Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

A Graph-based Model for Multimodal Information Retrieval

PhD THESIS

submitted in partial fulfillment of the requirements for the degree of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Serwah Sabetghadam
Registration Number 1128896

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dr. Andreas Rauber
Second advisor: Dr. Mihai Lupu

External reviewers:
Univ.Prof. Dr. Fabio Crestani. University of Lugano, Italy.
a.Univ.Prof. Dr. Josef Küng. Johannes Kepler University of Linz, Austria.

Vienna, 17th January, 2017

Serwah Sabetghadam | Andreas Rauber

Declaration of Authorship

Serwah Sabetghadam
Favoritenstrasse 9-11, 1040, Wien

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 17th January, 2017

Serwah Sabetghadam

To My Loved Ones

تقدیم به عزیزانم

بخصوص

به وجود سرشته به عشق گلگل و دل پر از مهر بابا

Acknowledgements

When I look back over the last few years, I see that this PhD journey could not be accomplished without the presence and support of precious people in my life.

First and foremost, I want to express my gratitude to my advisor, Andreas Rauber, who agreed to work with me. I am so grateful for his confidence in my ability to develop new ideas, for mentoring me finding my way through research challenges with his inspiring guidance and encouragement, and for his support at critical moments. I am also very thankful to my co-advisor, Mihai Lupu, for his constant presence and patience in discussions, for his guidance and suggestions, and for his availability at any time.

A special appreciation goes to my doctoral thesis committee, Fabio Crestani and Josef Küng, who accepted to review my thesis. I am also thankful to all anonymous reviewers of my papers who provided valuable feedback to improve my work. I want to express my thanks to the PhD school committee for making it possible to pursue my studies at TU Wien. Notably, Clarissa Schmid for her constant kindness and help in study-related problems, from the very beginning until these last days of defense. Also I am thankful to Katharina Engel and Mamen for their help in different issues that happened in this way.

I am also thankful to Martha Larson for providing me an opportunity to join their team for a research visit at Delft University and Radboud University in the Netherlands. Although short, it was a valuable experience for me to collaborate with a new group in another university.

Further, I want to thank all my colleagues at the IR group for fruitful discussions and collaborations. In particular, my office mates, João and Navid, and later Florina and Aziz, for their feedbacks and for tolerating me in difficult moments. In particular, João for his continuous positive energy, good mood and smiling face during hard days, Aziz and Aldo for helpful discussions any time needed, and Linda for the fun during long working days and weekends at university. I also thank Ralf for his informative discussions and reviews, particularly reading some chapters of my thesis.

Special thanks goes to the friends who accompanied me in this way. Notably, Rostik for being always and always ready for any help, and for all the memorable times we had. Peter Kan for encouraging me in difficult times and for sharing his enthusiasm and inspiration in research, and Soheil for his empathy in tough days. I am specially thankful to my close friend Mitra from the IBS group for sharing the ups and downs of this path

together, for the encourages we would get at the end of each appointment to tackle the problems again and again, and for the memorable afternoons and all the Hafiz poems we read together.

I am particularly thankful for the very close friendship of Tara and Mansooreh, who were like family members for me in Vienna. Tara became my very first and one of the best friends and roommates I had in Vienna. Her positive energy and attitude were a great help for me especially at the beginning of this journey. I am thankful for her company and creating an atmosphere of "feeling at home". I am equally thankful to Mansooreh for her open heart, presence and help in different challenges I faced in my academic and non-academic life, and for her very helpful negotiations and encourages.

I want also to thank all those people who made these years much easier and fun, in particular Mona, Peyman, and Amir for all memorable nights we created together. I am also thankful to all PhD school colleagues for the parties and get-togethers we had. I miss those times these days.

When it comes to my family, I can only express my deep gratitude for their faith and trust in me to pursue my life path freely. I express my special thanks to my mom and dad, Hajyeh and Abbas, for their constant support and encouragement to continue and never give up, and for their unconditional love. I sincerely thank my siblings for their unfailing support in thick and thin, which gave me peaceful mind to go ahead. I am thankful to my sister, Somayeh, who was like part of me in sorrow and joy, just kilometres away, and for her inner peace that would calm me down in unbalanced days. To my brother, Arash, the one I could always rely on in everything, and for his open arms whenever I needed help. I love you all and thank you being pillars in my life.

I dedicate this work to my loved ones, especially, to my mom and dad who nourished all moments of my life with love and love and love.

Abstract

Finding useful information from large multimodal document collections such as the WWW is one of the major challenges of Information Retrieval (IR). Nowadays the proliferation of available information sources—text, images, audio, video and more—increase the need for multimodal search. Multimodal information retrieval is about the search for information of any modality on the web, with unimodal or multimodal queries. For instance, a unimodal query may contain only keywords, whereas multimodal queries may be a combination of keywords, images, video clips or music files.

Users have learnt to explain their information need through keywords and expect the result as a combination of different modalities. Search engines like Google and Yahoo often show related videos or images in addition to the text result to the user. Usually, in a keyword based search, only the metadata information of a video or an image (e.g. tag, caption or description) is used to find relevant results. This approach is limited to textual information only and does not include information from other modalities. There is few options such as Google image search, which considers the image features to perform the image search task based on. In case the user query is an image, or a combination of a video file and keywords, the question arises how can a search engine benefit from different modalities in the query to retrieve multimodal results.

Usually, search engines build upon text search by using non-visual information associated with visual content. This approach in multimodal search does not always result in satisfying results, as it completely ignores the information from other modalities in ranking. To address the problem of visual search approaches, multimodal search reranking has received increasing attention in recent years.

In addition to the observation that data consumption today is highly multimodal, it is also clear that data is now heavily semantically interlinked. This can be through social networks (text, images, videos of users on LinkedIn, Facebook, or the like), or through the nature of the data itself (e.g. patent documents connected by their metadata - inventors, companies, semantic connections via linked data). Structured data is naturally represented by a graph, where nodes denote entities and directed/indirected edges represent the relations between them. Such graphs are heterogeneous, describing different types of objects and links. Connected data poses a challenge is traditional IR method which is based on independent documents. The question arises whether structured IR can be an option for retrieving more relevant data objects.

In this thesis, we propose a graph-based model for multimodal information retrieval. We consider different relation types between information objects from different modalities. A framework is devised to leverage individual features of different modalities as well as their combination through our formulation of faceted search. We denote an inherent feature, metadata or property of an information object as its facet. We highlight the role of different facets of the user's query in visiting different parts of the graph. We employ a potential recall analysis on a test collection and further highlight the role of multiple facets, relations between the objects, and semantic links in recall improvement. Furthermore, we perform an analysis on score distribution on the graph at large number of steps to investigate the role of different facets and link types in the final performance of the system. The experiments are conducted on ImageCLEF 2011 Wikipedia collection, as a multimodal benchmark dataset containing approximately 400,000 documents and images.

Contents

Acknowledgements **ix**

Abstract **xiii**

Contents **xv**

1 Introduction **1**
1.1 Motivation 1
1.2 Challenges 2
1.3 Research Questions 4
1.4 Contributions 5
1.5 Thesis Structure 6
1.6 Publications 7

2 Context **9**
2.1 Background 9
2.2 Related Work 22

3 Astera Model **35**
3.1 Model Definition 35
3.2 Evaluation Design 40

4 Reachability Analysis **47**
4.1 Relevant Objects Distribution 47
4.2 Reachability from Different Facets 48
4.3 Reachability through Different Links 51
4.4 Recall Analysis of Different Topic Categories 57
4.5 Graph Visit from Different Facets 63
4.6 Summary 67

5 Precision Analysis **69**
5.1 Precision Analysis with Different Facets 69
5.2 Precision Analysis by Adding Semantic and Similarity Links 74
5.3 Score Analysis in the Graph - $\alpha, \beta, \delta, \gamma$ 79

5.4 Summary 89

6 Conclusion **91**
6.1 Summary 91
6.2 Research Questions Revisited 92
6.3 Future Work 94

A Computational Complexity Analysis **97**
A.1 Matlab Matrix Sizes and Memory Requirement 98
A.2 Conclusion 101

B Astera Software Architecture Document (SAD) **103**
B.1 Introduction 103
B.2 Architectural Representation 104
B.3 Logical View 104
B.4 Process View 106
B.5 Implementation View 108
B.6 Deployment View 110
B.7 Data View 110
B.8 Configuration 110
B.9 Similarity Links 115
B.10 Semantic Links 116
B.11 Implementation Strategies 117

List of Figures **125**

List of Tables **126**

Bibliography **129**

CHAPTER 1

Introduction

1.1 Motivation

There is a rapid growth of online multimodal content as well as personal data generation in our daily life. This trend creates the need for seamless information retrieval from different modalities such as video, audio, image, and text. Multimodal information retrieval is about the search for information of any modality on the web, with unimodal or multimodal queries. For instance a unimodal query may contain only keywords or only image examples, whereas multimodal queries may be a combination of images, video clips or music files. Web search engines over the past decade have evolved into being the primary gateways to accessing the ever-growing amount of data available online. Multimodal IR presents a great challenge since it deals with several data types and modalities, each having its own underlying retrieval model.

In order to successfully manage large multimedia resources, search engines should provide results to users in different modalities. However, users rather keep their conventional method of search by entering some keywords and receiving a multimodal result. Using different modalities —text, image, audio, or video—to improve an IR System is challenging since each modality has a different concept of similarity underneath. There has been a long stream of research in this area, ranging from associating image with text search scores, to sophisticated fusion of multiple modalities [FFFPZ05, MDS05, MS07, DLTX11, KKW^{+}14].

Popular search engines like Google, Yahoo and Bing build upon text search by using non-visual information associated with visual content. This approach in multimodal search does not always produce satisfying results, as it completely ignores the information from other modalities in ranking. To address the problem of visual search approaches, multimodal search reranking has received increasing attention in recent years.

Reranking leverages different sources and methods to create a new ranked list with relevant results in higher ranks. Mei et al. [MRLT14] performed a survey on reranking models of multimodal retrieval. In their view, work on multimodal reranking is divided into four categories: 1) *Self-reranking*: methods that include data from the original ranking result such as Pseudo-Relevance Feedback, where the top-ranked results are regarded as "positive" in learning a ranking model. 2) *Example-based reranking*: methods to understand the query using accompanying examples. One instance of such query example is to upload an image to find similar ones in addition to textual query. 3) *Crowd reranking*: leverages crowd-sourced knowledge on the web to mine relevant patterns for a query. 4) *Interactive reranking*: a user edits a part of the search results, e.g. deletes or emphasizes a result.

We refer to related work that is principally based on similarity between independent documents as *non-structured IR* (i.e. category 2). Going beyond the document itself, in modern IR settings, documents are usually not isolated objects. Instead, they are frequently connected to other objects, via hyperlinks or meta-data [MCN06]. They provide mutual information on each other—forming a background information model that may be used explicitly. Sometimes this information link is explicit as related information (e.g. a Wikipedia document and its images), resulting in a network of related objects. Sometimes it is inherent in the information object, e.g. the relation between the subtitles and frames of a video file. Moreover, user-generated multimodal content, domain-specific multimodal collections, or platforms like the semantic web impose *structured IR* based on links between different information objects. Graph-based methods for reranking are a subset of self-reranking methods (category 1), in which a graph oriented search is performed based on relations between objects. Since 2005 there is a trend towards a combination of these search methods, leveraging both structured and non-structured IR [KSI$^+$08, DTCT10, EB11]. This offers new opportunities for multimodal retrieval, utilizing link information for reranking.

1.2 Challenges

Mostly, research in the area of structured search creates a graph of one type of relations e.g., similarity links with one modality (images/videos) [YMN10, HKC07, WLT$^+$12]. Others use only semantic relations between information objects based on various semantic databases [RSA04, TdM14, EB11]. For instance, Rocha et al. [RSA04] model the knowledge-base of a sample website by adding semantic links between various entities of the website.

Considering only one type of relation limits us to perceive information only from one perspective. Suppose that we have a Wikipedia collection, containing pages with images inside them. Assume that the search task is to perform a text-based image search. The collection is indexed for each language separately and the search engine finds the Wikipage containing the image and shows the image to the user. If the user can search only in English, she is limited to the result from the English collection, as some images may have only German caption or metadata. However, if we use a semantic link that links the same pages in two languages, we can reach in one step to German Wikipages and also their images, leading to retrieve a much richer result set.

Another example is a semantic relation between two images in a collection (e.g. two flower types from the same height in mountains), which are not similar to each other. Including both similarity and semantic connections in the retrieval process helps reaching a more diverse set of relevant results. How to exploit various relations to achieve a better performance is one challenge in graph-based multimodal IR.

Another challenge is the multimodality of information. In pure text retrieval, both an information need and the results are text-based. One aspect is information from one modality that is contained in another one. For example, a Wikipedia page may contain an image. This image can be indexed, processed and retrieved independently if we extract the image from the Wikipage.

Yet another aspect is multimodality of an information object itself. A website has textual features such as TF.IDF (Term Frequency.Inversion of Document Frequency), as well as visual facets as page layout or style. This is true for images as well. They can have visual features as SIFT (Scale-Invariant Feature Transform) [BETVG08] or CEDD (Color Edge Histogram) [CB08], as well as textual features based on their sub-titles, OCR'-text from logos. Hence, we can observe that any information object is inherently multimodal. How to take advantage of these included information is another challenge in multimodal IR.

