ** | | **TrackDJ** | |
|---|---|---|---|---|---|---|
| n | RDJ | Time (seconds) | Error (%) | Time (seconds) | Error (%) | Time (seconds) |
| 1,000 | 0.005380 | 0.26 | 0.053 | 4.83 | 0.032 | 11 |
| 10,000 | 0.005591 | 7.74 | 0.307 | 5.27 | 0.056 | 152 |
| 100,000 | 0.005692 | 580 ( 10 min ) | 0.197 | 7.50 | 0.078 | 1,869 ( 31 min ) |
| 1,000,000 | 0.005653 | 85,882 ( 24 h ) | 0.007 | 24.64 | 0.036 | 29,155 ( 8 h ) |
| 1,256,089 | 0.005656 | 135,549 ( 38 h ) | 0.061 | 27.04 | 0.036 | 35,879 ( 10 h ) |

solutions should strictly decrease when the error bound decreases, but in fact this is not the case in our experiments and the error of both solutions fluctuate sometimes. We believe this is because the real error is already quite small and close to the optimal. Note that even the best possible pseudo random number generator and min-wise hashing implementation are approximations to the theoretical ideals.

### 4.4.4 Summary

The main characteristics of the tested algorithms can be summarized as follows:

- *All-Pair* (Straightforward): Running time is predictable but can be too slow for large datasets; directly applicable for other similarity measures; running time increases in a quadratic rate with the number of objects in the dataset; can be still viable in practice for smaller datasets.
- *SampleDJ*: Extremely fast for the real datasets we used regardless of the dataset size; directly applicable for other similarity measures; can be too slow in some cases depending on the diversity value; running time is not predictable without prior knowledge about the input. A practical solution might be to try SampleDJ first and use TrackDJ in case of failure.

**Table 4.2** Running times, RDJ value and error for All-Pair, SampleDJ and TrackDJ for synthetic dataset to test effect of data characteristics and error bounds.

**(a)** Effect of RDJ value

| | #ofGroups:GroupSize | All-Pair | | SampleDJ | | TrackDJ | |
|---|---|---|---|---|---|---|---|
| n | G:Gs | RDJ | Time(hours) | Error(%) | Time(seconds) | Error(%) | Time(hours) |
| 524,288 | 32:16384 | 0.017 | 5.3 | 0.34 | 2 | 2.05 | 2.5 |
| 524,288 | 64:8192 | 0.0087 | 5.3 | 0.26 | 10 | 1.96 | 2.5 |
| 524,288 | 128:4096 | 0.00427 | 5.3 | 0.38 | 39 | 2.00 | 2.5 |
| 524,288 | 256:2048 | 0.00217 | 5.3 | 0.02 | 156 | 1.95 | 2.5 |
| 524,288 | 512:1024 | 0.00105 | 5.3 | 0.06 | 624 (10 min) | 1.90 | 2.5 |
| 524,288 | 1024:512 | 0.00052 | 5.3 | 0.13 | 2,502 (42 min) | 1.91 | 2.5 |
| 524,288 | 2048:256 | 0.00026 | 5.3 | 0.04 | 10,089 (3 h) | 1.91 | 2.5 |
| 524,288 | 4096:128 | 0.00013 | 5.3 | 0.39 | 40,635 (11 h) | 2.31 | 2.5 |

**(b)** Effect of error bound $\epsilon$

| Error Bound | SampleDJ | | TrackDJ | |
|---|---|---|---|---|
| $\epsilon$ | Time(seconds) | Error(%) | Time(seconds) | Error(%) |
| 0.2 | 0.3 | 0.098 | 5 | 0.022 |
| 0.1 | 1 | 0.021 | 20 | 0.369 |
| 0.05 | 4 | 0.015 | 80 | 0.003 |
| 0.025 | 15 | 0.016 | 318 | 0.004 |
| 0.0125 | 58 | 0.003 | 1271 | 0.081 |

- *TrackDJ*: Predictable $O(n)$ running time regardless of the input; applicable for similarity measures for which there exist hash functions with the LSH property (requiring additional analysis with respect to performance guarantees); solves the $O\left(n^2\right)$ running time problem which makes the diversity index practically computable even for large datasets.

Note, that all three approaches can be parallelized (cf. Section 4.3).

## 4.5 Characterizing the Diversity in Corpora: Example Studies

We now show the results of sample studies on a number of real world datasets. *Note, that low RDJ values correspond to high diversity.*

### 4.5.1 Development of Flickr over Time

We also studied the temporal development of tag annotation diversity in the social photo sharing site Flickr over the time period from 01 Jan 2005 until 05 Sept 2010 (see Section 4.4.1 for a description of the Flickr sample).

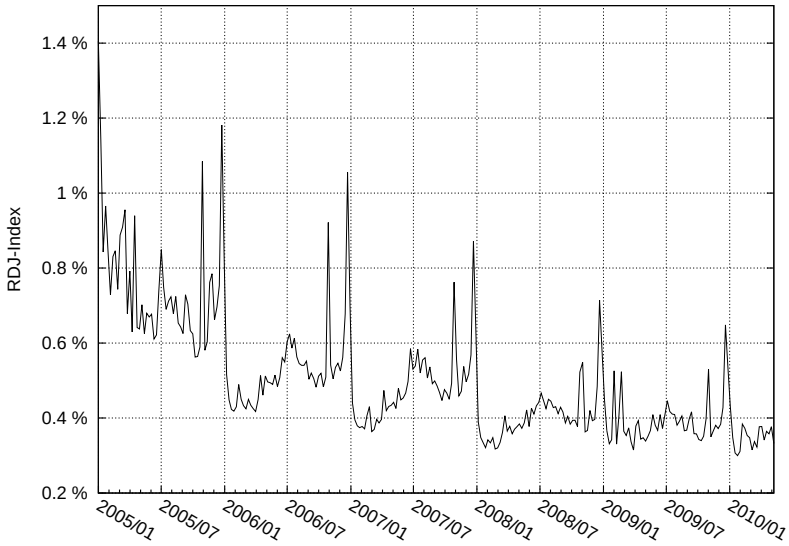



**Figure 4.1** Flickr photo tags similarity over the time period 2005-2010, revealing a trend towards higher annotation diversity throughout the years, as well as periodic peaks due to recurring annual events.

Our first observation from Figure 4.1 is a year wise increase in annotation diversity reflected by the decrease of the RDJ values. A possible explanation for the increasing topic diversity is that the user community has broadened over the years with Flickr becoming more and more popular for people of different ages, origins, and backgrounds.

Secondly, exploring diversity on a more fine-grained, monthly level reveals additional interesting patterns and periodicities. In order to obtain representative tags for the most interesting parts of the curve, we computed Mutual Information (MI) [MRS08] for each tag, which measures how much a joint distribution of terms and categories deviates from a hypothetical distribution in which terms and categories (time period for a peak and the rest of the year in our case) are independent of each other. Note that the number of photos captured for this figure remains rather constant over time, amounting to approx. 10,000 images per day. The RDJ value curve starts at its minimum at the beginning of each year (January) and remains at the lowest level till the end of March, reflecting high topic diversity. Photos in this time period are mainly described by tags for a rather broad range of topics like *winter*, *snow*, *vacation*, or *house*. In summer, the curve starts to increase steadily, and reaches its maximum in mid-July before it decreases steadily until the end of September, with the most representative tags in this time period being *graduation*, *wedding* and *beach*. In October the RDJ value increases sharply, reaching a peak by the end of that month. Our term analysis reveals that this is due to the popular holidays *halloween* and *thanksgiving*. Finally, the RDJ curve reaches its maximum at the end of December where Christmas is the dominating topic.

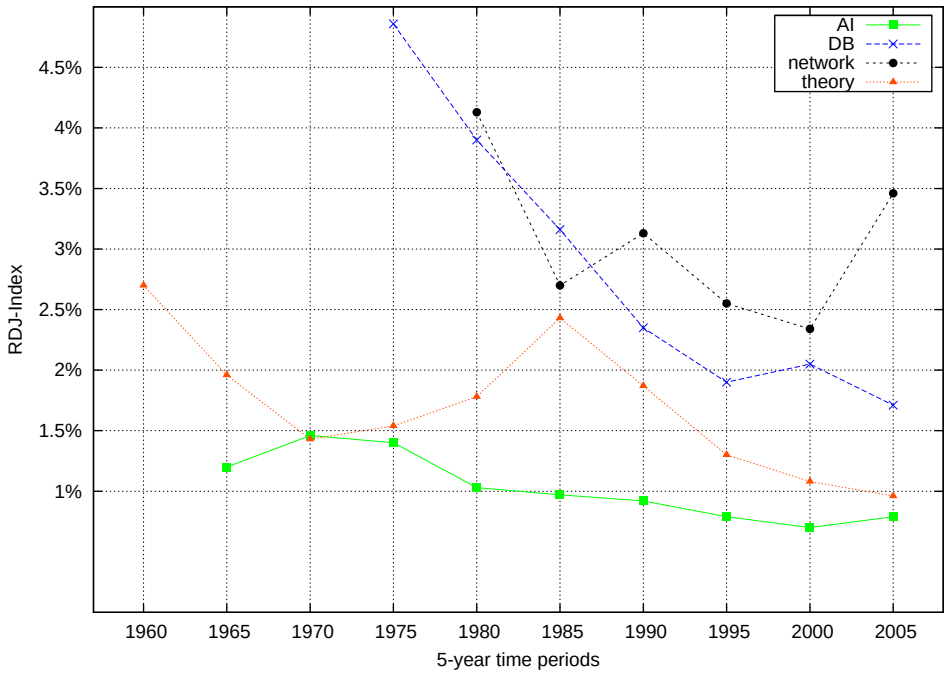



**Figure 4.2** Topic Diversity vs. 5-year Interval

### 4.5.2 DBLP Bibliography

To study diversity in the DBLP dataset, we classified Computer Science (CS) paper titles into groups based on publication year and publication venue. We intended to explore how topic diversities of Computer Science papers change with time and research area. We picked well-known publication venues to represent four research areas: Databases (DB), Artificial Intelligence (AI), Computing Theory and Computer Networks. The venues representing the areas are as follows: DB: *SIGMOD*, *VLDB*, AI: *AAAI*, *IJCAI*, Theory: *STOC*, *FOCS*, and Computer Networking: *SIGCOMM*, *INFOCOM*. Figure 4.2 shows how topic diversity changes with time. The figures indicate that the topic diversities of CS papers increased gradually over time with the exception of Network papers within the past 5 years. Also, curiosity driven papers (Theory, AI) seem to have higher topic diversities than application driven ones (DB, Networking).