There is an additional aspect of multimodal retrieval when we meet the challenge of variety of different features in each modality. In Information Retrieval, we search an information object based on one or more features of that modality. We compute the similarity between the same features of a query and each information object. For instance, for images, we can find similar images based on different features like edge histogram or color histogram of that image. We call these features of a document or an image, facets of that modality. In a text search engine, we index the available documents based on their containing terms/words and their frequencies. When a user searches based on some keywords, we find similar documents. This similarity is computed e.g. based on Cosine similarity of the query word vector and document word vector. However, a multimodal collection contains different modalities each having different features.

A multimodal query can be as simple as keywords (uni-modal) used in daily search for finding text, video or audio. It can be as complex as a combination of an image, an audio file, or a video fragment. For instance, a user loves nature and has seen a new flower. In this case, the features may be the color histogram or edge histogram of the image of the flower. However, if she uses a picture of this flower and the keyword “species”, then the information from both the query example and the keywords are needed. This brings up the challenge of dealing with a rich number of features in retrieving relevant information objects for users. However, there are still challenges of how to combine the result of features of different modalities. Some related work use early fusion to consider different features in the retrieval. In early fusion, a long feature vector is created from concatenating multimodal features. The search is conducted on this vector. Early fusion suffers from the curse of dimensionality problem, i.e. with high dimensional features, the distance between the information objects become very similar [AHK01]. For late fusion, the result of individual modalities are combined with different weighting strategies. One problem with late fusion is choosing the optimal weighting strategy for different modalities. Further, the fusion may not be good due to poor result of individual modalities. This brings the challenge of how to leverage features from different modalities in multimodal IR, meanwhile avoiding the shortcomings of early and late fusion.

1.3 Research Questions

As we have seen so far, on one side we have implicit and explicit relations between information objects. On the other side, we have heterogeneous modalities and their features or facets. There are numerous challenges in this context. The following are the research questions and a number of associated sub-questions that we attempt to answer in this thesis:

- **RQ1:** Can we define a graph-based model for multimodal multi-faceted information retrieval?
    - What are the nodes in such a model?
    - What kind of links can we leverage between different information objects?
    - How can retrieval be modelled in such a graph?
    - How are the result of different features utilized in the retrieval process?
- **RQ2:** In such a graph model, can the relevant nodes be reached?
    - How much can we improve reachability of relevant objects by leveraging multi-facet/poly-representation of query and information objects?
    - What is the effect of different facet combinations on recall behaviour of different query categories by their difficulty rating in benchmark evaluations?
    - What is the effect of adding semantic links between information objects?
    - Do different facets with the same recall value visit the same parts of the graph or the same relevant information objects?
- **RQ3**: In such a model can scores identify the relevant nodes?
    - What is the effect of facet combinations on precision?
    - Is the initial score of the nodes affecting the final score distribution in the graph? What is the effect of Metropolis-Hastings traversal?
    - Do the scores in the graph converge (i.e. does the graph stabilize in stationary distribution state)? and if so after how many steps?
    - What are the factors explaining the convergence state?

1.4 Contributions

To address these challenges, we propose a model and prototype, named Astera[1], to leverage hybrid search (structured and non-structured search) in order to handle the diverse nature of the nodes and edges in the multimodal content domain [SLR13, SLBR14]. We can model domain specific collections with the help of different relation types, and enrich the available data by extracting characteristic information from data objects, in the form of facets. By facet we mean any inherent feature or property of an information object. This definition allows considering a document under several points of view, each one being associated to a specific space. For instance, text documents have primarily a textual facet, but also others such as stylistic/layout facets (covered partially by image features), may contain images, or have time/versioning aspects (recency of information). Another example is image files which primarily have image facets (comprising several actual feature spaces such as color, edge or texture) but also other facets such as written information (as detected with OCR), metadata such as time, owner, etc.

This model decomposes the multimodal query in different representations/facets. For instance, we assume that a user has a query such as "red or black mini cooper". She can specify her request by determining which aspects of this Mini Cooper have attracted her, such as color and brand name here. We can take advantage of color histogram of images in the collection in addition to finding mini cooper cars to provide the results for the user.

In our model, we extract different facets of the query example. We extract different facets of all information objects in the collection too. Based on the query facets, we use top standard search results to activate a subset of points in the graph of information objects. From the activated points we continue the search in two dimensions: non-structured search (through similarity links) and structured search (through semantic links). After a defined number of steps in the graph, each information object receives scores from the combination of all its neighbours propagated from starting point sores. We show that our model matches the efficiency of non-graph based indexes, while having the potential to exploit different facets for better retrieval [SBR14].

Another distinguishing point of our work is the enriched connections between information objects. When the user searches for a movie such as "Pride and Prejudice", she does not necessarily know that this movie is in the category of movies in 1940-1950 (facet), it is one of the top products of director "Robert Z. Leonard" (semantic), or it is based on the Book "Pride and Prejudice" (semantic) of which there are many other books from the same author (semantic), that there are movies about them as well (semantic), or that there is another movie (semantic) from another author (semantic) but with highly similar characteristics (facet). These are instances of information that are modelled via different relation types and representations in our graph. These links can be supplied by the collection provider, or from external database, such as Linked Open Data for the semantic links.

---

[1] Astera is open-source and available at http://www.ifs.tuwien.ac.at/∼sabetghadam/Astera.html

The contributions of this work can therefore be summarized as below:

- A generic model for multimodal collections. In this model, information objects may be from any modality. This results in various types of relations between them.
- We formulate a form of faceted search. We define a facet as an inherent feature of an object. With respect to this definition, we decompose an information object to its facets. For example we could extract visual and textual facets of an image, or visual, textual and acoustic facets of an audio file. We can calculate similarity and relevancy based on common facets of these objects and the query. We investigate the effect of leveraging different facets in system performance. We show that based on the idea of poly-representation [LIK06], we reach higher precision and recall, when we use diverse facets.
- We present a highly configurable framework for multimodal information retrieval. Different traversal methods, facets and weighting strategies can be defined to configure this framework.
- We define the concept of reachability of relevant information objects. We define reachability as the possibility to reach to relevant information objects in the graph starting from some query. We investigate the role of different facets and links in improving recall. We demonstrate that our graph model of the collection outperforms standard indexed search results.
- We develop an approach to compare the query-dependent and query-independent Random Walks in the graph. We show the graph behaviour in the stationary distribution state with each of these methods.
- We show the applicability of our model on a multimodal domain by using the ImageCLEF 2011 Wikipedia collection dataset [TPK11].

1.5 Thesis Structure

Chapter 2 of this thesis is an introductory chapter to familiarize the reader with the field of multimodal Information Retrieval. The proposed model and evaluation methodology are presented in Chapter 3. We address RQ1 in this chapter. Chapters 4 and 5, each address the set of research questions introduced earlier. Finally we present concluding remarks in Chapter 6. What follows is a more detailed summary of each chapter's content:

- **Chapter 2** (**Context**) The first part of this chapter provides fundamental information about the technologies and concepts used in the remainder of this thesis. We particularly focus on the research domain of Information Retrieval and multimodality, faceted search, graph-based search are described. Further, we investigate two well-known graph search methods: Random Walks and Spreading Activation, and select the appropriate method for our model. We compare both in two cases of query-dependent and query-independent routing. In the second part of this chapter, we present the related work that influenced and motivated this dissertation.

- **Chapter 3** (**Astera Model**) describes the fundamental contribution of the thesis—the Astera framework. Different characteristics of this framework are described in this chapter. We investigate our first research question (RQ1) in this chapter, and introduce the evaluation methodology and the experimental setup that forms the basis of our empirical evaluations. The test collections, the set of test queries, and evaluation metrics are then introduced.
- **Chapter 4** (**Reachability Analysis**) This chapter provides answers to sub-questions in RQ2. It focuses on the most important aspect of reachability, and discusses how solutions can leverage poly-representation in order to optimize the cost of visiting more nodes to the benefit of higher recall.
- **Chapter 5** (**Precision Analysis**) discusses particularly how scores are distributed over the graph and how these affect precision. This chapter provides answers to sub-questions in RQ3.
- **Chapter 6** (**Conclusions**) summarizes and concludes the thesis and presents future research directions that emerged from this work.

1.6 Publications

This thesis is based on several published works.

Chapter 2 which shows a theoretical comparison of the two traversal methods is based on the paper:

- Sabetghadam S., Lupu M., and Rauber A., Which one do you choose? Spreading Activation or Random Walks?. In *Proceedings of Information Retrieval Facility Conference (IRFC)*, 2014

Chapter 3 which presents the model is based on these papers:

- Sabetghadam S., Lupu M., Rauber A., Astera - A generic model for multi-modal Information Retrieval, In *Proceedings of the Workshop on Integrating IR Technologies for Professional Search*, held in ECIR, 2013
- Sabetghadam S., Lupu M., Bierig R., and Rauber A., A combined approach of structured and non-structured IR in multi-modal domain, In *Proceedings of International Conference on Multimedia Retrieval (ICMR)*, 2014

- Sabetghadam S., Doctoral Symposium in International Conference on Multimedia Retrieval, In *Proceedings of International Conference on Multimedia Retrieval (ICMR)*, 2014

Chapter 4 which investigates the role of facets and links in reachability is based on these papers:

- Sabetghadam S., Lupu M., Bierig R., and Rauber A., Reachability Analysis of Graph Modelled Collections, In *Proceedings of 37th European Conference on Information Retrieval (ECIR)*, 2015
- Sabetghadam S., Lupu M., Bierig R., and Rauber A., A Faceted Approach to Reachability Analysis of Graph Modelled Collections. Under review.

Chapter 5 which elaborates the score distribution in the graph is based on the paper:

- Sabetghadam S., Bierig R., and Rauber A., A Hybrid Approach for Multi-Faceted IR in multi-modal Domain, In *Proceedings of Image Retrieval Conference and Labs of the Evaluation Forum (CLEF)*, 2014
- Sabetghadam S., Lupu M., and Rauber A,. Leveraging Metropolis-Hastings Algorithm on Graph-based Model for multimodal IR. In *Proceedings of First International Workshop on Graph Search and Beyond (GSB)*, held at SIGIR 2015
- Sabetghadam S., Lupu M., Rauber A., Random Walks Analysis on Graph Modelled Multimodal Collections, In *Proceedings of Second International KEYSTONE Conference*, 2016

CHAPTER 2

Context

In the first part of this chapter we introduce a few background concepts in the context of multimodal Information Retrieval. Afterwards, a survey of the related research work is presented in the second part. Primarily the focus is on the current research trends that are related to the important parts of this thesis.

2.1 Background

This section introduces some basic concepts that are crucial for understanding the remainder of the thesis. First of all, we define the concept of Information Retrieval and multimodal Information Retrieval in Section 2.1.1, which are the target environment for every presented contribution of this thesis. The notion of multimodality, discussed in this section, is one of the essential characteristics of Information Retrieval [HM09, WLT^{+}12] and plays a vital role and main motivation for the contributions presented in this work. Following, Section 2.1.2 presents the probability concepts used in this thesis. Section 2.1.3 describes two well-known methods of graph traversal in Information Retrieval: Spreading Activation and Random Walks. In this section, we compare these methods from two aspects of query-dependent and query-independent routing. Based on the findings we choose which one to use in our work. Finally, Section 2.1.4 presents the Metropolis-Hastings algorithm which is based on Random Walks, and is used in the experiments in Chapter 5.

2.1.1 Information Retrieval

The meaning of Information Retrieval (IR) is very broad. Very simple looking for information such as reading a number from a card can be defined as Information Retrieval. However, academically it is defined as finding material (usually documents) of an unstructured nature. The result of this finding is to satisfy an information need from large collections [SRM09].

Previously, Information Retrieval was mainly to find documents based on keywords. In recent decades, the digital content of web has increased dramatically. High rate of production of images, audio and video contents has created the web as a multimedia platform. Multimodal Information Retrieval is defined as the search for information of any modality on the web with an information need that can be unimodal or multimodal. A unimodal query may contain only keywords or an image example, whereas multimodal queries may be a combination of images, video clips, or music files. For instance, we can search a video clip with a query of text and image example.

To measure the performance of an IR system, we usually use two well-known measures: Precision and Recall. Precision is a measure of result relevancy. It is the ratio of the number of relevant documents found to the top N documents. For example P@20 shows the percentage of relevant documents in the top 20 results. Recall is the fraction of found relevant results for a specific query. It is the ratio of the number of relevant documents to the number of all relevant ones for that query.

Here, we define some IR concepts used in this thesis:

- **Information Object**: We call any kind of individual data object an information object.
- **Modality**: We call Text, Image, Audio and Video different modalities.
- **Facet**: We define a facet as an inherent property of an information object. It can be a feature like color histogram of an image, or metadata information of an object such as genre of a music file, or tags of an image.
- **Unimodal Information Object**: An information object which contains only one modlity, e.g., a text document or an image.
- **Multimodal Information Object**: An information object which contains more than one modality e.g. a video clip with subtitle, a document containing an image.

2.1.2 Probability Concepts

In this section we define probability concepts used in this thesis, particularly used in Random Walks (Section 2.1.3), and Metropolis-Hastings algorithm (Section 2.1.4) as traversal methods.

The Probability Ranking Principle

Assume $R_{d,q}$ be a random variable that says whether d is relevant to a query q. It takes a value of 1 if the document is relevant and 0 otherwise. In context, we write just R for $R_{d,q}$. Based on a probabilistic model, the order to present documents to the user is to rank documents by their estimated relevance probability to the query: $P(R = 1|d, q)$.

Robertson defines the probability ranking principle as follows [Rob97]: "If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data." Probabilistic models treat the process of document retrieval as a probabilistic inference. Similarities are computed as probabilities that a document is relevant for a given query. Based on Bayes theorem we have: $p(d|q) = \frac{p(q|d)p(d)}{p(q)}$ where $p(q|d)$ is the probability of occurrence of query q if we have document d. The Bayes Optimal Decision rule returns documents which are more likely relevant than nonrelevant [SRM09]:

$$d \text{ is relevant iff} P(R = 1|d, q) > P(R = 0|d, q) \tag{2.1}$$

Markov Chain

Markov chain is a random process that transits randomly from one state to another. Let X_t denote the value of a random variable at time t. Markov chain is usually characterized by the memorylessness property—the transition probabilities in the state space depend only on the current state of the random variable.

$$Pr(X_{t+1} = s_j|X_0 = s_k, \ldots X_t = s_i) = Pr(X_{t+1} = s_j|X_t = s_i) \tag{2.2}$$

Thus for a Markov random variable, the only knowledge needed to predict the next state is the current state of the random variable. Markov chain is referred to a sequence of random variables $(X_0, \ldots, X_n)$ generated through a Markov process. A chain is defined through its transition probabilities, which are the probabilities of going to state s_y from state s_x in one step,

$$W(x, y) = Pr(X_{t+1} = s_y|X_t = s_x) \tag{2.3}$$

The chain is started with a starting vector $\pi(0)$. Usually all elements are 0 except one element with value 1 which is the particular starting point. In further steps, the probability values spread over the possible state space. We have the probability of transition from any random state to all other states. Thus we define a probability transition matrix W where the probability of moving from state x to state y is $W(x, y)$. We write:

$$\pi(t + 1) = \pi(t)W \tag{2.4}$$

Using the matrix form, we see the equation as

$$\pi(t) = \pi(t - 1)W = (\pi(t - 2)W)W = \pi(t - 2)W^2 \tag{2.5}$$

Continuing in this fashion we obtain:

$$\pi(t) = \pi(0)W^t \tag{2.6}$$

A Markov chain may reach a stationary distribution state π^*, where the vector of probabilities in this state is independent from the initiating state values. The stationary distribution satisfies

$$\pi^* = \pi^* W \tag{2.7}$$

A Markov chain is given by a set of states $X = x_1, ..., x_n$ and a transition matrix W, such that $W(x, y)$ is the probability of jumping from node x to y. The property of Markov chain model is that for any probability distribution π over the state space X, and for any $\epsilon \in R$, a transition matrix W can be built such that $\exists t : ||W^t(x, X_i) - \pi(X)_i)|| < \epsilon, \forall x \in X, \forall i \in 1, n$. This means that there is an iteration t that the probability of going from node x to any of the other nodes $(W^t(x, X_i))$ does not change the distribution of the state $\pi(X)_i$. After a sufficiently large number of steps t, we can be in any state with a fixed probability.

We mention some properties of a Markov chain used in this thesis:

- Irreducible chain: a chain is irreducible if for any state, the probability of getting there given any starting point is more than zero. In this case, there is a positive number of steps that one can go from one state to another.
- Aperiodic chain: in an aperiodic chain, there is no rhythm in which states can be reached given a starting point. The number of steps to move between each two states is not multiple of some integer.
- Ergodic Chain: A Markov chain is ergodic if it is both irreducible and aperiodic.
- Stationary Distribution: After a sufficient number of steps, an ergodic chain converges to fixed probabilities for each of the states in the chain, which is called stationary distribution. This probability distribution is independent of initial probability in the chain.
- Largest eigenvalue is 1: Suppose $Wx = \lambda x$ for some $\lambda > 1$. Since the rows of W are non-negative and sum to 1, each element of vector Wx is a convex combination of the components of x, which can be no greater than x_{max}, the largest component of x. On the other hand, at least one element of λx is greater than x_{max}, which proves that $\lambda > 1$ is impossible.

Monte Carlo

Monte Carlo is originally developed by physicists to use repeated random sampling to compute a complex integral. It needs a probability distribution to generate inputs randomly. Usually the Monte Carlo method requires a large number of inputs to reach an accurate approximation of the solution. This method is utilized to draw samples from a hard distribution based on constructing a Markov chain, so called *Monte Carlo Markov Chain (MCMC)*. After a large number of sampling they can integrate the complex function [MU49, Has70].

2.1.3 Traversal Methods

One of the main challenges in graph modelled data is how to "intelligently" traverse the graph and exploit the associations between the data objects. Two highly used methods in retrieving information on structured data are: Markov Chain Random Walks and Spreading Activation. Crestani [Cre97] explains Spreading Activation as a method of associative retrieval to identify relevant information based on the associations between the information elements. Random walks is defined as a sequence of independent, distributed discrete random path selection in a graph of objects - also the basic method for PageRank [Ber05]. Using Random Walks to find related data objects has been investigated for example in [CTC05, Ber05, SJ01].

In this section, we investigate these two approaches from a theoretical point of view. We categorize the routing in a graph, as *query dependent* and *query independent*. We compare Spreading Activation and Random Walks according to these two categories and select the appropriate method for our model.

Spreading Activation The idea of application of Spreading Activation in IR originates from the works on Associative Retrieval, in which an explicit information objects are represented in the form of nodes and associations by links connecting these nodes.

This information is either already retrieved and is static, like the relation between the objects of information and indexings, or is dynamically achieved like based on user behaviour in the search session [Cre97].

Spreading activation has various utilizations, for instance, Salton and Buckley [SB88] leverage spreading activation for identifying related terms and documents to improve the retrieval process. Rocha et al. [RSA04] propose a model utilizing spreading activation for search in Semantic Web. Hussein and Ziegler use spreading activation in determining important elements in an ontology according to user's current context and past interactions [HZ08].

Random Walks Random Walk is a chain of states created by some stochastic process. A simple example is a random surfer moving randomly from one place to another. The Random Walk method considered in this thesis is Markov chain. With Markov property, the conditional probability of being in next state is dependent only on the current state, not any of the other past states. PageRank is one of the most prominent examples leveraging Random Walks. It ranks websites based on the authority weights given to each page of a web graph. This authority weight is independent of textual content and is based on the web hyperlink structure. PageRank is based on a random surfer model that can be viewed as a stationary distribution of a Markov chain [Ber05]. Another application of Random Walks is in Craswell and Szummer work, who model queries and clicked images in a graph [CS07]. They use Random Walks to retrieve more relevant images for each textual query. Furthermore, Clements et al. [CDVR10a] use Random Walks through a social annotation graph, comprising of people, tags, and content. They identify the influence of the design choices in social annotation systems on ranking tasks. Random walks has also been used in query expansion modelling. Collins et al. [CTC05] identify whether a potential expansion term reflects aspects of the original query in the stationary distribution of their model. They create a Markov chain framework based on different knowledge sources for term associations. This model in the stationary distribution is used to obtain probability approximates of different expansions of the query to use in query expansion.

Query Independent Routing

Random walks' most famous instance is PageRank, which is query independent, while for Spreading Activation Berthold et al. [BBK$^+$09] has shown that "*Pure Spreading Activation is pointless*". What is the difference?

Spreading Activation Spreading Activation is inspired by simulated neural networks without any training phase. Edge weights are based on the semantics of the modelled domain. The Spreading Activation procedure always starts with an initial set of activated nodes. Different values can be given to the initial nodes according to the task being solved. They are usually the result of a first stage processing of the query, e.g. a distance measure between the information objects and the query. During propagation, other nodes are activated and ultimately, a set of nodes and their respective activation values are obtained. Here we explain how to compute the activation values of the nodes after some iterations in the graph, independently of the query.

We denote the initial activation of the nodes as $a^{(0)}$ and the activation in t-th iteration as $a^{(t)}$. Three phases are commonly defined: preprocessing, spreading and postprocessing [BBK$^+$09]. The preprocessing phase consists of calculating an input value in_v for each node v by aggregating output values of its neighbours:

$$in_v^{(t)} = \sum_{u \in V} o_u^{(t-1)} \cdot W_{u,v} \tag{2.8}$$

where o_u is the output value of node u and V is the set of the nodes. Based on the input value, different functions are used to determine the activation value [Cre97], such as linear, sigmoid, threshold, or step function. We denote any of these functions as act. Based on it, we calculate the activation value of each node:

$$a_v^{(t)} = act(in_v^{(t)}) \tag{2.9}$$

Finally, an output function out determines how much output of a node is passed to its neighbours. We define it as:

$$o_v^{(t)} = out(a_v^{(t)}) \tag{2.10}$$

This function can be defined in a way to avoid retentioning of activation from previous iterations. This way it helps control the overall activation in the graph [Cre97]. Putting all these equations together, we obtain the following general formula to calculate the activation for the next step:

$$a_v^{(t+1)} = act\left(\sum_{u \in V} out(a_u^{(t)}) \cdot W_{uv}\right) \tag{2.11}$$

Weighting in Spreading Activation There is no specific constraint in Spreading Activation on weight definition or weight values on the edges. It is generally application dependent. For example, Rocha et al. [RSA04] define the edge weight based on the ratio of common semantic neighbours of nodes u and v to all semantic neighbours of node u. Hussein and Ziegler [HZ08] define the weighting based on the context defined by an expert in the preliminary step of system definition.

Memory Spreading Activation algorithm In this variation of spreading activation, we propose an input function on received energy to manage the amount of energy spreading in the network. The amount of energy a node receives in each step t is the sum of the energy of its neighbours. Part of this received energy has been sent two steps before from the same node to its neighbours. We subtract this part from the whole received energy to prevent energy bias near activated nodes. We define the *energy capacity* of nodes as a vector sm, which contains the sum of the edge weights for each node. We define the energy capacity of node i as $sm_i = \sum_{j=1}^{n} W_{ij}$ where j goes over the columns for row i. This is the energy it is able to carry to its neighbours. It may be less or more than the energy it has at any point in time, as a function of the weights of its neighbours. We assume m as $diag(sm)$ which converts vector sm to the diagonal matrix with the vector values on the diagonal. Here, we define the energy received in step t as follows:

$$m = W \cdot 1 \cdot I_r \tag{2.12}$$

where W is $r \times r$ matrix, 1 is $r \times 1$ vector, and I_r is identity matrix of size r .

$$\begin{aligned}
a^{(1)} &= a^{(0)} \cdot W \\
a^{(2)} &= a^{(0)} \cdot W^2 - a^{(0)} \cdot m \\
a^{(3)} &= a^{(0)} \cdot W^3 - a^{(1)} \cdot m \\
a^{(t-2)} &= a.W^{t-2} - a^{(t-4)} \cdot m \\
a^{(t-1)} &= a.W^{t-1} - a^{(t-3)} \cdot m \\
a^{(t)} &= a^{(0)} \cdot W^t - a^{(t-2)} \cdot m \\
&= a^{(0)} \cdot W^t - a^{(0)} \cdot W^{t-2} \cdot m + a^{(t-4)} \cdot m^2 \\
&= a^{(0)} \cdot W^t - a^{(0)} \cdot W^{t-2} \cdot m + a^{(0)} \cdot W^{t-4} \cdot m^2 - a^{(t-6)} \cdot m^3 \\
&= a^{(0)} \sum_{k=0}^{t-1} (-1)^k \cdot W^{(t-2k)} \cdot m^k
\end{aligned} \tag{2.13}$$

where $a^{(0)}$ is the activation vector and $a^{(t)}$ is the energy distribution in step t.

Substituting Equation 2.12, we have:

$$\begin{aligned} a^{(t)} &= a^{(0)} \sum_{k=0}^{t-1} (-1)^k \cdot W^{t-2k} \cdot (W \cdot 1 \cdot I_r)^k \\ &= a^{(0)} \sum_{k=0}^{t-1} (-1)^k \cdot W^{t-2k+k} \cdot 1^k \cdot I_r^k \\ &= a^{(0)} \sum_{k=0}^{t-1} (-1)^k \cdot W^{t-k} \cdot 1^k \cdot I_r \end{aligned} \tag{2.14}$$

In each step, we decrease the self-energy received by subtracting the multiplication of activation value of two steps before to the energy capacity of this node $(a_{t-2} \cdot m)$.

Random Walks Different variants of Random Walks exist, but the Markov Chain Random Walks is by far the most commonly used in the IR literature, and we focus on it here. The transition probabilities between the states in a Markov chain form a matrix. In our case, they are represented by the W matrix of transition weights. By $W_{u,v}$ we now understand $P(v|u)$, the probability of moving from node u to node v. The matrix W is row-stochastic, i.e. the probabilities on one row sum up to 1. The objective of Random Walks is to reach a probability distribution over the set of nodes V. If we view this as a vector $p \in \mathbb{R}^n$, we can denote by $p^{(0)}$ and $p^{(t)}$ the initial probability distribution, and respectively the probability distribution over the set of nodes at time t, which is computed by:

$$p^{(t)} = p^{(t-1)}W = p^{(0)}W^t \tag{2.15}$$

Weighting in Random Walks Weighting the edges for Random Walks presents as much flexibility as in Spreading Activation, with the one constraint that the matrix must be row-stochastic. For example, Craswell and Zsummer [CS07] define the weight based on the normalized value of the number of clicks between two nodes: $P_{t+1|t}(k|j) = C_{jk} / \sum_i C_{ji}$. They define the nodes in the graph based on queries and documents. The edge between a document and a query indicates a click for that query-document pair. The weight on the edge is proportional to the total number of clicks from all the users. Also they utilize the factor of *self-transitivity* (st). This parameter helps significantly to control the walk pace. It can be interpreted as the importance of staying in the same node.