### 4.5.3 Diversity in Clustered Data

Cluster analysis or "clustering" refers to the division of a set of objects into subsets (called clusters) so that objects in the same cluster are more similar and objects in different clusters are less similar. In many contexts unsupervised machine learning techniques like hierarchical, partitional or spectral clustering are employed to achieve this goal. Intuitively, the higher the number of clusters in a particular dataset, the higher its diversity. In this section we want to verify if this property is reflected by the RDJ index. To this end, we analyzed

**Table 4.3** Size, title and RDJ value (in percent) per category in Reuters Corpus, per group in Flickr-Groups collection, and per cluster in US-Census data

**(a)** Reuters RCV1 Categories

| Size | News Category | RDJ |
|---|---|---|
| 299,612 | Corporate/Industrial | 4.31 |
| 204,820 | Markets | 4.79 |
| 66,339 | Economics | 4.69 |
| 35,769 | Government/Social | 5.73 |
| 35,279 | Sports | 3.45 |
| 33,969 | Domestic Politics | 5.21 |
| 31,328 | War, Civil War | 5.81 |

**(b)** Flickr Groups

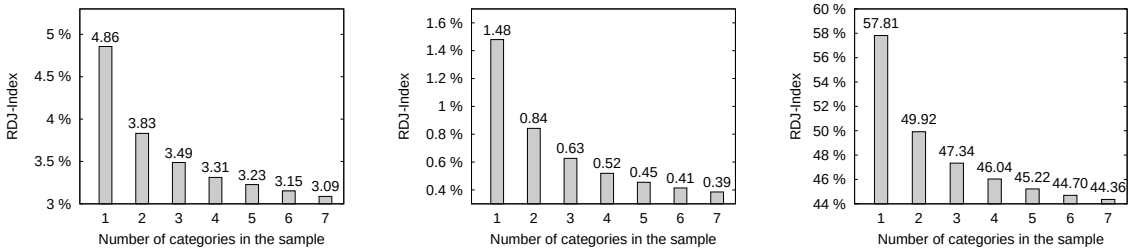
| Size | Group Title | RDJ |
|---|---|---|
| 139,344 | Pictures Of England | 1.63 |
| 121,391 | Dark Art | 0.57 |
| 98,901 | Aircraft Photos | 1.99 |
| 89,606 | Absolutely beautiful | 0.51 |
| 76,265 | Visual Arts!! | 0.61 |
| 73,632 | Lonely Planet:'Leaving' | 0.48 |
| 71,158 | Lighthouse Lovers | 4.56 |

**(c)** UCI US-Census Education Based Clusters

| Size | Educational Background | RDJ |
|---|---|---|
| 562,837 | High School, Diploma or Ged | 49.41 |
| 366,116 | Some College, But No Degree | 49.22 |
| 273,281 | 5th, 6th, 7th, or 8th Grade | 51.63 |
| 213,941 | Bachelors Degree | 51.36 |
| 174,653 | 1st, 2nd, 3rd, or 4th Grade | 70.97 |
| 109,834 | N/a Less Than 3 Yrs. Old | 84.55 |
| 107,142 | 10th Grade | 47.56 |

three real world datasets:

1. *RCV1:* The "Reuters Corpus Volume 1" (RCV1) is a corpus of news feeds released for public research in 2000 [LYRL04], and employed in several works on clustering and classification of text documents [SCK+06, MRS08, PSF10]. RCV1 consists of text documents divided into news categories like "sports", "politics" and "economics" which we used as cluster ID's.
2. *Flickr Groups:* The second dataset consists of tagged photos from different interest groups in Flickr like "Pictures of England" or "Aircraft Photos" and general groups like "Absolutely Beautiful" or 'Visual Arts". Here we used the group IDs as cluster IDs and image tags as textual content.




**(a)** Reuters RCV1 **(b)** Flickr Groups **(c)** UCI US-Census

**Figure 4.3** Diversity increases with a growing number of categories in Reuters Corpus, groups in Flickr-Groups, and educational levels in US-Census datasets

3. *US Census 1990:* In demographics clustering is defined as the gathering of various populations based on factors such as ethnicity, economics, or religion. For our experiments we used the US Census Data from 1990 [MTH02] consisting of details from around 2.5 millions US citizens[4]. Attributes include, for instance, *income*, *dedication* and *educational background*. We selected the latter for obtaining the clusters, with examples of cluster IDs being "High School Graduate", "Diploma", or "Bachelor Degree".

Table 4.3 provides an overview over the topics in the datasets and the number of instances per topic. For each dataset we selected the top $n = 7$ largest clusters. We computed sets of all $\binom{n}{k}$ possible cluster combinations for $k = 1, \ldots, n$ clusters. In order to obtain balanced cluster sizes, we restricted the number of instances, $s$, per combination to the size of the smallest cluster, and randomly selected $\frac{s}{k}$ instances from each cluster in the sample. For the RCV1, Flickr and UCI Census datasets we obtained 31,328, 71,158 and 107,142 instances, respectively. Finally, for each number of clusters $k$ we computed the average RDJ index value across all cluster combinations.

Figure 4.3 shows the RDJ index vs. the number $k$ of clusters contained in the sample sets. *The key observation is that the RDJ value indeed decreases with the number of clusters.* This shows that topic diversity in the sample sets is mirrored by the Jaccard-based RDJ index. Despite of the large structural and conceptual differences between the distinct corpora and the large differences between the absolute RDJ values, the decreasing pattern observed is remarkably similar across corpora.

A comparison of diversity values in different corpora and for specific clusters reveals further interesting insights (cf. Table 4.3). Generally, the diversity in the Flickr dataset is highest (corresponding to lower RDJ values) as the tags used for diversity computation can be defined by users and are not restricted. The Reuters dataset contains more restricted vocabulary, and is less diverse. Finally, attributes in the UCI Dataset are well defined, the number of possible terms per instance is small (68), and the "vocabulary" limited, which results in high RDJ values. We also studied the correspondence of RDJ diversity values to different topics. For example in Table 4.3a the first five categories including "Industrial" (*RDJ=4.31*) or "Markets" (*4.79*) are more general and therefore more diverse (an exception being "Government/Social" with (*RDJ=5.73* )). "Domestic Politics" (*5.21*) and "War, Civil War" (*5.81*) are sub-categories that are more specific and less diverse. The Flickr groups "Visual Arts" (*0.61*) and "Absolutely beautiful" (*0.51*) are more general than "Lighthouse Lovers" (*4.56*) and "Aircraft Photos" (*1.99*). Finally, for the census data, we observe that the group "N/a Less Than 3 Yrs. Old" (*84.55*), which mainly represents babies, has the lowest diversity.

[4]The set is available in the UCI machine learning repository (http://archive.ics.uci.edu/ml/index.html).

## 4.6 Discussion

In this chapter we addressed Problem 2 of efficient analysis of topical diversity in very large collections of data. The topical diversity measure enables comparison of such datasets in terms of their topical similarity and can additionally be used to gain insights into the structure of a particular collection. Existing state-of-the-art methods are performing with quadratic complexity, making analysis of large corpora impossible.

As a solution we proposed two novel, efficient algorithms for estimating the diversity of whole corpora with probabilistic guarantees, and provided illustrative example studies on different datasets. Our first method TrackDJ makes use of a min wise hashing properties and can be executed in a linear time. The second method SampleDJ even requires constant time with respect to the dataset size. In this context we are the first tackling the problem of efficiently computing the diversity of whole corpora. We evaluated our methods on real world datasets such as scientific publication directory DBLP with over one million paper titles and metadata from 144 million Flickr images. Additionally, we constructed synthetic datasets with different diversity values. The conducted experiments confirm our theoretical findings; for example, SampleDJ was able to accurately compute the diversity value for 20,000,000 records in about four minutes whereas the state-of-the-art algorithms would spend more than a year for this computation.

Additionally, the proposed RDJ index can be used for computing efficient heuristics for determining the number of clusters in large datasets, which is required as a parameter in many clustering techniques such as k-means [Jai10], k-medoids or the expectation-maximization algorithm for clustering. The increased efficiency make more real-time applications possible such as diversity computation in large streaming scenarios or handling large datasets not fitting into memory. It may be interesting to adapt proposed algorithms to different similarity measures like Euclidean distance and cosine similarity by using LSH and its variants. Furthermore, it would be interesting to study properties of distributed versions of the described algorithms. Multiple machines provide more resources for solving the problem, but how to partition the dataset so that running time and network communication is minimized may require further study. Further, we aim to extend our techniques to other multimedia types such as videos or photos, where diversity based on visual properties (e.g. color distributions or shapes) may reveal further insights about social content sharing environments such as Flickr or YouTube. Finally, as discussed at the beginning of the chapter, careful clustering of an inverted index according to access policies can improve the retrieval efficiency under privacy constrains.

# 5 Privacy-Oriented Content Analysis

In the previous chapters we assumed that the sensitivity of a particular document depends only on its frequency in the dataset. In this chapter we address Problem 3 and show how the sensitivity can be estimated not only for textual, but also for multimedia content using machine learning techniques. In this chapter we tackle the problem of supporting users in making privacy decisions in the context of document sharing (especially for large batch uploads of multimedia documents). User uploaded contents are typically very topically diverse. Obviously decisions about privacy degree of each object can be done in two steps: first identify the object type and topic context - was this photo taken at the beach or in a house, does it show a sport event or a wedding. and second - estimate the sensitivity degree of those topic context. Since each of those steps can introduce certain error, we propose a machine learning algorithm to directly estimate the privacy degree based on textual and visual features of a multimedia object. Moreover, we develop algorithms for privacy-oriented search and privacy-based diversification. We are aware that building alarm systems to support sharing decisions of private content and enabling privacy-oriented search can be seen as contradicting goals; privacy-oriented search is not negative per se, as it can be used for retrieving private content users are comfortable to share, and, more importantly, can help with the early discovery of privacy breaches. However, as with almost every technology, it requires sensible handling and constructive usage.

## 5.1 Contributions

Our first goal in this chapter is to provide techniques to automatically determine adequate *privacy settings* for newly shared documents. To this extent, we exploit an average community notion of privacy gathered through a large-scale social annotation game. We then employ the obtained community feedback to build classification models on selected visual features and textual metadata, and apply these models to estimate adequate *privacy settings* for newly uploaded images and search results. Such alert systems could be directly integrated into social photo sharing applications like Flickr or Facebook, or provided as a browser plugin.

The second aspect that we study in this chapter is *privacy-oriented search*. Existing systems do not enable users to directly search for private content in a systematic way. The motivation for enabling privacy-oriented search is two-fold: First, users should be able to retrieve resources about themselves (or about their children or other relatives) published by third parties at an early stage, so that measures such as contacting owners of servers or providers can be taken. Second, the degree of privacy of the information need behind a query can be ambiguous for a search engine. For example, a user querying "*ronaldo*" could be interested in photos showing either professional or private life of this football star. In this work, we use classifier outputs to conduct privacy-oriented search, which enables users to directly discover private information about a specific topic (see Figure 1.1 for an example).

Finally, we develop algorithms to perform *privacy-based diversification* of search results (i.e. retrieving a "mixture" of private and public content) to minimize the risk of user dissatisfaction in cases where queries are ambiguous with respect to the privacy aspect of the information need. The motivation for this is analogous to topic-related diversification [GS09]: to cover different information needs and provide an overview over the whole search result space rather than just a list of top-ranked results.

## 5.2 Data

An individual's notion of privacy is prone to continuous change. For example, something a 16 year old youth may not consider sensitive may later lead to a rejection of his job application. Therefore, independent opinions can be very useful in order to support the user for adequate privacy judgment, and we build on an average community notion of privacy collected in this study. Our intent is to automatically suggest images with potentially sensitive content (and leave the final decision to the user), rather than images that a particular person would not share with the world.

In order to obtain an appropriate dataset with labeled private and public image examples, we performed a user study in which we asked external viewers to judge the privacy of the photos available online. To this end, we crawled 90,000 images from Flickr[1], using the "most recently uploaded" option to gather photos uploaded in the time period from January to April 2010. As we planned to investigate whether textual annotations can help us automatically selecting suitable privacy settings for an image, we only crawled images annotated with at least five English tags.

Of course, Flickr does not provide access to photos explicitly declared as private by other users; therefore, in this work we focus on private images that are publicly available on the Web. However, we think that classification and feature engineering techniques employed in this work are general enough to be also applied to learn classification models from hidden private content.

[1] until 2011

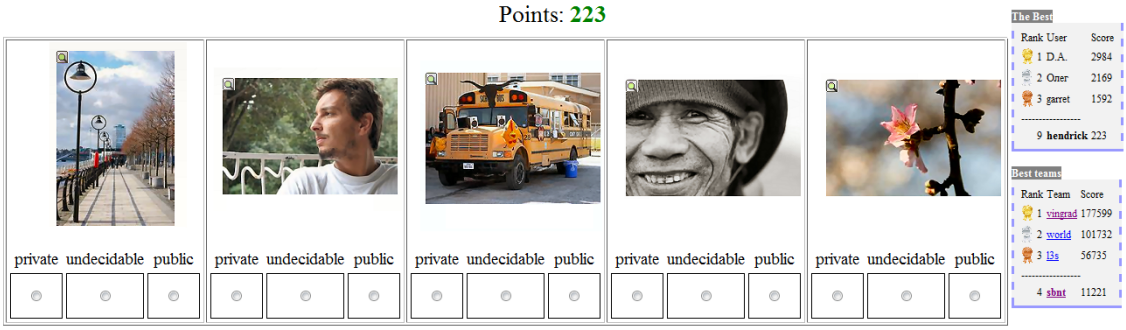



**Figure 5.1** User interface of the privacy game. The user marks each photo as either "private", "public" or "undecidable". The system displays the current score of the user and the scores of the competing user teams.