Comparing Spreading Activation and Random Walks in Query-independent Routing To compare these two methods, in using Spreading Activation the graph should be in the form of a stochastic matrix. Each row shows the probability of moving from a node to its neighbours summed to 1. We can have still customized weightings on the edges, but it should satisfy the stochastic property of the matrix. Random Walks based matrices are stochastic inherently, as they are created based on a stochastic process.

Based on Equation 2.10, the output of a node in Spreading Activation is the result of applying the activation and output functions on the input of the node. If the input function is defined as the linear function and the output and activation functions are the identity function, then Equation 2.11 in Spreading Activation can be written as

$$a_v^{(t+1)} = \sum_{u \in V} a_u^{(t)} \cdot W_{uv} \tag{2.16}$$

which in compact form is:

$$a^{(t+1)} = a^{(0)} \cdot W^{t+1} \tag{2.17}$$

Comparing Equations 2.15 and 2.17 we observe that in a query independent case, both Random Walks and Spreading Activation are identical. In this case, the convergence of the weighting matrix is important since there is no limit to stop the walk in the graph. Another important factor is the required time for convergence. The reason is that most of the observations in Random Walks are in the stationary distribution state. This state is when the matrix has converged.

According to the Perron-Frobenius Lemma [GVL96], the power iteration of a matrix W converges to its stationary distribution if the matrix is ergodic (irreducible and aperiodic). In graph terminology, ergodic refers to a connected and not bipartite graph.

Stationary distribution in Random Walks has its own applications (e.g. in PageRank). It provides the probability distribution over all nodes after convergence. However, as Berthold et al. [BBK$^+$09] proved, pure Spreading Activation is meaningless. They show that without any constraints such as fan-out or path constraint, Spreading Activation converges to query-independent fixed states. Spreading Activation provides customized solutions. It uses different activation and output functions (e.g. to control the input/output energy of a node) and heuristic restrictions (e.g. activation constraint to not activate all nodes on the propagation path) in various applications.

Self-transitivity One of the factors affecting convergence speed in Markov chain is the self-transitivity value (st). A high st value slows down the walk while a low value speeds up. The probability to reach neighbour v from node u is updated as:

$$P_{t+1|t}(v|u) = \begin{cases} (1-st)W_{u,v} & v \neq u \\ st & v = u \end{cases} \tag{2.18}$$

In compact form, the transition matrix becomes $stI + (1-st)W$ which has $st + (1-st)\lambda$ as eigenvalue. We know the largest eigenvalue of the stochastic matrix W is 1. Therefore, the eigenvalue of the new matrix remains 1 as well. In addition, the new matrix remains stochastic and its largest eigenvalue with value 1 is the same as the largest eigenvalue of the old matrix W.

In Spreading Activation, self-transitivity is referred to as “inertia”. It can be used to retain the previous state partially during iteration: $a^{(t)} = a^{(t-1)} + Wa^{(t-1)}$. In a closed form is $a^{(t)} = (I + W)^t a^{(0)}$ [BBK+09] with the same eigenvector of W. Using inertia, the weight matrix is changed to add a self loop of unit weight to each node.

We observe that the self-transitivity factor is applicable in both methods without affecting the eigenvector of the weighting matrix.

Query Dependent Routing

It is usually desirable in IR that graph traversal be dependent on the query. We look now at how this has been done in the literature, for the two methods studied.

Spreading Activation In order to avoid pure Spreading Activation, leading to query independent results, common heuristic constraints are defined:

- Distance constraint [Cre97]: imposes a hard limit on the number of iterations the activation can traverse.
- Fan-out constraint [Cre97]: to cease activation in very high fan-out nodes (indicating common nodes).
- Path constraint [Cre97]: some edges are preferred to others in transferring the activation energy. This is applicable by using preferential paths reflecting application dependent preference.
- Concept type constraint [RSA04]: some nodes are not traversed in the activation process. In this case we do not propagate through special types of concepts.
- Accumulation [BBK+09]: as a form of *iteration with memory*, this approach modifies the iterations to take into account not only the last state (of the activation propagation), but the sequence from the beginning. As a closed formula, we have:

  $$a^* = \sum_{t=0}^{\infty} \eta(t) \cdot a^{(t)} = \sum_{t=0}^{\infty} \eta(t) W^t a^{(0)} \tag{2.19}$$

  where η is decaying factor. It is used to ensure convergence.

Random Walks One of the methods in Random Walks is to compute the probability dependent on the query. For instance, Richardson and Domingos [RD02] modify the random surfer model used in PageRank by considering the query in the probability computation. Assuming $R_q(u)$ as a measure of relevancy between query q and page u, they suggest: $P_q(v|u) = R_q(u) / \sum_{k \in V(u)} R_q(k)$ where $P_q(v|u)$ is the probability of going from node u to v, and $V(u)$ is the set of neighbours of u.

Another way of query dependent Random Walks is to employ the Metropolis-Hastings algorithm (detailed explanation is in Section 2.1.4). In this method, a Markov chain is created based on existing adjacency matrix to approximate a distribution which is hard to sample from. This algorithm starts from a random candidate node as the first state. Choosing each next state of this Markov chain is based on the ratio of relevancy-of-the-candidate-node/relevancy-of-the-current-node to the query. If the candidate node is more relevant to the query, then it is chosen as target node for the next jump. Otherwise, the jump is decided with a probability.

Comparing Spreading Activation and Random Walks in Query-Dependent Routing Comparing the two methods, we find that simple constraints like the distance threshold are applicable in both methods to make the traversal query dependent. For example, we can stop Random Walks after a number of steps [CS07], or applying distance threshold in Spreading Activation [CS07, RSA04]. The path and concept type constraints in Spreading Activation (as applied in [RSA04]) make the graph traversal domain or context dependent, rather than strictly query dependent. Translated to a Random Walks model, the type and path constraints, would assign zero probability to certain edges or nodes of specific types.

By defining different types of constraints, Spreading Activation provides more options to customize the traversal. An example is to define edge constraint; another instance is to filter some edge types in the traversal. Further, we can define edge weighting which does not necessarily comply with stochastic property. Whereas in Random Walks, probabilities are assigned based on assumed relevance in the context of IR.

2.1.4 Metropolis-Hastings Algorithm

Metropolis-Hastings is one of the algorithms based on Monte Carlo Markov Chain (MCMC) to obtain samples from a complex probability distribution $\pi(x)$, which is hard to draw samples from. Suppose that we know a $\widetilde{\pi}(x)$ as a function that is proportional to the desired probability distribution $\pi(x)$, and the relation between $\pi(x)$ and $\widetilde{\pi}(x)$ is like $\pi(x) = \frac{\widetilde{\pi}(x)}{K}$. The normalizing constant K may not be known. It is not necessary to calculate the normalization factor K, as it is often extremely difficult to compute.

Through Metropolis-Hastings algorithm, we generate a sequence of sample values from $\widetilde{\pi}(x)$. The more we produce samples, the more the distribution of values approaches to the desired distribution $\pi(x)$. The samples are generated iteratively. We need a jumping distribution W, which holds the information about each sample and its neighbours. The next candidate sample (y) is dependent only on the current sample value (x), found from $W(x, y)$. At the end, we create a sequence of samples into a Markov chain which has the same stationary distribution of $\pi(x)$.

When the algorithm picks a candidate sample value, with some probability, this candidate is either accepted (the candidate value is used in the next iteration) or rejected and the candidate value is used with a probability. This probability is determined based on the $\widetilde{\pi}(x)$ value for current and candidate sample with respect to the desired distribution $\pi(x)$. The algorithm is shown in the following steps:

1. Start with initial value x that $\widetilde{\pi}(x) > 0$

2. Using the current x value, sample a candidate point y from jumping distribution $W(x, y)$, which is the probability of returning the value of y giving the previous value of x.

3. Given this candidate of y, the transition probability is calculated as follows:

   $$Pr(x, y) = W(x, y)\lambda(x, y) \tag{2.20}$$

   where λ is defined as:

   $$\lambda(x, y) = \frac{\widetilde{\pi}(y).W(y, x)}{\widetilde{\pi}(x).W(x, y)} \tag{2.21}$$

   Note that $\lambda(x, y)$ does not require knowledge of the normalizing constant because reducing the equation $\widetilde{\pi}(y)/\widetilde{\pi}(x)$ eliminates K. It is dependent on the values of $\widetilde{\pi}(x)$ and $\widetilde{\pi}(y)$; and also on the weight of edges between these two nodes. If this jump increases the density ($\lambda > 1$), accept y and set the next sample $x_t = y$. Repeat step 3. If choosing y decreases the density ($\lambda < 1$), accept u with the probability of λ, else reject it and return to step 2. In other way, when ($\lambda < 1$), we draw a sample S from the uniform probability of (0,1). If $S \leq \lambda$ we accept it, if not we reject it.

   The final value of λ can be summarized as

   $$\lambda(x, y) = min\left[\frac{\widetilde{\pi}(y).W(y, x)}{\widetilde{\pi}(x).W(x, y)}, 1\right] \tag{2.22}$$

In order to reach a unique equilibrium state for a Markov chain, the chain should be ergodic (both irreducible and aperiodic). There may be different jumping distributions for Metropolis-Hastings—based on the way we generate the next candidate to generate the chain of samples. Two general approaches are [Wal04]: 1) Random Walks - the new state y is dependent on the current state x. 2) Independent chain - the probability of jumping to point y is chosen from a distribution of interest, independent of the current value. This method is usually used in asymmetric Metropolis-Hastings: a jump distribution is symmetric if $W(x, y) = W(y, x)$. Where in asymmetric version $W(x, y) \neq W(y, x)$.

In practice, either Metropolis-Hastings accepts y as next candidate (when $\lambda > 1$) and $x_t = y$, or modifies the weight of $W(x, y)$ with the factor of λ. The weight of this link for the next step is $W(x, y) \cdot \lambda$. According to the stochastic property, sum of the probabilities on outgoing edges of a node is 1. As in each step, edge weights are adjusted by Metropolis-Hastings, the sum may get lower than 1. In this case, the link is accepted with the probability of $\lambda < 1$. The decreased value of the weight of this edge is given as self-transitivity value to the node. This implies that staying in this state is preferred to choosing that specific jump.

The application of Metropolis-Hastings method in information retrieval is mostly in search in peer-2-peer networks [FRA$^+$05, AFJG06]. Ferreira et al. [FRA$^+$05] have designed a protocol for locating a specific object regardless of the topology of the network through uniform sampling from peer-to-peer networks. Zhong et al. [ZSS08] use random walks and focus on convergence time for different network sizes. They investigate the probability distribution of visiting nodes. In order to go beyond peer-2-peer networks and apply Metropolis-Hastings in IR, we need a jumping distribution, i.e. weighted links between nodes. Such links may be similarity/semantic or a mixture of the two. The difficulty, as we will see, is ensuring the stochastic and ergodic nature of the chain.

After we obtained an understanding about Metropolis-Hastings algorithm, we define some concepts in this algorithm in more detail as follows:

Reversibility The candidate-generating density W holds the stochastic property of $\sum_{y=1}^{k} W(x,y) = 1$, where k is the number of objects that are possible to reach in one step from x. If $W(x,y)$ satisfies the reversibility condition for all x, y, we have:

$$\widetilde{\pi}(x)W(x,y) = \widetilde{\pi}(y)W(y,x) \tag{2.23}$$

However, usually this is not the case. For instance, for some x, y we have:

$$\widetilde{\pi}(x)W(x,y) > \widetilde{\pi}(y)W(y,x) \tag{2.24}$$

This way, the process moves often from x to y and rarely vice versa. To reduce the number of moves from x to y, later in Metropolis-Hastings algorithm, λ is modified to: $\lambda(x,y) = min\left[\frac{\widetilde{\pi}(y).W(y,x)}{\widetilde{\pi}(x).W(x,y)}, 1\right]$. If $\lambda = 1$, then $x_t = y$, otherwise if $\lambda < 1$, we select y with probability of λ.

Transition Function According to Metropolis-Hasitng algorithm, we sample from $W(x,y) = Pr(x \longrightarrow y|q)$ and accept the move with probability $\lambda(x,y)$. The transition probability is given by:

$$Pr_q(x,y) = W(x,y)\lambda(x,y) = W(x,y).min\left[\frac{\widetilde{\pi}(y).W(y,x)}{\widetilde{\pi}(x).W(x,y)}, 1\right] \tag{2.25}$$

This implies on how we define high-order transition probabilities after t steps:

$$Pr_q^{t+1}(x,y) = \sum_{i=1}^{k} Pr_q^t(x,z_i)(z_i,y) \tag{2.26}$$

where k is the number of common nodes z between x and y, and Pr^t is the transition probability of starting from x and moving t steps further.

Stationary Distribution/Convergence We want to achieve a query dependent stationary distribution such that:

1. The probability in node x is proportional to the probability that this node is relevant to the query
2. At any other node (non-relevant) the probability is zero

Mixing Time Number of steps to reach a stationary distribution is a key issue in successful implementation of Metropolis-Hastings, which is called mixing time. After this time, the probability to be in any specific node x will be $\pi(x)$.