As labeling a large-scale image dataset requires considerable manual effort, we conducted the user study as an annotation game (cf. Figure 5.1). Social annotation games are popular tools for solving tasks that are difficult for machines, and easy for human users, such as image labeling [vAD04]. At each step of the game we presented five photos to a participant of the study. For each photo, the participants had to decide if, in their opinion, the photos belonged to the private sphere of the photographer. To this end, we asked the participants to imagine that images presented to them were photos they took with their own cameras, and mark these images as "private", "public", or "undecidable". We provided the following guidance for selecting the label: "*Private* are photos which have to do with the private sphere (like self portraits, family, friends, your home) or contain objects that you would not share with the entire world (like a private email). The rest is *public*. In case no decision can be made, the picture should be marked as *undecidable*." [2] If the user could not decide at first glance, she could choose to display metadata of the photo or use a link to see the photo in its original context on Flickr. To obtain sufficient evidence for the privacy label assignments, each photo was shown to at least two different users. In case of a disagreement, the photo was queued to be shown to additional users. Users gathered points for each classified photo. As incentive for providing accurate answers, credit was not just given for the number of labeled images, but also for inter-user agreement. The more similar the choices of the user and the other players (their assessments, of course, not being visible to the user in advance), the higher was the score of the user. Our privacy game had an unlimited number of levels and was online over the course of two weeks.

[2] All rights on pictures published in this document are reserved to the image owners. The pictures were available on Flickr in May 2012. The pictures are supplied with their Flickr ids. These ids can be used for obtaining further legal information.

We asked our participants to complete at least one level of the game with 1000 pictures; in addition, the users could continue playing to gather more points for themselves and their teams. Over the course of the experiment, 81 users between ten and 59 years of age worked in six teams. For example, one of the teams consisted of graduate computer science students working together at a research center; other teams contained users of social platforms including Facebook and some online forums[3]. Altogether 83,820 images were annotated[4]. After cleaning (remove unregistered users and photos annotated by less than two different people) 37,535 images remained. This set was used for our experiments.

Our analysis revealed that in the cleaned dataset around 70% of photos were labeled as public or undecidable by all of the participating judges. This relatively high percentage is expected as we originally crawled only images publicly available on the Web. Around 13% of the images were labeled as "private" by all the judges, and 28% received "private" votes from at least one of the judges. Overall the dataset contained 4,701 images labeled as private, and 27,405 ones labeled as public by 75% of the judges, which we use as ground truth for our classification experiments in Section 5.5.1. Thus, depending on the voting-based threshold applied, our sample indicates that a publicly available set of images from Flickr can contain 13-28% of private images.

We computed the inter-rater agreement using the Fleiss $\kappa$-measure [Gwe10], a statistical measure of agreement between individuals for qualitative ratings. Note that according to Fleiss' definition, $\kappa < 0$ corresponds to no agreement, $\kappa = 0$ to agreement by chance, and $0 < \kappa \leq 1$ to agreement beyond chance. We measured $\kappa$ on a randomly selected subset of 100 images from the ground truth set labeled by 36 users, where every user rated each of those images, and obtained $\kappa$=0.6.

## 5.3 Features

Image privacy is a very subjective concept. It is influenced by a number of different factors such as place, objects and persons on the photo as well as the point in time the photo was taken.

Content-based features of an image like the presence of edges, faces, or other objects may give some insights about its degree of privacy. In particular, in this work, apart from textual features such as title and tags, we have selected image features which can help discriminate between natural and man-made objects/scenes (the EDCV feature) and features that can indicate the presence or absence of particular objects (SIFT). We have also selected features that measure the colors within the image (color-histograms), and a feature that measures the presence or absence of faces within the image.

Visual features and associated metadata are later used for training of models for privacy classification. Note that, similar to other machine learning scenarios, "hints" provided

[3] facebook.com, forum.vingrad.ru, forum.sbnt.ru and others.

[4] The anonymized data as well as some of the applications based on the described classification techniques are presented at http://l3s.de/picalert

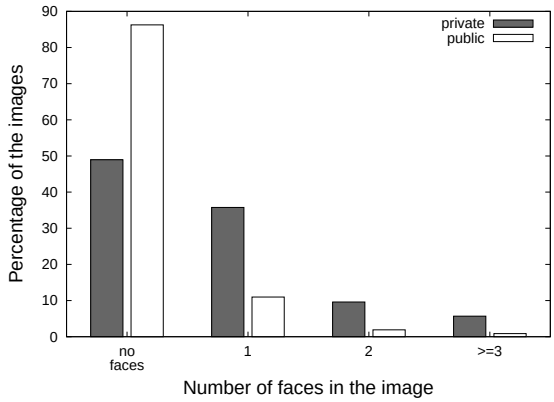



**Figure 5.2** The distribution of human faces among private and public photos

by individual features do not suffice for predicting the privacy character of all photos; however, in Section 5.5.1 we will show that the *combined evidence* of multiple features yields applicable results.

### 5.3.1 Visual Image Privacy Features

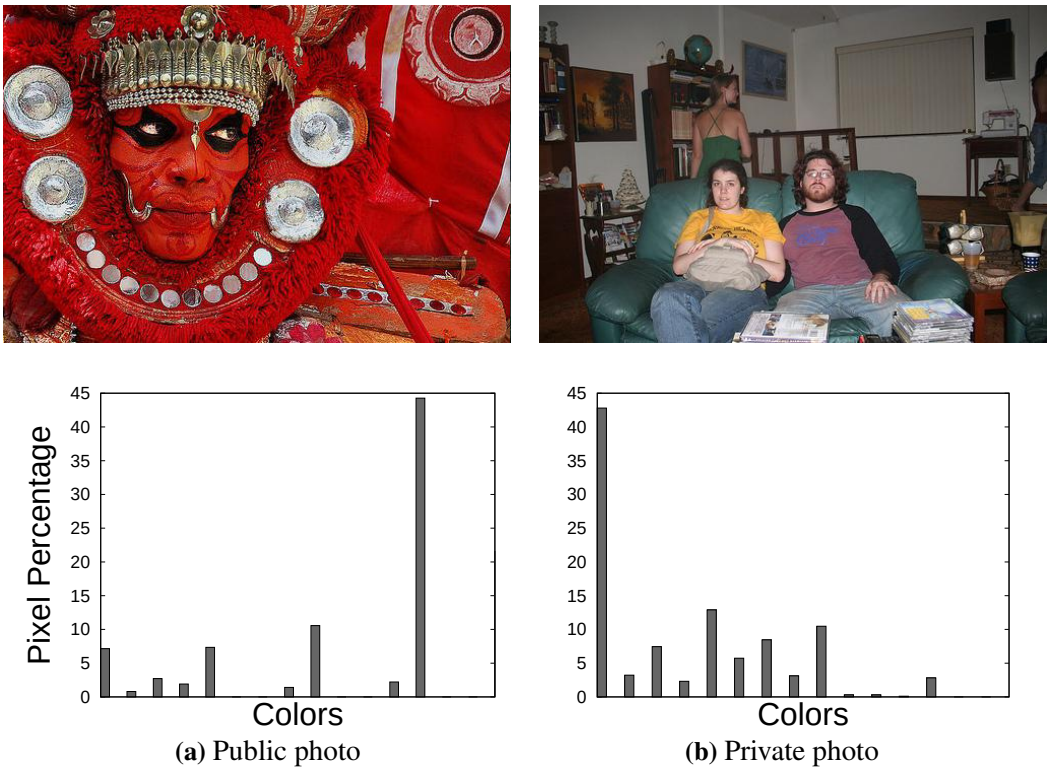
Digital images are internally represented as two-dimensional arrays of color pixels. This representation is difficult to use directly for classification because it is highly multidimensional and subject to noise. Instead, a process known as feature extraction is typically used to make measurements about the image content. Image features come in many forms, from the very low-level, to so-called high-level features. Low-level features are typically formed from statistical descriptions of pixels, whilst high-level features are those that have a directly attributable semantic meaning. For this study, we have selected a range of image features that could potentially be used in building a classifier that can discriminate public and private images automatically.

In the following descriptions of the visual features, we attempt to discuss theoretical motivations for the use of the particular feature, and also describe some observations from the use of the features for classification experiments within our privacy image dataset. Please note that many of the illustrations need to be viewed in color (with a quality printer or as PDF) in order to understand them fully.

#### Face detection

We first study the hypothesis that there might be a relationship between the occurrence of faces in an image and the background of the scene with respect to privacy. Figure 5.2 denotes the correlation between the number of detected faces in a photo and its privacy value in our dataset.

We observe that the occurrence of faces in a picture is strongly associated with a high degree of privacy although a considerable number of faces also can be found in public images. The detection of faces within an image is a high-level feature extraction operation.

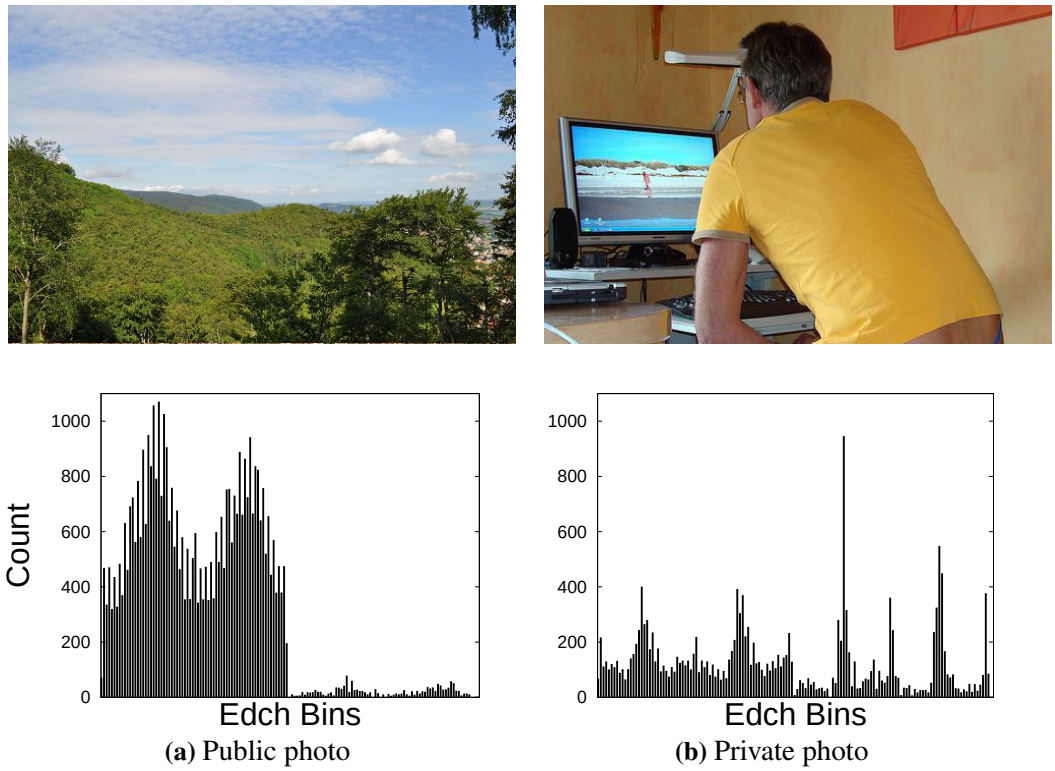



**(a)** Public photo **(b)** Private photo

**Figure 5.3** Example of a public photo with a few dominant colors and a private photo.

**Figure 5.4** The top row show the most discriminative colors for public images and the bottom row for private images from our dataset. The colors have been arranged by hue to aid visualization. It can be seen that the discriminative colors of public images are more saturated (i.e. more vibrant) than for private images.

In order to find faces in images we used an implementation of the Viola-Jones face detector [VJ02], which uses a boosted cascade of weak classifiers to detect frontal and profile faces within the image. Two types of summary feature were generated using the face detector; the first is a single numeric feature that counts the number of faces detected within the image. The second is a variable-length vector that encodes the relative area of the bounding box around each detected face. Note that private photos do not necessarily show persons (e.g. photos of letters and private workspaces); furthermore, faces are partially not recognized from certain angles or if parts of the faces are hidden.




**(a)** Public photo **(b)** Private photo

**Figure 5.5** Example of a public photo dominated by incoherent edges and a private photo of a working place with a mix of coherent and incoherent edges.

### Color Histograms

Intuitively, color may be an indicator for certain types of public and private image. For example, public images of landscape scenes are very likely to have a specific color distribution.