2.2 Related Work

We divide the related work on multimodal information retrieval (IR), in respect with relations between information objects, in two categories of non-structured and structured IR. Fundamental IR is based on retrieval from independent documents. We refer to this type of IR as non-structured IR, in which an explicit relation between the information objects is not considered. We provide a brief overview on this part of related work in Section 2.2.1. Leveraging a graph of relations between information objects imposes structured multimodal IR. We provide the related work in this area in Section 2.2.2. Another challenge in the context of multimodal retrieval is how to fuse the results of different modalities or how to use a combination of different features in the retrieval process. Hence, we provide an overview on fusion methods in Section 2.2.3. Finally, as a part of the study in this thesis is primarily based on Poly-representation principle in IR, we provide an overview on the related work in this area in Section 2.2.4

2.2.1 Non-structured Multimodal IR

Most popular search engines like Google, Yahoo, and Bing, build upon text search techniques by using e.g. user-provided tags or related text for images or videos. This approach for multimodal search ignores the visual information completely and does not always achieve satisfying results [HKC07, CMS07, DJLW08, HCY08, LSW10]. Approaches which leverage only text search techniques have limited access to the data as they ignore totally the information of visual content and the indexes do not contain multimodal information. Another reason is that the surrounding text is usually noisy and this decreases the performance of text-based multimodal search engines. Content-Based Image Retrieval (CBIR) is one of the earliest methods that started to consider image content in the retrieval process. Datta et al. [DJLW08] did a thorough survey on this subject. Many systems considered similarity measures only based on the content of the images, which are called as pure CBIR systems [CBGM02, MM99, PPS96, SWS$^+$00]. One of the earliest research in this category is to utilize low level features representation of image content and calculate the distance to the query examples to capture similarity.

Top documents in the result list are the ones which are visually similar to the query example. On the other hand, systems that include a combination of text and image content in addition to a flexible query interface are considered as composite CBIR systems [FPZ04, JWL06], which complies principally with the definition of multimodal IR. This category is suitable for web image retrieval as most images are surrounded by tags, hyperlinks and other relevant metadata. For example, in a search engine the text result can be reranked regarding the similarity of the results to a given image example of the query [Bin15, Goo15]. Kennedy and Chang [KC07] use concept detectors to improve the video search performance. They mine the initial search result to discover contextual related concepts. Their method is unsupervised. They leverage these concepts to refine the initial video search result. Martinent et al. [MS07] propose to generate automatic document annotations from inter-modal analysis. They consider visual feature vectors and annotation keywords as binary random variables. They create the multimodal indexing process through inter-modal analysis. In particular, they use the textual modality and image features to annotate images in a test collection. The more mutual information between modalities exists, the stronger is the dependency between the modalities. Lazaridis et al. [LARD13] leverage different modalities such as audio and image to perform a multimodal search. Their project named I-Search is a multimodal search engine, in which a multimodality relation is defined between different modalities of an information object, e.g. between an image of a dog, its sound (barking) and its 3D representation. They define a neighbourhood relation between two multimodal objects which are similar in at least one of their modalities. However, in I-Search, neither semantic relations between information objects (e.g. a dog and a cat object) is considered, nor the importance of these relations in answering a user's query.

One of the well-known methods in IR to improve the results is relevance feedback, which can be applied on multimodal IR as well. In this method, the system receives information from an external source to provide a better representation of user needs. This information can be obtained from assessors or user clicks who mark the results as relevant or non-relevant. Pseudo-Relevance Feedback (PRF), also known as blind relevance feedback, automates the manual part of relevance feedback. This method is usually based on three steps: first, selecting the pseudo-positive and pseudo-negative samples from the initial ranked list; second, training classifier on both categories of the samples, and third, reranking the results with the relevance scores predicted by the classifiers. As an instance of leveraging PRF in multimodal retrieval, Yan et al. [YHJ03a] propose an algorithm for video retrieval based on the results from both text and image modalities. For the image retrieval, they use basic nearest neighbour image matching combined with classification based PRF. In the retrieval algorithms with poor performance, it is no more appropriate to consider the top-ranked results for positive feedback. Using negative pseudo-relevance feedback, Yan et al. [YHJ03b] sample from bottom-ranked results for negative feedback in video retrieval. They consider the positive examples from user's query, and for negative examples they use the worst matching examples identified based on a generic similarity metric.

In addition to using features from different modalities, the interaction process with users is considered in multimodal retrieval as well. Cheng et al. [CYK$^+$05] suggest two interactive retrieval procedures for image retrieval. The first method incorporates a relevance feedback mechanism based on textual information while the second one combines textual and image information to help users find a target image. Hwang and Grauman [HG10] have also explored ranking object importance in static images, learning what people mention first from human-annotated tags. One of the ideas in query formulation for multimodal IR is to integrate different modalities to initialize the query. Hubert and Mothe [HM09] suggest a combination of ontology browsing and keyword-based querying. Combining these two modes enables users to complement their queries with keywords for which they do not identify corresponding categories. Natsev et al. [NNT05] add to this method leveraging a bagging strategy. They create multiple ranking lists using only positive examples and use both positive and negative examples to learn models for a query or a concept. These lists are combined to generate the final results.

Wang et al. [WYJ16] propose a model for search which is based on semantic similarity of a query and documents. They target social network systems, in which social documents are fed into a concept extractor. Queries are considered in two modalities of text and image. For each of these modalities they find its concepts based on the concepts of the top ranked results of that modality. They calculate the relevance to the query based on two defined measures of informativeness and semantic relevance. This model can be mapped to Astera by adding a facet that contains semantic information, which Wang et al. [WYJ16] propose to provide for an information object. Semantic similarity can be computed between semantic facets of each two information objects. Astera has the advantage of considering different relation types between information objects, while Wang et al. [WYJ16] calculate the relevance score only based on semantic facets of the information objects.

Sometimes in literature, multimodal IR is called as reranking. For instance, Mei et al. [MRLT14] definition of reranking (leveraging different modalities, like image or video, rather than only text to find a better ranking) is compatible to what we define as multimodal IR. Mei et al. [MRLT14] did a thorough survey on reranking methods in multimodal information retrieval. They categorize related work in four groups: 1) Self-reranking: mining knowledge from the initial ranked list. They define self-reranking method as discovering relevant visual patterns from the initial ranked list. The top ranked results are objective for mining relevant information. This is based on the analysis on click-through data from a large search engine log that users are more interested in the top-ranked part of search results [WZZL10]. 2) Example-based reranking: leveraging few query examples (e.g. images or video clips) that the user provides along with the textual query. These examples are used to improve the search performance. 3) Crowd-reranking: utilizing knowledge obtained from crowd as user-labelled data to perform meta-search in a supervised manner. 4) Interactive reranking which reranks involving user interactions.

Regarding this categorization, graph-based methods belongs to self-reranking category.

We explore this category in more detail. Mei et al. [MRLT14] divide the self-reranking methods in the following sub-categories based on how the initial ranked list is mined. 1) Clustering-based reranking [WZZL10, WJH$^+$07]: this method is based on the assumption that relevant documents tend to be more similar to each other. Therefore, clustering the initial ranked results filters non-relevant documents. However, the challenge is to cluster initial noisy result and rank inside each cluster. 2) Pseudo-relevance feedback (PRF) which is described earlier in some paragraphs above. 3) Graph-based methods: Mostly the methods in this category are inspired by Page-Rank techniques for document search. In this way, the relevance score of a document is propagated through the entire graph structure, and the traversal method is Random Walks. Usually a graph $G =< V, E >$ is made based on the initial ranked list, where $v \in V$ corresponds to a node of any modality and $e \in E$ is the similarity link between the two objects. The links between the images/videos are similarity links. The initial relevance scores of each document can be propagated to other similar nodes until the graph reaches a stationary distribution state. The Random Walks is biased with the initial score, whereas the stationary distribution probability as final relevance score is used for reranking.

Structured IR is similar to the above graph-based methods, with the difference that the nodes in the graph are not necessarily from top ranked list. This means that our definition of structured IR is not limited to only a part of information objects of a sample collection.

2.2.2 Structured Multimodal IR

By structured IR we denote those approaches that consider the explicit relations between information objects. Usually in such models, a graph is created, which may be based on similarity or semantic relations between information objects. Nodes and edges may hold different definitions in each model. For example in Liu et al. [LLH$^+$07] graph-based model, the video search reranking problem is formulated in a PageRank fashion using Random Walks. The video shots are the nodes and the multimodal similarity (i.e., the textual and visual similarities) are taken as hyperlinks. The relevance score is propagated per topic through these hyper-links. Another example of leveraging graph-based search is what Srinivasan and Slaney [SS07] did by using a bipartite graph. They add content based information to image characteristics as visual information to improve the performance of their algorithm. They use Bag of Visual Words to represent image content, and apply Salient-point detection on images through SIFT descriptors. Each descriptor represented by codebook index provides the linguistic description of an image. A bipartite graph is created. One partition corresponds to linguistic words and one is mapped to visual words. Their model is based on Random Walks on bipartite graphs, representing joint models of images and textual content. The score of a link is based on the number of times a visual word is selected for an image. The stationary distribution in this graph shows the visual words that are more related to an image. This model is limited to images and only one feature of visual words. In Astera we have no limitation on the number and type of features used for an information object.

Jing and Baluja [JB08] cast the image-ranking problem into the task of identifying "authority" nodes on a visual similarity graph and propose VisualRank to analyze the visual link structure among images. Zitouni et al. [ZSOD08] find that the subset of image results based on text search which are relevant, are highly similar. They show that in a fully connected graph of images with similarity links between them, these relevant images create a dense component. Based on this observation, after they create a graph of images, they find the densest component in the graph and assign higher ranks to these images.

Clements et al. [CDVR10b] propose the use of a personalized Random Walk on a tripartite graph, which connects users to tags and tags to items. The stationary distribution of the Random Walk shows the probability of relevance of the items. Jing et al. [JB08] employ PageRank to rerank image search results. The hyperlinks between images are based on their visual similarities. They choose the "authority" nodes to answer the image queries. Yao et al. [YMN10] make a similarity graph of images and aim to find authority nodes as results for image queries. Through this model, both visual content and textual information of the images is explored. Hsu et al. [HKC07] leverage context reranking as a Random Walk over a graph of video stories. The links in the graph are based on similarities between different video stories. The final scoring value is a combination of the initial text similarity and stationary distribution scores. Tian et al. [TYW$^+$08] use a graph-based approach to formulate video search reranking into a Bayesian optimization problem. They maximize the consistency of ranking score between visually similar video shots, while they maintain the minimum ranking distance.

The structured search engine NAGA [KSI$^+$08] provides the results of a structured (not keyword) query by using subgraph pattern on an Entity-Relationship graph. They find subgraphs of the whole graph which matches the query graph. They define matching for each word vertex in the query graph, if there is the same vertex in the whole graph. Targeting RDF data, Elbassuoni and Blanco [EB11] select subgraphs to match the query and do the ranking by means of statistical language models. As a desktop search engine, Beagle++ utilizes a combination of indexed and structured search [MPC$^+$10]. Magatti [MSBT11] provides a combination of graph and content search as well. For example, in an organization, members have hierarchical relations by their roles, meanwhile there are related documents to them. The related concepts are leveraged for unsupervised search reranking. They train an SVM model based on the pseudo-positive and -negative samples from initial ranked list of results. Wang et al. [WLT$^+$12] propose a graph-based learning approach with multiple features for web image search. They create one graph per feature. Links in each graph are based on the similarity of the images in each graph based on that feature. The weight of different features and relevance scores are learned through this model.

Related research on the ImageCLEF 2011 Wikipedia collection is generally based on a combination of text and image retrieval [TPK11]. To our best knowledge, there is no approach that has modelled the collection as a graph structure and no approach has therefore leveraged the explicit links between information objects and between information objects and their features.

According to the categories defined by Mei et al. [MRLT14], our model belongs to the first category, as we create a graph of information objects, and also to the second category, as we use query examples in addition to the textual query to find relevant information objects. However, it is different in creating the graph, as we include all information objects in the collection, not only the top-ranked results of facets' search. We use the top results as initiators in the graph as starting points of the traversal in the graph. In addition, we utilize different types of relations between information objects of a collection. In related work it is mostly similarity links, however in our model we add part-of and semantic links as well.

Graph-based Learning Graph-based learning approach is one of the semi-supervised learning methods which has attracted great research interest in the past years [ZBL+04, ZGL+03]. In this approach, a graph is defined where the vertices are labelled and unlabelled examples in the dataset, and edges (may be weighted) reflect the similarity between each two examples. These methods can be viewed as estimating a function f on the graph with satisfying two conditions: 1) The function should be smooth on the whole graph. 2) The function should be consistent with prior information, such as labeling information of several instances. The availability of large collections of data and limited labelled data has turned semi-supervised learning as one of the methods to deal with large amount of data [CSZ09]. Yang et al. [YH10] extract different features from each image. They create a training set based on labelled search results. Yuan et al. [YHWW06] apply a graph-based learning method named manifold-ranking on video annotation, and He et al. [HLZ+04] use the same algorithm in image retrieval. They use supervised learning to perform reranking.

There are also graph-based learning approaches that are used for reranking [HKC07, JB08, TYW+08]. Wang et al. [WLT+12] propose a multi-graph learning method for web-based image retrieval. They define different graphs for each feature of image modality. Each element in the graph matrix is the similarity between x_i and x_j indicating the same feature of image i and image j. They create an elementary ranked list which is updated in each iteration of learning to minimize the rank score difference of highly similar images. They assign different weights for each feature and use linear combination on their corresponding graph matrices.

However, Astera is a different approach towards multimodal retrieval. The proposed model of Wang et al. [WLT+12] is limited to image retrieval, whereas in Astera we can have different modalities. Further, we combine the multi-layer graph in one graph. We consider each information object as one node. Its different features are connected as separate nodes (facet node type) to the information object node. Moreover, in Astera, in addition to similarity links we consider semantic links between the information objects. Structurally it is allowed in Astera to have multiple edges between two information objects.