The global color histogram of an image is one of the simplest forms of image feature. The histogram is constructed by assigning each pixel in the image to a histogram bin based on its color. The global color histogram completely discards all information about how the colors appear spatially within the image.

In this study, the histogram was calculated in the HS (Hue, Saturation) color space with each color-dimension split evenly into four segments, resulting in a histogram consisting of 16 bins.

Previous studies have shown that skin-tones across ethnicities are tightly clustered in HS space [RMG98], so HS histograms should be particularly good at efficiently modeling skin tones and predicting the presence or absence of people, which can be observed to be a high indicator of privacy. We also observed, as shown in Figure 5.3 that public photos often contain a few very strong discrete colors (resulting in a *spiky* histogram) whereas in private photos the colors are distributed more uniformly. The former might correspond

to artistic make up and an artificial environment, whilst the latter is typical for amateur photos. More specifically, as shown in Figure 5.4, which was generated from the weights of the classifiers discussed in Section 5.4.1, we perceive that public images tend to contain more fully saturated colors (vibrant colors), whilst the colors in private images tend to be more desaturated. We also observed that black and white photos correspond to private portraits in most cases.

### Edge-Direction Coherence Vector

The edges within an image are a very powerful feature for discriminating between different types of scene, and thus might be useful for privacy classification.

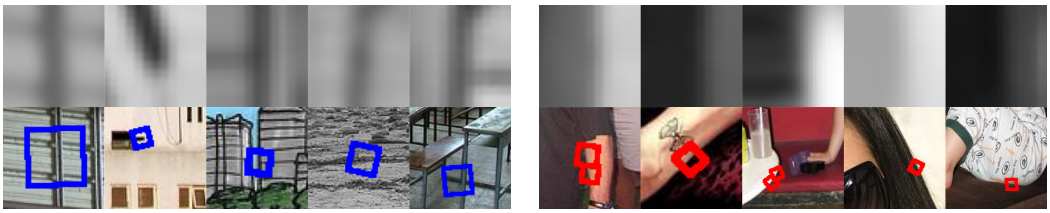
In particular it has been shown previously that edge-based features can be a particularly good discriminator of indoor and outdoor images [KPK10]. Images taken in artificial environments tend to be dominated by strong, straight, near-vertical lines, whilst those of natural environment tend to predominantly contain shorter and weaker edges in all directions [VJZ98]. The check of the top 100 positive and negative images, correctly classified using the edge feature revealed that just 20% of private photos were taken outdoor whereas 11% of public ones were taken indoor.

The edge-direction coherence vector (EDCV) [VJZ98] consists of a pair of histograms that encode the lengths and directions of edges within the image. The first histogram encodes incoherent edges, whilst the second histogram encodes coherent edges. Coherent edges are edges within the image that are relatively straight and long, whilst incoherent edges are short and/or non-straight. Each bin of the histograms represents a small range of edge directions, whilst the magnitude of the bin is proportional to the summed lengths of all the coherent (or incoherent) edges with that particular direction in the image. We implemented the algorithm described in [THD+06], where edges and their directions are found using the Canny edge detector. Coherent edges are those that deviate in direction by less than five degrees over their length, and have a pixel length greater than 0.002% of the image area. In terms of privacy classification, the EDCV feature of public images depicting landscape scenes is likely to be dominated by incoherent edges. Private images are much more likely to contain a mixture of both coherent and incoherent edges from a mixture of the presence of people (mostly incoherent) and the background (coherent).

This is illustrated in Figure 5.5 where the two histograms are shown as one larger histogram with the incoherent vector on the left and coherent vector on the right. Public images of cityscapes, that contain some natural features (i.e. trees or people) would also contain a mix of both coherent and incoherent edges, however, there might be some changes in the distribution of these edges with respect to their orientation.

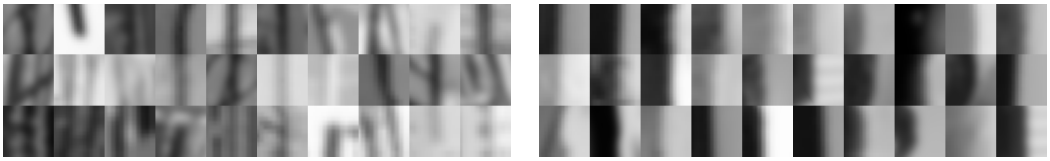
**(a)** Public photo **(b)** Private photo

**Figure 5.6** Examples of public and private photos. The most dominating features are surrounded with boxes; the brightness of the boxes is proportional to the strength of the features.



**(a)** Public features **(b)** Private features

**Figure 5.7** Some discriminative public and private SIFT features along with example snippets from the original images.



**(a)** Public features **(b)** Private features

**Figure 5.8** Top-30 discriminative public and private SIFT features

### Quantized SIFT Features

Private and public photos typically tend to be taken in specific contexts. For example, pictures can be taken in public places like stadiums, supermarkets and airports, or in private places like home, car, or garden. Accordingly the objects contained in a photo, like sport equipment, furniture, human and animal body parts could be different and thus give us some insights about an image's privacy. Examples of features dominating the classification decision using the machine learning model described in Section 5.4.1 are visualized in Figure 5.6. Recent advancements in low-level image description have led to approaches that are based on the detection and description of locally interesting or salient regions depict-

**Table 5.1** Top-50 stemmed terms according to their Mutual Information values for "Public" vs. "Private" photos in Flickr

| Terms for Private photos | | | | |
|---|---|---|---|---|
| studio | photograph | sexi | strobist | februari |
| blond | kid | hat | love | woman |
| year | smile | black | eye | famili |
| white | fun | portrait | selfportr | peopl |
| dress | valentin | day | friend | parti |
| femal | boy | birthday | face | model |
| fashion | daughter | children | coupl | self |
| babi | cute | child | girl | beauti |
| me | nikon | pretti | hair | man |
| photographi | young | lip | happi | wed |

| Terms for Public photos | | | | |
|---|---|---|---|---|
| craft | mountain | natur | rock | draw |
| sky | tree | snow | jan | flower |
| januari | art | paint | lake | winter |
| view | citi | car | moon | landscap |
| anim | cat | road | north | build |
| frozen | island | full | ice | bridg |
| water | abstract | hdr | park | sea |
| hous | boat | fire | bird | handmad |
| architectur | sunset | sunris | cloud | macro |
| night | dog | illustr | design | river |

ing objects and textured regions. Many region detection and description approaches have been proposed, however, the SIFT [Low04] descriptor remains the most popular descriptor currently. The SIFT descriptor is a 128 dimensional vector that describes local texture. Because the application of an interest point detector can lead to a variable number of SIFT features for different images, a popular technique for using SIFT features for classification is to quantize the features into a small vocabulary of "visual terms", and to create sparse, fixed length vectors for each image that encode the number of occurrences of each type of visual term. In this study, we detected interest regions in each image by detecting peaks in a difference-of-Gaussian pyramid [Low04]. Approximate $k$-means [PCI+07] was used to learn a 12000 term codebook for vector quantization from a random sample of one million SIFT features from the dataset.

SIFT-based features are commonly used for generic object recognition because similar combinations of SIFT features will often occur in different images of the same class of object. As public and private images often have very different types of objects depicted in them, SIFT-based features have the potential to be a powerful discriminator. Figure 5.7 shows some examples of the discriminative public and private visual terms within the context of an image from which the term occurs. In order to obtain the most discriminative

features, we computed the Mutual Information (MI) measure [MRS08] from information theory, which measures how much the joint distribution of features (visual terms) and categories deviates from a hypothetical distribution in which features and categories ("private" and "public" in our case) are independent of each other. Figure 5.8 visualizes the top-30 discriminative SIFT features for public and private images. Typically, SIFT features correlated to public images are observed to be of highly textured regions, whilst those corresponding to private images tend to depict more linear features such as edges. More specifically, as can be seen from Figure 5.7, SIFT features corresponding to public features tend to come from regions containing text and symbolic shapes, and also from patterns like in the case of an image containing flowers. Conversely, many SIFT features associated with private features correspond to human body parts or items like furniture, typically occurring in private environments.

#### Brightness and Sharpness

We also studied features like brightness and sharpness that have been shown to be an indicator of photo attractiveness [SPS09]. In the same study, private amateur photos were also not found very attractive by a community of users. Our *brightness* feature is a single valued number that is calculated by averaging the luminance values of all the pixels in the image.

### 5.3.2 Textual Image Privacy Features

In addition to visual features, the textual annotation of images available in Web 2.0 folksonomies such as Flickr can provide additional clues on the privacy of photos. This holds partly due to correlations of topics with privacy-related image content.

Table 5.1 shows the top-50 **stemmed** tags automatically extracted from our labeled dataset of private and public photos described in Section 5.2 using MI in a similar way as in Section 5.3.1. Obviously the majority of tags used in images falling into the private category describe personal concepts like family (babi, famili, child), emotions and sentiment (happi, love, beauti), and concepts related to the human body (hair, face, eye), whereas tags in the public category mostly refer to nature motives (snow, sunset, tree), architecture (bridg, build, architect), and inanimate objects (car, rock). This result indicates that privacy decisions can be correlated with specific terms found in tag annotations, and illustrates the potential merit of additional textual metadata for privacy prediction tasks.

## 5.4 Privacy Explorer

In this section we describe the Privacy Explorer system, which provides two types of privacy-based search mechanisms: 1) *privacy-oriented search* retrieving the most private search results, and 2) *privacy-based search result diversification* providing an overview of search results with varying degree of privacy.

### 5.4.1 Classification Model

In order to predict the privacy of photos we use a supervised learning paradigm which is based on training items (photos in our case) that need to be provided for each category. Both training and test items, which are later given to the classifier, are represented as multidimensional feature vectors. These vectors can be constructed using $TF$ or $TF \cdot IDF$ weights of tags or titles, and the visual features described above. Photos labeled as "private" or "public" are used to train a classification model, using probabilistic or discriminative models (e.g., SVMs).

In our classification experiments we use linear support vector machines (SVMs) classifiers. SVMs construct a hyperplane $\overrightarrow{\omega} * \overrightarrow{x} + b = 0$ that separates the set of positive training examples (photos manually tagged as "private" in our case) from a set of negative examples (photos tagged as "public") with maximum margin. SVMs have been shown to perform very well for various classification tasks [MRS08].

For combining multiple features, a normalization step for the input vectors is needed, since the ranges of absolute values for different features can vary and are not directly comparable. One possible normalization technique is to employ a trained sigmoid function which maps feature values from arbitrary range into the range $[0, 1]$. After this transformation the dimensions of the different input vectors are in the same range and can be combined. SVMs provide as output a value which determines the distance of a new test image from the separating hyper plane. We used Platt's method [Pla99] to transform this output into a probability value.

### 5.4.2 Privacy-Oriented Search

In order to enable users to get an overview of the most private images relevant to a query, privacy-oriented search retrieves a set of relevant images, and re-orders them according to descending degree of privacy (see Figure 1.1 in the Introduction). In order to create a list of images ranked by privacy, we can estimate the likelihood of image privacy using the output of the SVM classifier trained on a set of images labeled as "public" or "private" by the users. We described the training process of the classifier and feature selection in Section 5.3.

### 5.4.3 Privacy-Based Diversification

A privacy-aware image search engine should provide an overview of the available search results, taking into account their privacy. For example, for a query "wedding make-up pictures", users might want to retrieve some professional images from wedding catalogs, examples of private wedding photos, or recent wedding photos of the user's personal friends available on Web 2.0 platforms. In contrast to Web image search, which focuses on a few relevant results, privacy-based diversification can minimize the risk of user's dissatisfaction in such scenarios. In what follows, we describe a framework to handle diversity in image

search with respect to the privacy dimension. This framework enables us to obtain result sets of images covering a broad range of privacy degrees from publicly available images to private photos of personal friends.

### Diversification Quality and Estimates

$\alpha$-nDCG presented in Section 2.1.4 is the standard evaluation measure in the context of topic-based search result diversification. We now describe how $\alpha$-nDCG can be adapted to the privacy-based diversification. In topic-based diversification, information nuggets represent some mutually independent binary properties of documents [CKC+08]. In contrast, probability of image privacy has a continuous range of values. In our context, we interpret the notion of information nugget $n_i \in [0, 1]$ as a numeric property of the image with $n_i = 1$ being the most private. In our user study the values of the privacy nuggets are directly judged by the users. For the automatic diversification we used the classifier output described in Section 5.4.1 as estimate for the privacy nuggets values.

A document can contain information nuggets related to different topics. In the context of image privacy, a nugget $n_i$ represents the privacy degree of an image. For example, let $n_1 = 0.8$, $n_2 = 0.9$, and $n_3 = 0.3$ be three such privacy nuggets. Intuitively, the similarity between $n_1$ and $n_2$ is higher, than the similarity between $n_1$ and $n_3$. Then, the gain of viewing the sequence of images containing the nuggets $n_1, n_2$ is lower than the gain of viewing $n_1, n_3$.

In this scenario we cannot assume privacy-related information nuggets in different search results to be mutually independent. Therefore, we propose a more general $\alpha$-nDCG-G measure that takes into account similarity of the information nuggets contained in the $k^{th}$ result to the information nuggets contained in the higher ranked search results:

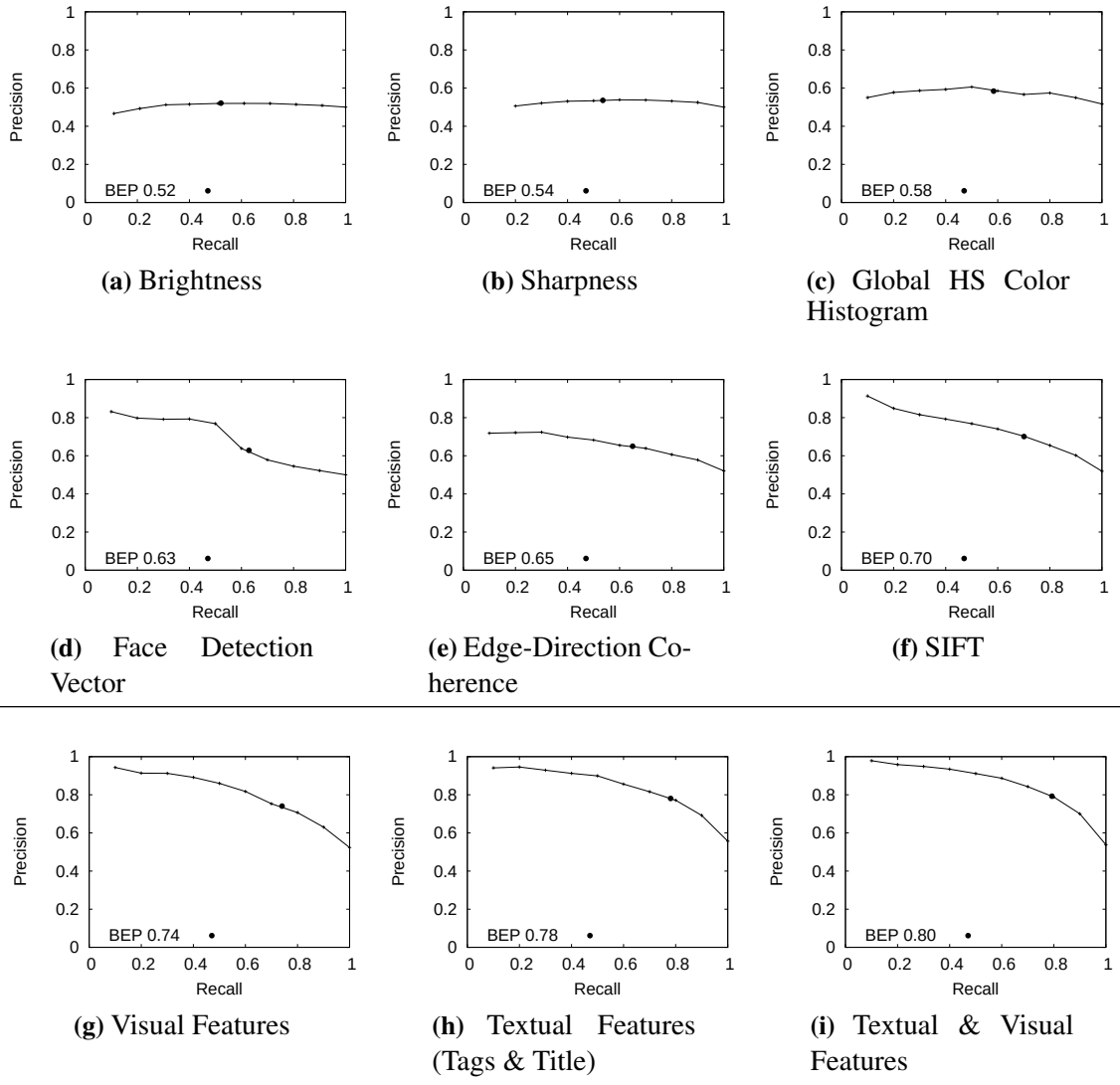
$$G[k] = \sum_{n_i \in N} J(d_k, n_i) \times (1-\alpha)^{\sum_{j=1}^{k-1} \sum_{n_y \in N} J(d_j, n_y) \times \mathrm{sim}(n_i, n_y)}, \tag{5.1}$$

where in the equation $G[k]$ is the $k^{th}$ element of the gain vector, $N = \{n_1, \ldots, n_m\}$ is the space of the possible nuggets, $J(d_k, n_i)$ is a binary judgment of containment of nugget $n_i$ in search result $d_k$, $\alpha \in (0, 1]$ is a factor to trade-off relevance and novelty of a search result, and $\mathrm{sim}(n_i, n_y)$ is the similarity between the nuggets $n_i$ and $n_y$.

In general, Equation 5.1 relaxes the nugget independence assumption of the standard $\alpha$-nDCG. This equation can be applied to estimate quality of a search result in presence of information nuggets with continuous range of values. As in this work we perform diversification only with respect to the privacy dimension and image $d_k$ only contains one particular privacy nugget $n_k$, we can simplify Equation 5.1 for this particular scenario as follows:

$$G[k] = (1-\alpha)^{\sum_{j=1}^{k-1} \mathrm{sim}(n_k, n_j)}, \tag{5.2}$$

where $n_k$ and $n_j$ are the privacy nuggets in images $d_k$ and $d_j$ respectively. A possible estimate for similarity of the privacy nuggets $n_k$ and $n_j$ is the difference in their privacy values: $\mathrm{sim}(n_k, n_j) = 1 - |n_k - n_j|$.




**(a)** Brightness **(b)** Sharpness **(c)** Global HS Color Histogram

**(d)** Face Detection Vector **(e)** Edge-Direction Coherence **(f)** SIFT

**(g)** Visual Features **(h)** Textual Features (Tags & Title) **(i)** Textual & Visual Features

**Figure 5.9** Precision-recall curves for textual and visual features.

### Diversification Algorithm

For automatic diversification, nugget privacy values $n_k$ and $n_j$ required for the gain computation in Equation 5.2 are computed with probability estimates obtained through the classification methods described in Section 5.4.1.

Let $L$ be the list of top-k images sorted by the probability of their relevance to the user's information need. Analogous to [AGHI09, GS09] Algorithm 8 starts with the most relevant image at the top of $L$, and selects subsequent candidates by greedily optimizing the estimated gain in Equation 5.2. The worst case complexity of Algorithm 8 is $O(l \times r)$, where $l$ is the number of images in $L$ and $r$ is the number of images to be selected for the result list.

**Algorithm 8:** Proc Select Diverse Images

**Require:** list $L$ of top-k images ranked by relevance, number $r$ of diverse images to be selected.

**Ensure:** list $R$ of the relevant and diverse images.

$R \leftarrow \emptyset$
**while** $|R| < r$ **do** {less than $r$ elements selected}
  $i^* \leftarrow \text{argmax}_{i \in L \setminus R} G[i]$
  $L \leftarrow L \setminus \{i^*\}$
  $R \leftarrow R \cup \{i^*\}$
**end while**

## 5.5 Evaluation

We now elaborate on the quantitative results of different experiments for classification, privacy-oriented search, and diversification using the visual and textual features motivated in the previous sections.

### 5.5.1 Classification

For our classification experiments, we used the SVMlight [Joa99] implementation of linear support vector machines (SVMs) with default parameterization.

Our dataset described in Section 5.2 contained 4,701 images labeled as private, and 27,405 ones labeled as public by 75% of the judges. In order to obtain a balanced set, we randomly restricted the set of public photos to a subset of 4,701 images, so that the number of private images was equal to the number of public ones. From this set, we randomly selected 60% as training data for building our classifiers, and 40% as test data, with each dataset containing an equal proportion of public and private instances.

We conducted our classification experiments on balanced datasets, which is commonly done the literature (see e.g. [YHBO10, SPS09]) in order to capture general classifier properties independent of the a-priory class probabilities on a particular dataset. Our quality measures are the precision-recall curves as well as the precision-recall break-even points for these curves. The break-even point (BEP) is the precision/recall value at the point where precision equals recall, which is equal to the $F_1$ measure, the harmonic mean of precision and recall, in that case. The results of the classification experiments for selected visual features described in Section 5.3 and the combination of visual and textual features are shown in Figure 5.9. The main observations are:

- Visual features like sharpness, brightness and color histograms that have been shown to be relevant in the context of photo attractiveness classification in previous work [SPS09] do not achieve significant discriminative performance in our scenario.

- Experiments with object related visual features, however, presented in the second row, show a substantial improvement. The occurrence of faces in photos is an intuitive indicator for privacy, reflected by a BEP of 0.63 for the face feature. The edge-direction coherence feature performs slightly better, and achieves a BEP of 0.65. In our experiments, SIFT features outperform all of the other visual features (BEP = 0.70).
- The *combination of all available visual features* further increases the BEP to 0.74.
- The pictures we used for classification experiments, contained good quality textual metadata (e.g titles and at least three English tags). Thus the *text features* could provide a short but concise summary of the image content and result in a BEP of 0.78.
- Finally, the *combination of the visual and textual features* leads to an additional performance boost with a BEP of 0.80, showing that textual and visual features can complement each other in the privacy classification task. However, classification with only visual features alone also produces promising results, and can be useful if no or insufficient textual annotations are available as is the case for many photos on the web.

Note, that it is possible to trade recall against precision for sensitive applications. For instance, we obtain prec=0.88 for recall=0.6, and prec=0.93 for recall=0.4 for a combination of textual and visual features; even when restricting ourselves to *only visual features* we still obtain practically applicable results (prec=0.82 for recall=0.6, and prec=0.89 for recall=0.4). This is useful for finding specifically strong candidates of private photos in large photo sets. In the context of photo sharing applications this might allow for tuning the sensitivity of personal "privacy warning" systems according to individual preferences.

Although privacy is a subjective concept, our data annotation captures an aggregated community perception of privacy, which turns out to be correlated to textual and visual features in a plausible way, and can be predicted.

### 5.5.2 Ranking and Diversification

#### Query Set

In order to obtain queries for ranking and diversification experiments, we filtered a set of image-related queries from an MSN search engine query log[5] based on their target URLs. To this end, we considered only queries containing the term "image" or "photo" in these URLs.

We manually filtered out queries related to adult content, and randomly selected 50 queries for our experiments (examples being "picture of a power plant", or "fall pictures"). For each of these, we retrieved photos and corresponding metadata from the top-1000 Flickr search results.

[5] http://research.microsoft.com/en-us/um/people/nickcr/wscd09/

To assess the quality of privacy-based ranking and diversification in a real-world scenario, we performed a user study with 30 participants. We presented the keyword queries and their top-10 results retrieved by different ranking methods to the users in random order. Users were asked to judge the privacy level for each image on a 4-point Likert scale, with 1 corresponding to "clearly public" and 4 to "clearly private". Apart from the modified Likert scale the user assessment was a replication of the game described in Section 5.2.

### Ranking by Privacy

For each query, we computed privacy-oriented rankings as described in Section 5.4.2. The list of test photos in descending order of their user-assigned privacy value was considered as ground truth for our experiments. We compared the order of the automatically generated rankings using Kendall's Tau-b [Kru58]. We chose the Tau-b version in order to avoid a systematic advantage of our methods due to many ties produced by the high number of photos with equal user ratings.