Semantic link search There are related works in utilizing semantic links in the task of IR [TdM14, CLM$^+$13, DMDH02, SSH14, TDCM12]. One of the relation types in Astera is semantic links. If we model a collection containing only semantic links, then we can relate Astera to the related work in semantic web. For example, Tonon et al. [TDCM12] use a hybrid search on Linked Open Data. They follow semantic links in the scope of one and two steps, and show the effect of different types of semantic links in precision and recall. They retrieve better result by exploring selected semantic links than any semantic links (e.g. following *sameAs* links gives better performance compared to following *wikilink* types). Experiments are performed on the effect of different semantic links on precision and recall in SemSearch collection 2010 and 2011. They rank all properties with likelihood of leading to a relevant result in one and two steps. They find that although some links like *wikilink* is highly frequent, it does not lead to promising pages. However, links like *sameAs* – linking the same pages with different languages – result in higher precision and recall.

Some related works leverage semantic links to find interesting subgraphs. An example is the work by Chen et al. [CLM$^+$13] to create a multimedia story of related images and videos through navigating related nodes in Linked Open Data (LOD). They utilize a node recommendation mechanism to choose next nodes to follow. Choosing a link is based on interestingness of each connection. They use a machine learning algorithm to calculate the interestingness of connections based on rarity of the subject or object with a specific predicate, and the number of incoming links to a subject or an object.

Tiddi et al. [TdM14] propose to follow semantic links to search for the objects sharing the same path or walk. They create a cluster of the link traversal to discover new knowledge. They create clusters from the nodes visited on the way, indicating the optimal cluster as C^+ and not desired cluster as C^-. The defined traversal method does not need prior knowledge and neither uses indexing. They simply traverse the links to gather new unknown knowledge. They use A^* search to find the lowest cost path from a given initial node to another node. They do not have a preference but only any node in the distance of j from the initiative node. Each time a new path is chosen, a new part of the graph is revealed. They chain the walks to the entities discovered at the current iteration and evaluate them based on a defined F measure. Doan et al. [DMDH02] introduce GLUE as a system which utilizes machine learning techniques to find Semantic mappings between different ontology elements. Safadi et al. [SSH14] utilize the combination of textual and visual features to improve video retrieval. They use video subtitles to enrich with textual information the scenes of a movie. They use WordNet to capture concepts related to the textual information, and to map visual scenes to the concepts. They improve the performance of video retrieval by combining the text and semantic information they obtain.

In Astera we provide a hybrid search model that is not limited to work on RDF data. Our work deals with different modalities and relation types.

2.2.3 Multimodal Fusion

Pradeep et al. [AHESK10] provide a survey on multimodal fusion for multimedia analysis. They give an overall view on fusion methods with different levels of multimodal fusion. We briefly summarize it here.

Different Fusion Levels

Fusion of different modalities is performed at two levels, feature level (known as early fusion) and decision level (known as late fusion). In early fusion, different features are extracted, concatenated and make a larger feature vector, which is given as input to the search task. Existing studies reveal that high dimensional feature space leads to similar distances between the samples [BGRS99, WCCS04a], which degrades performance in distance based algorithms. This is one of the main drawbacks of using early fusion with combination of different features. One way to deal with this issue is to use separate ranked list for each feature, and then fuse the results [WCCS04b]. This method is usually called "late fusion", "multimodal fusion" or "multimodality learning" [SWS05]. In late fusion, local decisions are made based on individual features, then a fused decision is taken. This approach is mostly applied in image and video retrieval analysis [SC96, VNH03]. Chang et al. [WCCS04a] provide a framework to find independent modailities for multimodal fusion. They find the best combination of multiple modalities through supervised learning.

Wang et al. [WHH$^+$09] propose an algorithm which integrates the graph representations from multiple features for video annotation. In this algorithm, they create a graph for each feature. They fuse multiple graph features into a regularization framework. Then, they run semi-supervised learning on this fused graph. Zhao et al. [ZYYZ14] make a graph of each feature, test its performance and combine all features in a multimodal learning. They use three levels of low-, medium - and high-level features. Hybrid multimodal fusion is a hybrid fusion strategy which combines both feature- and decision-level strategy. In principle graph-based approaches can be seen as non-linear fusion of heterogeneous ranked lists.

With a labelled fusion set, the task of fusion can be formulated as a learning issue. For instance, Iyengar et al. [IN03] and Snoek et al. [SWS05] perform the fusion with Support Vector Machine (SVM) models. Wu et al. [WLCS04] propose a combinatory model for fusing different features with two constraints. First an optimal weighted linear combination of single modalities is learned. Second, they maximize the feature independence. They run separate classifiers for each feature, then utilize a meta-classifier on the result of these individual classifiers.

Both feature level and decision level fusion methods can be divided in three categories as follows:

(1) Rule-based Fusion Method First are the rule-based fusion methods which include the most widely used approach of weighted linear fusion. Weighted linear fusion can be applied on feature level and decision level fusion. In this method the decision of different methods may use normalized weighting. There are different methods of weight normalization such as z score, min-max, decimal-scoring or sigmoid function [JNR05a]. Wang et al. [WKYJ03] apply linear weighted fusion on different feature results of video color, motion and texture results. The same approach is applied in Kankanhalli et al.'s [KWJ06] work on results of color, motion and texture features for face detection, monologue detection and traffic monitoring. In late fusion, Lucey et al. [LSC01] use individual decisions of audio MFCC and video Eigenlip features, and combine them linearly with different weightings. They use the final result for recognizing spoken words. Hua and Zhang [HZ04] apply weighted linear combination on individual results of six image features (color histogram, color moment, wavelet, block wavelet, correlagram, blocked correlagram) for the task of image retrieval. Yan et al. [Yan06] use the same approach on Text (closed caption, video OCR), audio, video (color, edge and texture histogram), and motion for video retrieval.

Another approach of rule-based fusion methods is the majority voting method. It is a special case of weighted combination that an equal weight is given to all individual decision results. The final result is based on the majority of the classifiers that reach similar decisions [SP04]. The third approach in rule-based fusion is custom-defined rules. Holzapfel et al. [HNS04] use this approach in multimodal integration. They combine 3D pointing gestures and speech features of robots in a room as natural interactions. Based on the n-best lists generated by each of the event parsers they perform multimodal fusion in the decision level.

(2) Classification based Method The second category of feature fusion comprises the classification-based methods. A range of classification techniques are used to classify the multimodal features into different classes. Given D features, we need to fuse the result of D classifiers, one for each feature. One of the very well-known methods is Support Vector Machine (SVM). For example, the result of different intermediate classifiers for video, audio and text scores is concatenated into a large vector and is given to another classifier to identify the concept class. Adams et al. [AIL$^+$03] leverage a late fusion approach to detect semantic concepts in videos (e.g. sky or fire-smoke). They use separate classifiers for each modality (textual, visual and audio) and create a semantic vector of scores of these intermediate concept classifiers. This vector is given as a semantic feature to a meta-classifier. Iyenger et al. [INN03] use the same method on audio and video results for semantic concept detection.

Bayesian Inference is another method located in the classification-based methods. In this method, the multimodal information is combined based on the rules of probability theory [LYS02]. It can be applied both at the feature level and at the decision level. The result from different classifiers or the observations obtained from multiple modalities are combined, and an inference on the joint probability is derived [RG03]. In Bayesian inference, we assume that the modalities are independent and the joint probability of hypothesis H is computed.

$$p(H|I_1, I_2, ..., I_n) = \frac{1}{N} \prod_{k=1}^{n} p(I_k|H)^{w_k} \tag{2.27}$$

where N is the normalization factor of posterior probability $p(H|I_1, I_2, ..., I_n)$, term w_k is the weight of kth modality, and $\sum_{k=1}^{n} w_j = 1$. One of the main advantages of using Bayesian inference is the possibility of including prior knowledge about the likelihood of the hypothesis H in the inference. However, without knowing about the prior probabilities Bayesian inference does not perform well [Red07]. It can be based on fused decision or combined feature vectors. Pitsikalis et al. [PKPM06] apply Bayesian inference on MFCC feature of Audio and Shape and texture features of video for speech recognition. Xu and Chua [XC06] apply this method on a hybrid decision result on multiple features of audio, video, text and weblog in sport video analysis.

(3) Estimation-based Fusion Methods The methods in this category are used to estimate the state of a moving object by processing data from multiple modalities. Usually these methods are used to track an object based on the information from multiple modalities as video and audio. Kalman filter [Kal60, RG99] is one of these methods, in which real-time dynamic low-level data can be processed. Leo et al. [LGG04] proposed a method to estimate the position of a single speaker based on Kalman filter. They utilize audio features like velocity and acceleration of the single sound source, and also the camera image point. Town [Tow07] applies a decision level fusion on Video (face and blob), and ultrasonic sensors features for human tracking.

Estimation-based methods are mainly used to find a moving object, which does not comply with the target of this thesis. Further, in the current version of Astera we use no learning in the feature or decision level. Therefore, no classification-based method is used. However, we connect extracted facets to their information objects and treat them as individual nodes. This provides possibilities for both early and late fusion. We provide the linear late fusion of normalized weighted scores of facets like in the rule-based fusion method. In addition to the first level of fusion, we leverage the graph model of a collection to leverage relations as well.

2.2.4 Poly-representation

The poly-representation principle is based on utilizing different cognitive and functional representations of information objects to enhance the quality in IR [LIK06]. We refer to the definition that Ingwersen and Järvelin [IJ06] gave on poly-representation: “the more interpretations of different cognitive and functional nature, based on an IS & R situation, that point to a set of objects in so-called cognitive overlaps, and the more intensely they do so, the higher the probability that such objects are *relevant* (pertinent, useful) to a perceived work task/interest to be solved, the information (need) situation at hand, the topic required, or/and the influencing context of that situation.”

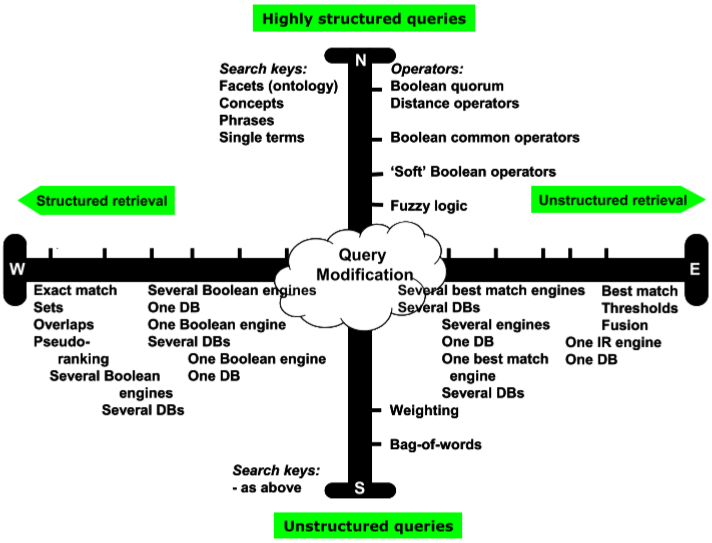


Figure 2.1: The extended structural poly-representation continuum ([LI05])

White [Whi06] shows that relevance feedback methods based on the poly-representation idea improve the retrieval performance. Further, poly-representation has been used to help the user describe her information need in different forms [Ing96] and increases IR performance [KF07].

Larsen et al. [LIK06] show a spectrum on query modifications from highly structured queries to unstructured queries (Figure 2.1). The query formulated with Boolean operators is a highly structured query compared to Bag-of-words query which is in the unstructured query end-point (pole E on Figure 2.1). Using facets with different weighting or using concept structure with operators are considered as structured queries as well.

In addition, they show another continuum for search engine retrievals in Figure 2.1, which ranges from exact match on structured pole W to the best match on unstructured pole. For example, Larsen [Lar02] and Christoffersen [Chr04] who use "One Boolean engine" and "Several DBs" are placed at the structured retrieval axis. Larsen [Lar04] applies multiple databases and is located more in unstructured retrieval pole. Kelly et al. [KDF05] are placed in the SE part as they use one system and no different representations of documents but with weighting of different representations of the user's cognitive space. The common baseline retrieval in IR evaluation tests is placed at the outmost corner of SE.

Larsen and Ingwersen [LI05] divide the poly-representation in two kinds of representations: first cognitively different representations which is based on interpretations by different users; second functionally different representations derived from the same actor such as writing image and diagram captions or adding references in a document.

In cognitive poly-representation scenarios the cognitive space of the user is utilized. In this approach, beside different representations of the query, different actors in the interaction process contribute in enriching the representations (e.g., how the user formulates her request and problem state [IJ06]). Ingwersen and Järvelin [IJ06] list many possibilities to enrich the cognitive representation such as the search profile content of the user, which is utilized in recommender systems. Another way is to use information extracted from the current user such as her domain knowledge or describing the problem statement. Kelly et al. [KDF05] performed experiments on this type of poly-representation . They asked 13 users with 45 topics on 2004 TREC HARD track. There were four questions that the users had to answer before they do the search: the time of searching, describe if she knows already about the topics, what she wants to know about the topic and any other keywords or tagging about the topic.

There are different ways of representing an information space. The varieties are combination of two or more databases, applying different indexing methods, applying ontologies on data, or utilizing external features contextual to the document contents like inlinks/outlinks [LIK06]. Skov and colleagues [SPLI04] use different representations of a document like abstract, titles, or titles of the references as different representations of a document. Skov et al. [SLI08] run poly-representation tests on overlaps between five functionally and/or cognitively different representations of documents. They show that the results generated from the overlap of three or four representations of different nature are higher than those generated from two representations or only single field. These experiments are run with both structured and unstructured queries. Highly structured queries show higher precision compared to queries expressed in natural languages. They conclude that using structured queries is a necessity to implement poly-representation more efficiently.