The main observations are:

- The original Flickr ranking does not consider the privacy of the images in the search results. This was reflected by a small $\tau_b$ value of -0.04. In contrast, our methods show a clear correlation with the user-based privacy ranking.
- Since not all of the photos in the set contained usable metadata, ranking based on text with ($\tau_b$= 0.26) is outperformed by ranking using visual features ($\tau_b$=0.31).
- Finally the combination of textual and visual features provides the best ranking performance ($\tau_b$=0.33). However, ranking with only visual features can still be useful for cases and applications where no or insufficient textual photo annotation is available.

The results of the pairwise t-test between the different ranking types confirm their statistical significance for a confidence level of 95%. We obtained good classification results with default parameters for SVM. Although the tuning options such as soft-margin parameter C or alternative kernels might lead to further improvements, this is outside of the focus of this work.

### Diversification

We used $\alpha$-nDCG-G in order to measure the quality of privacy-based image diversification described in Section 5.4.3. In our experiments we used an $\alpha$ value of 0.5 to balance the amount of re-ranking of the original search result with respect to the privacy dimension. The (hypothetical) optimal ranking for normalizing DCG was obtained through diversifying images directly by their user defined privacy scores.

The main result observations shown in Figure 5.10 are:

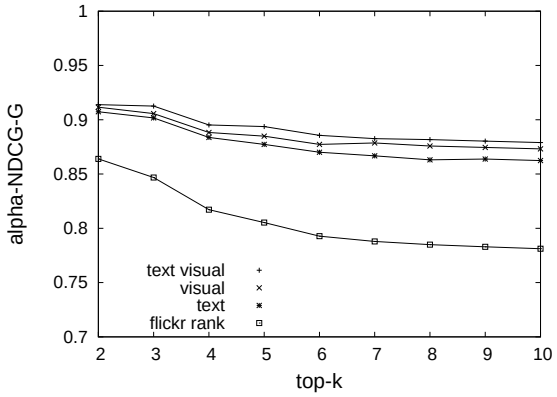



**Figure 5.10** $alpha$-nDCG-G for the initial Flickr ranking and top-10 diversified search results using textual and visual image features for classification

- Our diversification approaches clearly help to diversify Flickr query results according to the privacy dimension, and, compared to the original Flickr rank, achieve an improvement of 8.7% using textual and 9.8% using visual features.
- The combination of visual and textual features leads to a further average improvement (10.5%).

For each ranking position $k \geq 2$ we conducted pairwise t-tests between the different ranking methods for a confidence level of 95%. The tests between the original rank and other methods confirm the statistical result significance. The difference between the textual features and the combination of the textual and visual features is significant for $k \geq 5$.

## 5.6 Discussion

In this chapter we addressed Problem 3 to automatically determine adequate *privacy settings* for photos newly shared at social applications and applied classification using various visual and textual features to estimate the privacy degree of images.

Classification models were trained on a large-scale dataset with privacy assignments obtained through a social annotation game. We made use of classifier outputs to compute ranked lists of private images as well as search results diversified according to the privacy dimension. Our classification and ranking experiments show the best performance for a hybrid combination of textual and visual information. However, the approach of using only visual features shows applicable results as well and can be applied in scenarios where no textual annotation is available (e.g. personal photo collections or mobile phone pictures). One additional result from our privacy game was that although there exist a common notion of privacy in the community such as higher privacy for family photos, some aspects still vary. For example, people who own pets see a picture of a dog as private whereas other do not. Thus an interesting future research direction is personalization of our techniques to better reflect privacy preferences of individual users and user groups.

Additionally, in future work we aim to study other enhanced features in the privacy context such as color-SIFT, or applying and testing various alternative machine learning approaches for classification and ranking. Furthermore, due the complexity of the task, it makes sense to consider gathering even larger training sets for performance boosting. For gathering additional training data, techniques like active learning algorithms, which take intermediate user feedback into account, might help to further optimize the choice of sample images presented to human assessors. As the perception of "privacy" is highly subjective and user-dependent, we would like to study recommender mechanisms, such as collaborative filtering, to account for individual user preferences, and to provide personalized results. Most of the mobile devices nowadays are equipped with high definition cameras. Photos taken with smartphones or tablets can differ from the pictures available by Flickr. We plan to experiment with a dataset of pictures taken with mobile phones where we expect a larger proportion of private images. We further plan to include new features like context recognition which is possible due to recently released powerful operation systems for smartphones.

# 6 Discussion and Future Work

## 6.1 Summary of Contributions

Advances in networking infrastructure and increased popularity of Web 2.0 platforms enabled support in sharing of large amount of data for individual users as well as social and professional communities. However, this development calls for mechanisms enabling efficient and at the same time privacy preserving data exchange. In this thesis we addressed three important challenges in this context, namely: (1) Efficient privacy-preserving indexing and retrieval of shared documents; (2) Scalable content diversity analysis for large document corpora suitable for supporting efficient access control to an inverted index; and (3) Privacy-oriented content analysis of shared content to automatically distinguish public from potentially sensitive content.

### 6.1.1 Privacy Preserving Content Indexing and Retrieval

First, we addressed the problem of efficient and privacy-preserving sharing of distributed documents within working groups in an enterprise or the Web by introducing *ZERBER+R* - an indexing facility, which can efficiently locate and select *top-k* relevant documents the user is allowed to access. *ZERBER+R* provides probabilistic guarantees for document confidentiality and enables efficient document management and index maintenance. This *r*-confidential index does not require users to trust a central authority and is tunable with an *r* parameter to trade between efficiency and confidentiality protection, bounding the amount of information that can be leaked from the index taking into account the background knowledge of the potential adversary. To this extent, the index relies on a centralized set of *largely untrusted* index servers and offers resistance against inappropriate information disclosure even if *k*-1 index servers are compromised (where $k$ is a parameter). To provide tunable resistance to statistical attacks, *ZERBER* employs a novel term merging scheme that has minimal impact on the index lookup costs. Our experiments have shown that *ZERBER* makes economical use of network bandwidth, requires minimal key management, and answers

queries almost as fast as an ordinary inverted index. Whereas *ZERBER+R* supports efficient retrieval and ranking of search results, an interesting direction for future work in this area is related to an incremental enhancement of the index by integrating advanced retrieval features, e.g. proximity search or phrase search, while maintaining confidentiality guarantees for the indexed data.

### 6.1.2 Scalable Content Diversity Analysis

Secondly we proposed two novel, efficient algorithms for content analysis of large document corporas to estimate their topical diversity with probabilistic guarantees. This diversity measure can be employed to study structural properties of large data structures such as inverted index and is potentially useful to improve access control properties by partitioning the index. Whilst state-of-the-art diversity measurement algorithms require quadratic computational time with respect to the dataset size, our first method, called TrackDJ, can be executed in a linear time and the second method, called SampleDJ, even in constant time. We evaluated our methods on real world datasets such as scientific publication directory DBLP with over one million paper titles and metadata from 144 million Flickr images as well as on synthetic datasets with different diversity properties and were able to practically prove our theoretical findings and revealed interesting insights into temporal development of communities, periodicities, and trends on Flickr.

### 6.1.3 Privacy-Oriented Content Analysis

Finally, we applied image classification using various visual and textual features to estimate the privacy degree of photos. Our Support Vector Machines - based classification models were trained on a large-scale dataset with privacy assignments obtained through a social annotation game. Furthermore, we provided privacy-oriented search and privacy-based search result diversification capabilities. In this context, we made use of the classifier outputs to compute ranked lists of private images as well as search results diversified according to the privacy dimension. Moreover, we analysed the usefulness of textual and visual information in the context of privacy-oriented image classification. Our classification and ranking experiments have shown the best performance for a hybrid combination of textual and visual information. Nevertheless, the classification approach using only visual features has shown promising results as well. This approach can be applied in scenarios where no textual annotation is available (e.g. personal photo collections or mobile phone pictures). Overall, our results have confirmed that classification with Support Vector Machines, in particular using the combination of textual and visual information, is a suitable and effective approach to automatically determine sensitivity degree of images to support users in selecting adequate privacy settings. We have also observed that although there exist a common notion of privacy in the community, some aspects vary. Thus an interesting future research direction is personalization of our techniques to better reflect specific privacy preferences of individual users and user groups.

## 6.2 Open Research Directions

Security and privacy in modern information systems gained increased importance not only in enterprises or social web communities but also in cloud computing, where the functionality of data management and sharing is not possible without privacy protection guarantees. The results presented in this thesis can be directly applied in these scenarios and open various research directions.

A promising direction for further research is privacy-preserving multi-keyword ranked search over encrypted cloud data. It would be interesting to examine how far structured and semi-structured data such as graphs or relational data can be handled, especially employing various access control policies and possible trust negotiations in dynamic sharing environments. For structured data, query rewriting and result filtering mechanisms need to be adjusted to bind the possible leakage of information. For semi-structured data, adaptation of the state-of-the-art information retrieval approaches like synonym queries, proximity search and query suggestions to privacy-preserving indexing proposed in Chapter 3 are further interesting research directions.

Additionally, the RDJ index proposed in Chapter 4 can be used for computing efficient heuristics for determining the number of clusters in large datasets, which is required as a parameter in many clustering algorithms such as k-means [Jai10], k-medoids or the expectation-maximization algorithm. Clustering of an inverted index according to the access control policies can provide a tunable trade-off between the retrieval efficiency and the storage space usage. On the one hand, cluster-per-community indexes can be highly efficient but space consuming. On the other hand, a single index, though saving storage space, is limited in its retrieval efficiency due to a necessary scan of a posting list checking access right for every user query and each posting element. Moreover, the techniques for automatic classification of documents according to their sensitivity proposed in this thesis can enable further fine tuning of the system such that more sensitive documents receive better protection.

Finally, as the perception of "sensitivity" can be highly subjective and user-dependent, in the future we would suggest to employ personalization techniques and study recommender mechanisms, such as content-based or collaborative filtering, to account for individual user preferences, and to enable personalized settings with respect to the sensitivity estimation and privacy protection. Different object types such as video, audio or micro-blog entries will require adjustments or further development of methods for sensitivity estimation investigated in Chapter 5 so far. Especially appealing is to study distributed systems, where the data of different privacy degree is shared among different servers (including cluster and P2P scenarios). Furthermore, interdisciplinary multicultural studies would potentially reveal additional insights into privacy perception of different communities. The concept of "privacy" can be further changing over time, opening interesting directions for temporal analysis on such perceptions for individuals as well as for user groups.

# A

# Curriculum Vitae

Sergej Zerr, born on 27 April 1977, in Karaganda, USSR.

**Studies**

| | |
|---|---|
| **12/2006 - 12/2015** | Ph.D. studies.<br>Gottfried Wilhelm Leibniz Universität Hannover, Germany.<br>Ph.D. thesis: "Privacy Preserving Content Analysis, Indexing and Retrieval for Social Search Applications." |
| **10/2003 - 03/2006** | M.Sc. in Information Engineering.<br>University of Osnabrück, Germany and University of Twente, The Netherlands.<br>Master thesis: "Resource discovery in distributed environment - Authorised search in restricted working areas". |
| **03/2000 - 06/2003** | Computer Science Engineer (Dipl.-Inf. (FH)).<br>University of Applied Sciences, Osnabrück, Germany.<br>Diploma thesis: "Analysis and integration of a memory-based database system in data-intensive functions of the client server system HYDRA to reduce reply times". |

**Professional Experience**

| | |
|---|---|
| **01/2008 - 12/2015** | Researcher and project manager. L3S Research Center, Gottfried Wilhelm Leibniz Universität Hannover, Germany. |
| **Project Management** | "QualiMaster", "Gute Arbeit", "IT Ecosystems", "IRIS" (Forschungsinitiative Sicherheit), "LearnWeb". |
| **Research & Development** | EU Project "TenCompetence" |
| **Teaching**: | Web Science Lecture (teaching assistant). Invited lectures in Advanced Methods of IR. Supervision of numerous bachelor and master theses, interns and student projects. |

| | |
|---|---|
| **04/2006-12/2007** | Software developer Conweso GmbH OsnabrückGermany Development of Web-based software solutions. |
| **01/2004 - 12/2005** | Student/research assistant in the CASHMERE-int project. University of Osnabrück, Germany. The CASHMERE-int project: standards development in the context of the Semantic Web. |
| **09/2002 - 08/2003** | Internship in software development. MPDV Mikrolab GmbH, Mosbach, Germany. Development of an OOP - framework (C++/Delphi). Integration of a memory based DBMS (SQLite) into the company framework (HYDRA). Development of software for production data acquisition (C++, Delphi, SQLite). |
| **07/2001 - 08/2002** | Student assistant. University of Applied Sciences, Osnabrück, Germany. Development of software for mathematical analysis and electrotechnical engineering. |

**Research Project Management**

| | |
|---|---|
| **2014-2015** | EU project Qualimaster (www.qualimaster.eu) FP7, Project Number:619525 |
| **2015** | BMBF project eLabour (www.sofi-goettingen.de/index.php?id=1218) |
| **2012-2015** | BMBF project Re-SozIT (www.sofi-goettingen.de/index.php?id=1086) |
| **2010-2012** | NTH project IT-Ecosystems (www.it-ecosystems.org) |
| **2008-2010** | LUH project Forschungsinitiative Sicherheit (IRIS) |
| **2008-2009** | EU project TENCompetence (www.tencompetence.org) IST-2004-02787 (WP management) |

**PC Memberships & Reviews**

| | |
|---|---|
| **2015** | IEEE Transactions on Dependable and Secure Computing (TDSC), Reviewer |
| **2015** | Springer, Datenbank-Spektrum (DASP), Reviewer |
| **2014** | International Journal of Technology Enhanced Learning (IJTEL), Reviewer |
| **2014** | The 40th International Conference on Very Large Databases (VLDB 2014), Sub-Reviewer |

| | |
|---|---|
| **2014** | The 36th European Conference on Information Retrieval (ECIR 2014), Sub-Reviewer |
| **2013** | The 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013), PC Member |
| **2010** | ACM Transactions on Information and System Security Journal (ISSJ), Reviewer |
| **2009** | Workshop on Trust, Risk, Reputation and Recommendation on the Web (TR4Web 2009), PC Member |
| **2008** | The 7th International World Wide Web Conference (WWW 2008), Sub-Reviewer |

**Received Grants and Awards**

| | |
|---|---|
| **08/2012** | ACM SIGIR 2012 Student conference participation grant The 35th International ACM SIGIR Conference on Research and Development in Information Retrieval |
| **01/2011** | Ministry for Science and Culture of Lower Saxony Multimedialer Hochschullehrpres für Niedersachsen, Campusmerge 2011. "LearnWeb2.0 for CLIL: Promising ways to teach content through a foreign language" |
| **09/2007** | German Academic Exchange Service (DAAD) Course participation grant EDBT Summer School on Database Technologies for Novel Applications |

**Selected Postgraduate Courses**

| | |
|---|---|
| **01/2011** | Leadership skills Gottfried Wilhelm Leibniz Universität Hannover Germany |
| **06/2010** | Conversation techniques in teaching Gottfried Wilhelm Leibniz Universität Hannover Germany |
| **06/2010** | Design of lectures Gottfried Wilhelm Leibniz Universität Hannover Germany |
| **03/2010** | Plan and design teaching activities Gottfried Wilhelm Leibniz Universität Hannover Germany |
| **09/2009** | Information Retrieval ESSIR 2009 - European Summer School in Information Retrieval PaduaItaly |
| **09/2007** | Databases EDBT Summer School on Database Technologies for Novel Applications Bolzano Italy |

# Bibliography


[ACN+09] Rabeeh Abbasi, Sergey Chernov, Wolfgang Nejdl, Raluca Paiu, and Steffen Staab. Exploiting Flickr tags and groups for finding landmark photos. In *Advances in Information Retrieval*, volume 5478 of *LNCS*. Springer, 2009.

[AEG+07] Shane Ahern, Dean Eckles, Nathaniel S. Good, Simon King, Mor Naaman, and Rahul Nair. Over-exposed?: Privacy patterns and considerations in online and mobile photo sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 357–366, New York, NY, USA, 2007. ACM.

[AGHI09] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 5–14, New York, NY, USA, 2009. ACM.

[AGS09] Ching Man Au Yeung, Nicholas Gibbins, and Nigel Shadbolt. Providing access control to online photo albums based on tags and linked data. In *Social Semantic Web, AAAI Spring Symposium 2009*. AAAI Press, 2009.

[AHM+09] Fabian Abel, Eelco Herder, Ivana Marenzi, Wolfgang Nejdl, and Sergej Zerr. Evaluating the benefits of social annotation for collaborative search. In *Proceedings of the 2nd Annual Workshop on Search in Social Media, co-located with ACM SIGIR 2009 Conference on Information Retrieval, Boston, USA*, SSM'09, July 2009.

[AMNZ09a] Fabian Abel, Ivana Marenzi, Wolfgang Nejdl, and Sergej Zerr. LearnWeb2.0: Resource sharing in social media. In *Proceedings of the Workshop on Social Information Retrieval for Technology-Enhanced Learning at the*

*International Conference on Web-based Learning (ICWL'09)*, volume 535 of *SIRTEL'09*. CEUR-WS.org, August 2009.

[AMNZ09b] Fabian Abel, Ivana Marenzi, Wolfgang Nejdl, and Sergej Zerr. Sharing distributed resources in LearnWeb2.0. In *Proceedings of the 4th European Conference on Technology Enhanced Learning*, EC-TEL'09, pages 154–159. Springer, 2009.

[AS72] D.L. Alspach and H.W. Sorenson. Nonlinear Bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, Aug 1972.

[BA05] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE'05, pages 217–228, Washington, DC, USA, 2005. IEEE Computer Society.

[Bar06] Susan B. Barnes. A privacy paradox: Social networking in the united states. *First Monday*, 11(9), Sept. 2006.

[BBAV09] Mayank Bawa, Roberto J. Bayardo, Jr, Rakesh Agrawal, and Jaideep Vaidya. Privacy-preserving indexing of documents on the network. *The VLDB Journal*, 18:837–856, August 2009.

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM symposium on Principles of database systems*, PODS '02, pages 1–16, New York, USA, 2002. ACM.

[BC05] Stefan Büttcher and Charles L. A. Clarke. A security model for full-text file system search in multi-user environments. In *Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, volume 4, pages 13–13, Berkeley, CA, USA, 2005. USENIX Association.

[BCF01] Elisa Bertino, Silvana Castano, and Elena Ferrari. Securing xml documents with author-x. *IEEE Internet Computing*, 5:21–31, May 2001.

[BCFM00] Andrei Zary Broder, Moses Charikar, Alan Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60:630–659, June 2000.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Proceedings of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, 2004.

[BG05] Kaouthar Blibech and Alban Gabillon. Chronos: an authenticated dictionary based on skip lists for timestamping systems. In *Proceedings of the 2005 Workshop on Secure Web Services*, SWS'05, pages 84–90, New York, NY, USA, 2005. ACM.

[BKZ15] Volker Baethge-Kinsky and Sergej Zerr. Die Erschließung von Primärmaterial qualitativer Studien für die Sekundäranalyse als Herausforderung für Sozialwissenschaften und Informatik. *Datenbank-Spektrum*, 15(1):33–39, 2015.

[Bla93] Matt Blaze. A cryptographic file system for unix. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS'93, pages 9–16, New York, NY, USA, 1993. ACM.

[BMT+05] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Minerva: collaborative p2p search. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB'05, pages 1263–1266. VLDB Endowment, 2005.

[BNST05] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE'05, pages 174–185, Washington, DC, USA, 2005. IEEE Computer Society.

[Bro97] Andrei Zary Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society.

[Bro00a] Andrei Zary Broder. Identifying and filtering near-duplicate documents. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, COM '00, pages 1–10, London, UK, 2000. Springer-Verlag.

[Bro00b] Andrei Zary Broder. Min-wise independent permutations: Theory and practice. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, ICALP '00, pages 808–, London, UK, 2000. Springer-Verlag.

[Bur98] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.

[cCM05] Yan cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Proceedings of the 3rd Applied*

*Cryptography and Network Security Conference*, ACNS'05, pages 442–455. Springer, 2005.

[CF08] Barbara Carminati and Elena Ferrari. Privacy-aware collaborative access control in web-based social networks. In *Proceeedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security*, pages 81–96, Berlin, Heidelberg, 2008. Springer-Verlag.

[CG98] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA, 1998. ACM.

[Cha02a] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 1st edition, 2002.

[Cha02b] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 380–388, New York, USA, 2002. ACM.

[CK97] P. Crescenzi and V. Kann. Approximation on the web: A compendium of NP optimization problems. In José Rolim, editor, *Randomization and Approximation Techniques in Computer Science*, volume 1269 of *Lecture Notes in Computer Science*, pages 111–118. Springer Berlin Heidelberg, 1997.

[CK06] Harr Chen and David R. Karger. Less is more: Probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 429–436. ACM, 2006.

[CKC+08] 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 *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 659–666. ACM, 2008.

[CLK04] Taenam Cho, Sang-Ho Lee, and Won Kim. A group key recovery mechanism based on logical key hierarchy. *Journal of Computer Security*, 12:711–736, September 2004.

[COZ+08] Juri Luca De Coi, Daniel Olmedilla, Sergej Zerr, Piero A. Bonatti, and Luigi Sauro. A trust management package for policy-driven protection & personalization of web content. In *Proceedings of the 9th IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY'08, pages 228–230. IEEE Computer Society, 2008.

[DC00] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'00, pages 256–263, New York, NY, USA, 2000. ACM.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, SCG '04, pages 253–262, New York, USA, 2004. ACM.

[DKLR00] Paul Dagum, Richard Karp, Michael Luby, and Sheldon Ross. An optimal algorithm for Monte Carlo estimation. *SIAM Journal on Computing*, 29:1484–1496, March 2000.

[DSZ12] Fan Deng, Stefan Siersdorfer, and Sergej Zerr. Efficient Jaccard-based diversity analysis of large document collections. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM'12, pages 1402–1411, New York, NY, USA, 2012. ACM.

[FBP+09] Flavio Figueiredo, Fabiano Belém, Henrique Pinto, Jussara Almeida, Marcos Gonçalves, David Fernandes, Edleno Moura, and Marco Cristo. Evidence of quality of textual features on the web 2.0. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 909–918, New York, NY, USA, 2009. ACM.

[FE08] A. Felt and D. Evans. Privacy protection for social networking platforms. In *Proceedings of the Workshop on Web 2.0 Security and Privacy 2008 in conjunction with 2008 IEEE Symposium on Security and Privacy*, Oakland, CA, 2008.

[Fea03] James D. Fearon. Ethnic and cultural diversity by country. *Journal of Economic Growth*, 8:195–222, 2003.

[FWY05] Benjamin C. M. Fung, Ke Wang, and Philip S. Yu. Top-down specialization for information and privacy preservation. In *Proceedings of the 21st IEEE International Conference on Data Engineering*, ICDE'05, pages 205–216, Washington, DC, USA, 2005. IEEE Computer Society.

[GA05] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES'05, pages 71–80, New York, NY, USA, 2005. ACM.

[GPK+13] Mihai Georgescu, Dang Duc Pham, Nattiya Kanhabua, Sergej Zerr, Stefan Siersdorfer, and Wolfgang Nejdl. Temporal summarization of event-related updates in wikipedia. In *Proceedings of the 22nd International World Wide*

*Web Conference, Companion Volume*, WWW'13, pages 281–284. International World Wide Web Conferences Steering Committee, 2013.

[GS09] Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th International Conference on World Wide Web*, WWW'09, pages 381–390, New York, USA, 2009. ACM.

[GTS01] M.T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proceedings of the DARPA Information Survivability Conference & Exposition II*, volume 2 of *DISCEX'01*, pages 68–82. IEEE, 2001.

[Gwe10] Kilem Gwet. *Handbook of Inter-Rater Reliability*. Advanced Analytics, LLC, second edition, 2010.

[HILM02] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD'02, pages 216–227, New York, NY, USA, 2002. ACM.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO'95, pages 339–352, London, UK, 1995. Springer-Verlag.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 604–613, New York, USA, 1998. ACM.

[Iye02] Vijay S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'02, pages 279–288, New York, NY, USA, 2002. ACM.

[Jai10] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[Joa99] Thorsten Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[KCN06] Christian Kohlschütter, Paul Chirita, and Wolfgang Nejdl. Using link analysis to identify aspects in faceted web search. In *Proceedings of the Faceted Search Workshop co-located with ACM SIGIR'2006*, Seattle, WA, 2006.