Further, Larsen et al. [LIL09] show that the result fusion based on cognitively/algorithmically different retrieval models perform significantly better. They define two types of dissimilarity: cognitive dissimilarity which is between different IR models (TF.IDF, BM25, LM); functional dissimilarity which is based on different versions of the same fundamental retrieval model (different values for b coefficient in BM25). They compare combinations of equally/unequally performing and cognitively dissimilar models performance with fusion of similar retrieval models. They design experiments to validate poly-representation for the retrieval platform. They compare the performance of 11 logical late fusion combinations with the performance of four individual models and their intermediate fusions based on the principle of poly-representation. Their results show that fusion of IR models does not always provide better results than an individual IR model. Only fusion of equally good IR models but conceptually/algorithmically dissimilar ones may outperform their intermediate fusions [LIL09]. For example, a retrieval model based on BM25 is conceptually and algorithmically different than one based on the language modelling (LM). They define three factors for best possible combination of different IR models. First is the level of dissimilarity conceptually/algorithmically between the constituent IR models; second, how equal and third, how well the component models perform.

CHAPTER 3

Astera Model

As mentioned in Chapter 2, there are several graph-based models for information retrieval in literature. Most of these models create a graph based on similarity links between images or videos. However, the data available today, especially user-generated data, is multimodal. For example, the user posts a video in a social network. Comments are written about this video (text modality), or some related images are posted. This is an example that shows the heterogenity of related information objects in today's generated data. Moreover, there are different relation types between these information objects, which is clearly beyond only similarity relation type. In principle, these objects are *related*, which could be from different aspects. To address these challenges in multimodal IR, we propose a model named Astera—a generic model for multimodal IR. In this chapter, we describe the characteristics of this graph-based model in Section 3.1. We represent information objects from different modalities and their relationships in this model. In the first part, we define the characteristics of our graph-based model. We then define our faceted search to compute the relevancy of an information object to the query. In the last part, we define our hybrid search on the graph. Afterwards, we instantiate the model based on the test collection in Section 3.2. We describe our evaluation methodology in this section in detail.

3.1 Model Definition

Astera is a general framework to compute similarity. We see the information objects as a graph $G = (V, E)$, in which V is the set of vertices (including data objects and their facets) and E is the set of edges.

In this model, we define facet as an inherent feature or property of an information object, otherwise referred to as a representation of the information object. This definition allows considering a document under several points of view, each one being associated to a possible space. For instance, text documents have primarily a textual facet, but also others such as stylistic/layout facets (covered partially by image features), the number of images they contain, or have time/versioning aspects (recency of information). Another example is image files which primarily have image facets (comprising several feature spaces such as color, edge, rotation) but also other facets such as written information (as detected with OCR), time, owner, etc. Each of these is a node linked to the original image object. Each object in this graph may have a number of facets. This has support in the principles of Information Retrieval, most notably in the theory of poly-representation [LIK06]. The aim is to leverage cognitive and functional representations of information objects to improve IR results.

We define four types of relations between the information objects in the graph. Together with their weights, they are as follows:

- **Semantic** (α): any semantic relation between two objects in the collection (e.g. the link between lyrics and a music file). The weight is defined based on the number of semantic relations between two nodes [RSA04]: $w_{ik} = N_{ik}/N_i$, where N_{ik} represents the number of information objects that both nodes i and k are related to, and N_i is the number of information objects connected to the node i.
- **Part-of** (β): a specific type of semantic relation, indicating an object as part of another object, e.g. an image in a document. This is a containment relation as an object is part of another one, and therefore we set the default weight to 1.
- **Similarity** (γ): relation between the facets of the same type of two information objects. The weight is the similarity value between the facets according to some facet-specific metric. For instance, we can compute the similarity between Edge Histogram facet of two images, or TF.IDF facet of two documents.
- **Facet** (δ): linking an object to its representation(s). Weights are given by perceived information content of features, with respect to the query type. For instance, with a query like "blue flowers", the color histogram is a determining facet that should be weighted higher. We find different facet weights experimentally. These weights should be learned for a specific domain, and even for a specific query e.g. via relevance feedback.

In addition to the edge weights just defined, we consider the use of a self-transitivity value (st) to emphasize remaining in a specific state. This value leaves part or all of the initial score/energy with the current node. The self-transitivity is also justified by the necessity of the graph to be aperiodic in order to reach a stable distribution when traversing it using Random Walks.

We can define a transition matrix W as follows:

$$W_{v|u} = \begin{cases} (1-st)w_{uv} & u \neq v \\ st & u = v \end{cases} \tag{3.1}$$

where $u, v \in V$.

A sample of the model is shown in Figure 3.1. We show the relations between three documents of *Elvis Presley*, *Graceland* and *Rockability*. The relation between each document and contained images is shown by β relations. For example, the node *Rockability* has two images. There is δ relation between each node and its facets, e.g. between a document and its TF.IDF facet, and an image and its Color Histogram facet. The similarity relation between the nodes is shown via γ relation between their correspondent facets, e.g., between TF.IDF facets of two documents or between Edge Histogram facets of two images. Semantic relations are shown between *Elvis Presley* and *Rockability* (via semantic relation *Genre*), and between *Elvis Presley* and *Graceland* (via semantic relation *Resting place*).

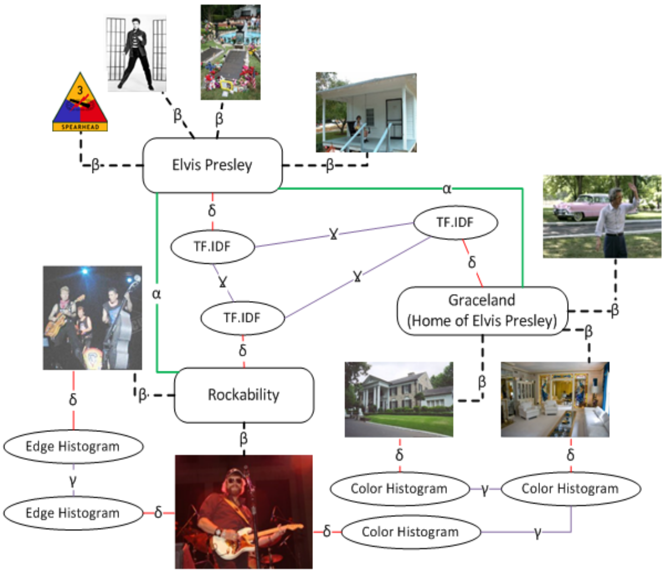


Figure 3.1: Different types of relations in the model

3.1.1 Query Formulation and Relevancy Computation

A query may contain different modalities. Our approach is to decompose the query into a list of facets of each modality. For instance, if a query is formed of keywords and image examples, we leverage the combination of its textual and visual facets. Relevance of a document to a text query is defined as similarity of the vectors of the words in the document to the query. How to define the similarity of a multimodal query with different types of information objects in the graph is a challenge.

We define the node $u \in V$, where V is the set of vertices in the graph. It has facet of $f(u)$, where $f(u) = \{v \in V, \exists \delta(u, v)\}$. For simplicity of notation, we will denote the ith facet of node u as u_{f_i}, although it is still a node. We define the same for the query q as follows:

$$u = \cup_{i=1}^{n} u_{f_i}$$
$$q = \cup_{j=1}^{m} q_{f_j}$$
$$l = |\{q_f | \overline{q_{f_i}} = \overline{u_{f_i}}\}|$$

where N is the number of facets of the node u, m is the number of facets of the query, and l is the number of the common facet types of $\overline{q_{f_i}}$ and $\overline{u_{f_i}}$. We define the relevance score value function (RSV) as follows:

$$RSV(q, u) = \sum_{i=1}^{l} norm(sim(q_{f_i}, u_{f_i})) \cdot w_{f_i} \tag{3.2}$$

where sim is the similarity function between the two facets, $norm$ is the normalizing function and w_{f_i} is the weight of facet f_i for this query. We compute the similarity (sim) between facet values of q_{f_i} and u_{f_i}. Usually the value of a facet is in the form of a feature vector. In case of no common facet, the RSV function output is zero.

We compute the similarity of all objects for each query facet and normalize. As we have a multimodal graph and in each step we may visit a node with different modality, we require a normalized value to be able to compare the relevancy values.

Different modalities have different facets. Reaching nodes of the same modality of query examples, we have all the facets in common (e.g. an image query and an image node). Visiting nodes with different modality than query examples, we calculate similarity for common facets. For instance, if we have an audio object and an image query, we can compare their textual facets (the TF.IDF facet of image metadata or OCR'ed text and TF.IDF facet of the audio tags or lyrics).

Facet Fusion As described in detail in Chapter 2 Section 2.2.3, the model supplies different levels and methods for feature fusion. Two well-known methods are early and late fusion. Early fusion is referred to as fusion in feature space, uni-modal features extracted from different modalities, concatenated into a long feature vector. This is opposed to late fusion, where we integrate the individual similarity results of different features of different modalities. Early fusion is conceptually more attractive because it addresses the problem of representation for multimodal objects. However, it suffers from curse-of-dimensionality, whereas late fusion is influenced by the quality of ranking of individual features.

The effect of each facet on transferring the energy is controlled by weight of that facet. In our model, facet nodes may have an initiating score. This implies that a similarity retrieval has already been applied. They have zero scores, if they are not selected in the top results. Each node may have multiple facets. The score of these facets are integrated based on weighted linear combination method and then propagated to the node's neighbours. Therefore, first we start from the results of different facets which places our method in the category of late fusion. However, this is not the only factor affecting the final score of a node. After top ranked node activation, we propagate the score of activated nodes into the graph. The final score is a function of both individual facet results in addition to the effect of link weights on propagating to the neighbours.

In this thesis, our multimodal graph approach can seamlessly integrate the result of multiple modalities. We make a form of late facet fusion by combining different facet scores and giving one score to the parent information object. In practice, we have no more a *modality* but a set of facets/features defining that modality. Starting from different facet results, we integrate results of different facet indexings, then propagate the energy to their neighbourhood. After a predefined number of steps, we rank the result based on their final energy. In the implementation of Astera, facet fusion is implicitly calculated by matrix multiplication.

Bipartite Graph Before moving on to the next section, on graph traversal, it should be pointed out that in any instantiation of the Astera model with only β relation between modalities, we end up with a bipartite graph. For instance, in the ImageCLEF 2011 Wikipedia collection with documents and images as part of a document, we observe that the modelled collection is a bipartite graph combined of images on one side and documents on the other side. There is no relation between images or between documents. In this case, energy flows totally from one side to the other.

One solution to avoid a bipartite graph, is to add the self-transitivity (st) value, in which a percentage of the energy of a node returns to itself. Another option that we investigate is to utilize other types of links, such as semantic and similarity between the information objects. This prevents a bipartite graph by construction.

3.1.2 Hybrid Search

Our hybrid ranking method consists of two steps: 1) First, an initial search with Lucene[1] for the text, and Lire[2] for the image modality elements of a query is performed. We obtain a set of activation nodes from these results. The computed scores, in both modalities, are normalized per query between (0,1) based on the min-max method [JNR05b]. 2) Second, using the initial result set of data objects (with normalized scores) as seeds, we activate the graph from N starting points per query facets. In each starting point, parallel multimodal search is conducted based on graph traversal method.

---

[1] https://Lucene.apache.org/

[2] http://www.lire-project.net/

Graph Traversal Method One of the main challenges in graph modelled data, is how to "intelligently" traverse the graph and exploit the associations between the data objects. Two highly used methods in retrieving information on structured data are Spreading Activation and Markov Chain Random Walks.

As described in detail in Chapter 2, we found that these two methods are identical—if Spreading Activation complies with the stochastic property in its weight definition. We use Spreading Activation in first part of our experiments (Chapter 4 and Chapter 5 Section 5.1). One reason is our customized definition of weighting on different relation types, which does not necessarily satisfy the stochastic property. Another reason is to apply different constraints such as distance constraint or path constraint in the traversal.

In Chapter 5, we investigate the score distribution in the graph in very large number of steps and stationary distribution. We cannot reach this state by using Spreading Activation method in our defined model. Therefore, we create a normalized weighting on the graph edges to satisfy the stochastic property and use Random Walks traversal method.