[KPK10] Wonjun Kim, Jimin Park, and Changick Kim. A novel method for efficient indoorï£¡outdoor image classification. *Journal of Signal Processing Systems*, 61:251–258, 2010.

[Kre89] Charles C. Krebs. *Ecological Methodology*. HarperCollins, 1989.

[KRS+03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 29–42, Berkeley, CA, USA, 2003. USENIX Association.

[Kru58] W. H. Kruskal. Ordinal measures of association. *Journal of the American Statistical Association*, 53(284):814–861, 1958.

[Lan08] Patricia G. Lange. Publicly private and privately public: Social networking on youtube. *Journal of Computer-Mediated Communication*, 13(1):361–380, November 2008.

[LDR06] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd IEEE International Conference on Data Engineering*, ICDE'06, pages 25–25, Washington, DC, USA, 2006. IEEE Computer Society.

[Ley] Michael Ley. The DBLP computer science bibliography. http://www.informatik.uni-trier.de/~ley/db/.

[Lie69] Stanley Lieberson. Measuring population diversity. *American Sociological Review*, 34(6):850–862, 1969.

[LM03] Christian. Lévêque and Jean-Claude Mounolou. *Biodiversity*. John Wiley & Sons, 2003.

[Low04] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, January 2004.

[LYRL04] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, December 2004.

[MDN+08] Ivana Marenzi, Elena Demidova, Wolfgang Nejdl, Daniel Olmedilla, and Sergej Zerr. Social software for lifelong competence development: Challenges and infrastructure. *International Journal of Emerging Technologies in Learning (iJET)*, 3(S1):18–23, 2008.

[MHW06] Soumyadeb Mitra, Windsor W. Hsu, and Marianne Winslett. Trustworthy keyword search for regulatory-compliant records retention. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB'06, pages 1001–1012. VLDB Endowment, 2006.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1, March 2007.

[MKNZ10] Ivana Marenzi, Rita Kupetz, Wolfgang Nejdl, and Sergej Zerr. Supporting active learning in CLIL through collaborative search. In *Proceedings of 9th International Conference on Web-based Learning*, ICWL'10. Springer, 2010.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, 2008.

[MS03] Gerome Miklau and Dan Suciu. Controlling access to published data using cryptography. In *Proceedings of the 29th International Conference on Very Large Data Bases*, volume 29 of *VLDB'03*, pages 898–909. VLDB Endowment, 2003.

[MSN11] Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. Incremental diversification for very large sets: a streaming-based approach. In *Proceedings of the 34nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR'11, Beijing, China, 2011. ACM.

[MTH02] Christopher Meek, Bo Thiesson, and David Heckerman. The learning-curve sampling method applied to model-based clustering. *Journal of Machine Learning Research (JMLR)*, 2:397–418, March 2002.

[MZ12] Ivana Marenzi and Sergej Zerr. Multiliteracies and active learning in CLIL - the development of LearnWeb2.0. *IEEE Transactions on Learning Technologies (TLT)*, 5(4):336–348, January 2012.

[MZAN09] Ivana Marenzi, Sergej Zerr, Fabian Abel, and Wolfgang Nejdl. Social sharing in LearnWeb2.0. *International Journal of Continuing Engineering Education and Life-Long Learning (IJCEELL)*, 19(4/5/6):276–290, 2009.

[MZN08] Ivana Marenzi, Sergej Zerr, and Wolfgang Nejdl. Providing social sharing functionalities in LearnWeb2.0. In *Proceedings of the TENCompetence Workshop: Stimulating Personal Development and Knowledge Sharing*, pages 30–31, Sofia, Bulgaria, 2008.

[Olk93] Olken. Random sampling from databases. In *Ph.D. Diss. (University of California at Berkeley)*, 1993.

[PCI+07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'07, pages 1–8. IEEE, June 2007.

[Pla99] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[PSF10] Odysseas Papapetrou, Wolf Siberski, and Norbert Fuhr. Text clustering for peer-to-peer networks with probabilistic guarantees. In *Advances in Information Retrieval*, volume 5993 of *LNCS*, pages 293–305. Springer Berlin / Heidelberg, 2010.

[RBS10] Davood Rafiei, Krishna Bharat, and Anand Shukla. Diversifying web search results. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 781–790, New York, USA, 2010. ACM.

[RCZS14] Markus Rokicki, Sergiu Chelaru, Sergej Zerr, and Stefan Siersdorfer. Competitive game designs for improving the cost effectiveness of crowdsourcing. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM'14, pages 1469–1478, New York, NY, USA, 2014. ACM.

[Ric01] John A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 3 edition, April 2001.

[RM10] Ismael Rafols and Martin Meyer. Diversity and network coherence as indicators of interdisciplinarity: case studies in bionanoscience. *Scientometrics*, 82(2):263–287, 2010.

[RMG98] Y. Raja, S. J. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG '98, pages 228–, Washington, DC, USA, 1998. IEEE Computer Society.

[RZS15] Markus Rokicki, Sergej Zerr, and Stefan Siersdorfer. Groupsourcing: Team competition designs for improving the cost effectiveness of crowdsourcing. In *Proceedings of the 24 International World Wide Web Conference*, WWW'15, 2015.

[Sch10] Peter Schaar. Privacy by design. *Identity in the Information Society*, 3(2):267–274, 2010.

[SCK+06] Nachiketa Sahoo, Jamie Callan, Ramayya Krishnan, George Duncan, and Rema Padman. Incremental hierarchical clustering of text documents. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 357–366, New York, NY, USA, 2006. ACM.

[SH10] Verbraucherzentrale Schleswig-Holstein. Statistische Erfassung zum Internetverhalten Jugendlicher und Heranwachsender, March 2010. Available at: http://www.vzsh.de/mediabig/109191A.pdf.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.

[Sim49] E. H. Simpson. Measurement of diversity. *Nature*, 163, 1949.

[Sin01] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001.

[Sob74] I. M. Sobol. *The Monte Carlo Method*. The University of Chicago Press, Chicago, IL, 1974.

[SPS09] Jose San Pedro and Stefan Siersdorfer. Ranking and classifying attractiveness of photos in folksonomies. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 771–780, New York, NY, USA, 2009. ACM.

[SSP09] Anna Cinzia Squicciarini, Mohamed Shehab, and Federica Paci. Collective privacy management in social networks. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 521–530, New York, NY, USA, 2009. ACM.

[Sti07] Andy Stirling. A general framework for analysing diversity in science, technology and society. *Journal of The Royal Society Interface*, 4(15):707–719, 2007.

[Stu] Stud IP Learning Management System (LMS). Available at: http://www.studip.de/.

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 44–55, Washington, DC, USA, 2000. IEEE Computer Society.

[THD+06] Mischa Tuffield, Stephen Harris, David P. Dupplaw, Ajay Chakravarthy, Christopher Brewster, Nicholas Gibbins, Kieron O'Hara, Fabio Ciravegna, Derek Sleeman, Yorick Wilks, and Nigel R. Shadbolt. Image annotation

with photocopain. In *Proceedings of WWW'06 Workshop on Semantic Web Annotations for Multimedia*, SWAMM'06, Edinburgh, Scotland, 2006.

[TZB+13] Nam Khanh Tran, Sergej Zerr, Kerstin Bischoff, Claudia Niederée, and Ralf Krestel. Topic cropping: Leveraging latent topics for the analysis of small corpora. In *Proceedings of the International Conference on Theory and Practice of Digital Libraries, TPDL 2013*, volume 8092 of *Lecture Notes in Computer Science*, pages 297–308. Springer Berlin Heidelberg, 2013.

[vAD04] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM.

[VJ02] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2002.

[VJZ98] Aditya Vailaya, Anil Jain, and Hong Jiang Zhang. On image classification: City images vs. landscapes. *Pattern Recognition*, 31(12):1921 – 1935, 1998.

[vLGOvZ09] Reinier H. van Leuken, Lluis Garcia, Ximena Olivares, and Roelof van Zwol. Visual diversification of image search results. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 341–350, New York, NY, USA, 2009. ACM.

[VSCY09] Nitya Vyas, Anna Cinzia Squicciarini, Chih-Cheng Chang, and Danfeng Yao. Towards automatic privacy management in Web 2.0 with semantic analysis on annotations. In *Proceedings of the 5th International Conference on Collaborative Computing*, CollaborateCom'09. IEEE, 2009.

[VSS+08] Erik Vee, Utkarsh Srivastava, Jayavel Shanmugasundaram, Prashant Bhat, and Sihem Amer Yahia. Efficient computation of diverse query results. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 228–236, Washington, DC, USA, 2008. IEEE Computer Society.

[WZ09] Jun Wang and Jianhan Zhu. Portfolio theory of information retrieval. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 115–122, New York, NY, USA, 2009. ACM.

[YHBO10] Che-Hua Yeh, Yuan-Chen Ho, Brian A. Barsky, and Ming Ouhyoung. Personalized photograph ranking and selection system. In *Proceedings of the international conference on Multimedia*, MM '10, pages 211–220, New York, NY, USA, 2010. ACM.

[ZBC11] Sergej Zerr, Kerstin Bischoff, and Sergey Chernov. GuideMe! the world of sights in your pocket. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE'11, pages 1348–1351, Washington, DC, USA, 2011. IEEE Computer Society.

[ZDC10] Sergej Zerr, Elena Demidova, and Sergey Chernov. deskWeb2.0: Combining desktop and social search. In *Proceedings of the Desktop Search Workshop, In conjunction with the 33rd Annual International ACM SIGIR 2010 Conference*, Geneva, Switzerland, 2010.

[ZdM+14] Sergej Zerr, Mathieu d'Aquin, Ivana Marenzi, Davide Taibi, Alessandro Adamou, and Stefan Dietze. Towards analytics and collaborative exploration of social and linked media for technology-enchanced learning scenarios. In *Proceedings of the 1st International Workshop on Dataset PROFIling & fEderated Search for Linked Data co-located with the 11th Extended Semantic Web Conference*, volume 1151 of *PROFILES'14*. CEUR-WS.org, 2014.

[ZDO+08] Sergej Zerr, Elena Demidova, Daniel Olmedilla, Wolfgang Nejdl, Marianne Winslett, and Soumyadeb Mitra. Zerber: r-confidential indexing for distributed documents. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT'08, pages 287–298, New York, NY, USA, 2008. ACM.

[ZMKL05] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 22–32, New York, USA, 2005. ACM.

[ZN08] Sergej Zerr and Wolfgang Nejdl. Privacy preserving document indexing infrastructure for a distributed environment. *Proceedings of the VLDB Endowment (PVLDB)*, 1(2):1638–1643, 2008.

[ZOC+09] Sergej Zerr, Daniel Olmedilla, Juri L. De Coi, Wolfgang Nejdl, Piero A. Bonatti, and Luigi Sauro. Policy based protection and personalized generation of web content. In *Proceedings of the 2009 Latin American Web Congress*, LA-WEB'09, pages 112–119, Washington, DC, USA, 2009. IEEE Computer Society.

[ZONS09] Sergej Zerr, Daniel Olmedilla, Wolfgang Nejdl, and Wolf Siberski. Zerber+R: top-k retrieval from a confidential index. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT'09, pages 439–449, New York, NY, USA, 2009. ACM.

[ZPD14] Sergej Zerr, Odysseas Papapetrou, and Elena Demidova. HEALTH+Z: confidential provider selection in collaborative healthcare P2P networks. In

*Proceeding of the 1st International Workshop on Privacy-Preserving IR co-located with 37th Annual International ACM SIGIR conference*, volume 1225 of *PIR'14*, pages 1–6. CEUR-WS.org, 2014.

[ZSH12] Sergej Zerr, Stefan Siersdorfer, and Jonathon Hare. PicAlert!: a system for privacy-aware image classification and retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM'12, pages 2710–2712, New York, NY, USA, 2012. ACM.

[ZSHD12] Sergej Zerr, Stefan Siersdorfer, Jonathon Hare, and Elena Demidova. Privacy-aware image classification and search. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 35–44, New York, NY, USA, 2012. ACM.

[ZSP+14] Sergej Zerr, Stefan Siersdorfer, José San Pedro, Jonathon S. Hare, and Xiaofei Zhu. NicePic!: a system for extracting attractive photos from Flickr streams. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'14, pages 1259–1260, New York, NY, USA, 2014. ACM.