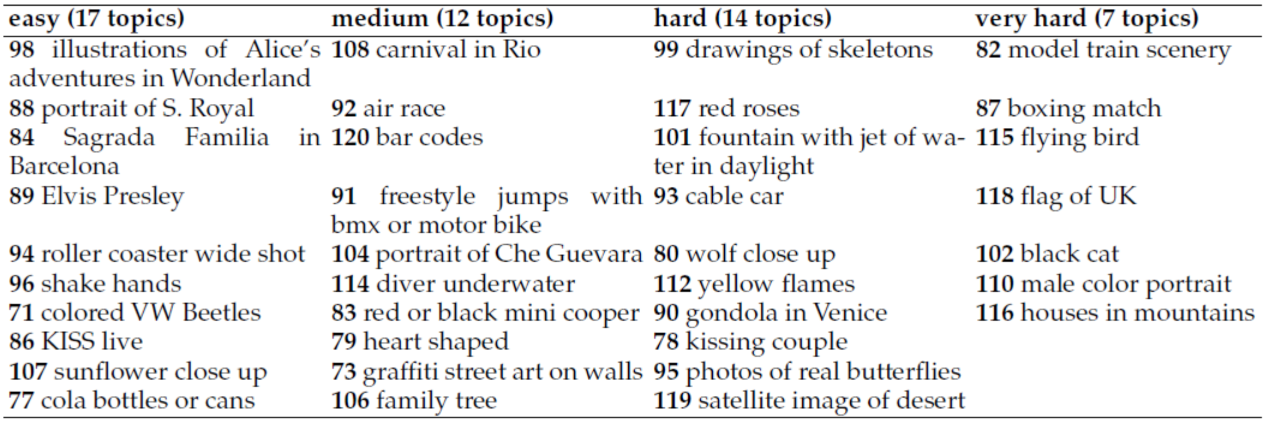
3.2 Evaluation Design

In this section we instantiate our model and describe the evaluation methodology we employ to evaluate retrieval effectiveness. It is necessary to use a standard dataset to evaluate the effectiveness of an information retrieval model. This dataset is mainly composed of three parts: the document collection, a set of test query topics and a corresponding set of relevance judgements to each query. This section is organized as follows: we first explain the benchmark data collection in subsection 3.2.1. We then discuss how we use different links in our model to enrich the dataset in subsection 3.2.2. In subsection 3.2.3 we describe the score normalization method we use in our experiments. As we use different modalities in our model, it is necessary to apply normalization on the search result of different modalities. Finally we explain the strategies we choose in our implementation due to hardware restrictions.

3.2.1 Data Collection

The ImageCLEF 2011 Wikipedia dataset is based on Wikipedia pages and their associated images. It comes with a set of 50 topics used for evaluation. The topics of this collection are classified based on the AP (Average Precition) values per topic averaged over all runs as follows [TPK11]:

easy: $MAP > 0.3$
medium: $0.2 < MAP <= 0.3$
hard: $0.1 < MAP <= 0.2$
very hard: $MAP < 0.1$



easy (17 topics)	**medium (12 topics)**	**hard (14 topics)**	**very hard (7 topics)**
98 illustrations of Alice's adventures in Wonderland	**108** carnival in Rio	**99** drawings of skeletons	**82** model train scenery
88 portrait of S. Royal	**92** air race	**117** red roses	**87** boxing match
84 Sagrada Familia in Barcelona	**120** bar codes	**101** fountain with jet of water in daylight	**115** flying bird
89 Elvis Presley	**91** freestyle jumps with bmx or motor bike	**93** cable car	**118** flag of UK
94 roller coaster wide shot	**104** portrait of Che Guevara	**80** wolf close up	**102** black cat
96 shake hands	**114** diver underwater	**112** yellow flames	**110** male color portrait
71 colored VW Beetles	**83** red or black mini cooper	**90** gondola in Venice	**116** houses in mountains
86 KISS live	**79** heart shaped	**78** kissing couple	
107 sunflower close up	**73** graffiti street art on walls	**95** photos of real butterflies	
77 cola bottles or cans	**106** family tree	**119** satellite image of desert	

Figure 3.2: Topics categories based on their difficulty [TPK11]

The topics in each class are shown in Table 3.2. The topics are divided into four categories of easy (17 topics), medium (10 topics), hard (16 topics) and very hard (7 topics). A large number of topics in easy and medium topics refer to a named entity, and are easily retrieved with textual approaches. On the other hand the hard and very hard topics are highly semantic. They cover types of queries that are supposedly better solved by textual, visual or multimodal retrieval.

The ImageCLEF 2011 Wikipedia collection is multimodal and therefore an appropriate choice for testing the rich and diverse set of relations in our model. The goal of this test collection is to retrieve images. Each image has one metadata file that provides information about name, location, one or more associated parent documents in up to three languages (English, German, and French), and textual image annotations (i.e. caption, description, and comment). The collection consists of 125,828 documents and 237,434 images. We parse the image metadata and create nodes for all parent documents, images, and corresponding facets. We enrich the collection by adding various link types, which we describe in the following section.

3.2.2 Adding Links

In the ImageCLEF 2011 Wikipedia collection we have images and documents. In addition to the part-of (β) relation between an image and its parent document, we add the three other: facet (δ), similarity (γ) and semantic (α) links to the collection.

Facet Links The facet (δ) relation between each information object and its corresponding facets are added to the graph. In order to perform the search, we extract facets from both images and documents. We use textual facets for documents. We utilize default configuration of a Lucene indexer, for three document textual facets of TF.IDF, BM25, and Language Model (LM) with Bayes Smoothing.

For images, we use both visual and textual facets. For visual facets, we use the four image features provided by the collection, as follows:

- CEDD: The Color and Edge Directivity Descriptor combines color and texture information of an image in a histogram [CB08]. It is represented in a 144 dimensional space.
- CIME: A border/interior classification algorithm which classifies pixels into interior or border and then builds a 64 bins histogram for each pixel type [SNF02], resulting in a 64 dimensional feature space.
- SURF: The Speeded-up Robust Features is a descriptor and detector, used for finding discrete image point correspondence where the image is transformed into coordinates. It is principally based on SIFT feature with small differences. The three main steps in SURF algorithm are: first, detecting interest points, second, representing the neighbourhood of each interest point by a feature vector, and third, matching the feature vectors between different images. SURF can be used for tasks such as object recognition, locating and recognizing objects, people, or faces, making 3D scenes, tracking objects, and extract points of interest [BETVG08]. It is represented in a 5000 dimensional space.
- TLEP: Cheng and Chen [CC03] suggest to segment the image into different regions, and collect useful features from each region to classify. They propose a descriptor which adopts two texture descriptors of color histogram and local-edge pattern (LEP) histogram [CC03]. The LEP histogram originates from LBP texture descriptor [OP99], which describes the spatial structure of a local texture. It considers the 3x3 neighbourhood of a pixel. For each pixel, we have 64 bin RGB (4x4x4), resulting in a 576 dimension descriptor.

For textual facets of the images, we consider the meta-data XML files of each image. These files include textual fields (caption, comment and description) of images. Using Lucene we index them as separate fields, and search based on a multi-field indexing. We use TF.IDF, BM25 and LM facets for each of these fields.

Similarity Links Similarity links (γ) connect the same facet type of two information objects. Theoretically we can add a similarity link based on a specific facet between an information object and all other information objects in the collection. For example, it can be between color histogram facets of all images, or between textual TF.IDF facets of documents. These connections create a highly connected graph. However, there would be a lot of weak similarity links in such a graph. In order to filter weak links, we set a limitation for adding a similarity link. For current experiments, we take top 10 similar neighbours of documents/images based on their textual/visual facets. The weight of these links is based on the similarity value between the same facets of an information object and its neighbour.

We perform similarity computations separately for English (EN), German (DE) and French (FR) fields as textual facets of each image. For example, we get the FR comment, caption and description of an image, create a document of these fields, and find similar images based on these fields. An image may have 30 similar neighbours in the case of having textual fields in all languages. We perform the same scenario separately for documents. For each document, we find top 10 similar ones in each language. In total we added 3,535,437 similarity links in the graph.

Semantic Links We use Linked Open Data to add semantic links. We connect the ImageCLEF 2011 Wikipedia collection to DBpedia through the equivalent pages in DBpedia for each wiki page in the collection. We map the correspondent version of DBpedia for Wikipedia 2011. The ImageCLEF2011 Wikipedia collection uses the ImageCLEF 2010 Wikipedia collection, which is based on the September 2009 Wikipedia dumps. Therefore, we used DBpedia version 3.4 which is based on Wiki dump September 2009. Each triple in DBpedia RDF is in the form of <source, predicate, target>. We only add a link from DBpedia if both source and target documents are in the collection. By adding all DBpedia links, a more connected, large scale graph is obtained.

We observed from DBpedia that there are representations of different concepts in one language, as well as representations of the same concepts in different languages. We refer to the links between the concepts in one language as intra-lingual semantic links, and between the same concepts in different languages as inter-lingual semantic links.

For adding intra-lingual links, we add semantic connections between English documents. These links are normal DBpedia properties like `http://DBpedia.org/property/type` or `http://DBpedia.org/property/title`. This way, after visiting a document, we follow its neighbours that may be images or other documents that are connected through semantic links. For instance, a document named *Battle of Leyte Gulf* contains 6 images as neighbours. After adding semantic links, this document connects to 13 other documents in the collection (e.g. *Pacific War* and *World War II*). In total, 55,544 links are added, which is considerable considering that the total number of documents in the collection is 125,828. These links are valuable in the sense that they provide a more connected graph of information objects.

For inter-lingual links, we consider the statistics Tonon and colleagues [TDCM12] provided on different link types. They rank all properties by their observed likelihood of leading to relevant entities. Their results demonstrate that links like `<DBpedia:wikilink>` are too general links that broaden to non-related objects. However, links like *sameAs* are more promising. It leads to better precision as they refer to the same or similar real world entity [TDCM12].

In order to add the `sameAs` link, we need the interlingual link information from DBpedia. This information is not available for the working dump version 3.4. This information is added from version 3.8 of DBpedia. We used the inter-lingual link information from this version. For each English document, we found the corresponding triple in French

Table 3.1: Top 7 most frequent semantic links after adding DBpeida links to ImageCLEF 2011 Wikipedia collection

link type	*frequency*
FR-DE sameAs	36,295
EN-DE sameAs	34,530
EN-FR sameAs	29,828
http://dbpedia.org/property/birth	5556
http://dbpedia.org/property/birthPlace	5551
http://dbpedia.org/property/death	3695
http://dbpedia.org/property/deathPlace	3672
http://dbpedia.org/property/location	2019
http://dbpedia.org/property/instrument	1302

and German. This triple is added via `sameAs` link to the collection. We added 29,828 available links for EN-FR documents, 34,530 links for EN-DE documents, and 36,295 between FR and DE documents.

3.2.3 Normalization

As we work with textual facets of both documents and images for similarity links, we have two different collections of documents (Wikipedia pages) and images metadata. An information object may receive activation score/energy through each of its links. For instance, an image may receive energy from its parent documents in English and German as well as from its similar image neighbours. In order to combine the received energy, we normalize the similarity values in each collection. There are different methods of modelling score distribution in information retrieval, like normal-exponential models, gamma distribution or z-score [AR11]. The distribution result is not necessarily between 0 and 1. The range and distribution of scores varies across different models. As a result, they cannot be compared between different engines [Rob07]. One main purpose of score distribution methods is to provide a score normalization to map the score into a probability of relevance. Cooper et al. [CGD92] argue that the systems *should* provide users an estimate of probability of relevance through logistic regression techniques. Robertson and Bovey [RB82] were the first ones who had the idea of using logistic regression. Given some training data, the role of logistic regression is to calibrate scores in a system. Mainly it is performed based on a view of the ranking result of the system [RB82]. To map this distribution to probability of relevancy, we use the regression function by Nottelman and Fuhr [NF03]. They compare logistic functions with linear function for different retrieval methods. They show that logistic function provides better estimation of relevance probability (Equation 3.3).

$$f_{log}(x) = \frac{exp(b_0 + b_1.x)}{1 + exp(b_0 + b_1.x)} \tag{3.3}$$

They calibrate their logistic function (via two parameters b_0 and b_1) based on the collection and the retrieval method.

In our case, we use a heuristic method to find the b_0 and b_1 parameters (because of lack of training data for images and documents in the collection). We consider the top 10 results of a specific facet for each query topic. The reason is to have a range of possible similarity values. Other values for the number of top results might have been chosen. We leave those for later experiments. We create a set of these top results. We define two conditions to find these parameters. As the first condition, we take the median (*med*) of this result set, and set the probability of the median to 0.5. This way, half of the scores have probability less than 0.5 and half more than that.

$$\begin{aligned}
&f_{log}(med) = 0.5 \\
&f_{log}(med) = \frac{exp(b_0 + b_1.med)}{1 + exp(b_0 + b_1.med)} = 0.5 \\
&b_0 = -b_1 \cdot med
\end{aligned} \tag{3.4}$$

As the second condition, we assume that $f(min) = 0.05$ where min is the minimum value in this collection. The target is to map probability 0 for minimum score: $f(min) = 0$, but the logistic function is of course always strictly greater than 0. So instead, in the absence of anything else, we consider $f(min) = 0.05$. This arbitrary value is in this case no better justified than in the case of the p-value threshold in statistical testing.

$$\begin{aligned}
&f_{log}(min) = 0.05 \\
&f_{log}(min) = \frac{exp(b_0 + b_1.med)}{1 + exp(b_0 + b_1.med)} = 0.05 \\
&b_0 + b_1(min) = -2.94
\end{aligned} \tag{3.5}$$

Based on these conditions, we find the fixed value of $b_0 = -b1 \cdot med$, and $b_1 = -2.94/(min - med)$. For instance, for image with Id 1537, we search for top 10 similar images based on the metadata TF.IDF facet. The similarity score results is the set: [0.15, 0.14, 0.13, 0.12, 0.11, 0.08, 0.075, 0.06, 0.045, 0.041]. The median of this list is 0.095. Based on Equations 3.4 and 3.5, parameters b_1 and b_0 take the values 30.42 and 2.88. Having these two parameters, we calculate the normalized value for all scores which is: [0.84,0.79, 0.74, 0.68, 0.61, 0.30, 0.35, 0.25, 0.17, 0.16]. These are the weights we assign to each neighbour as normalized similarity value. We perform this calculations for each image/document in the collection at index creation time.

3.2.4 Implementation Strategies

Subgraph Traversal The ImageCLEF 2011 Wikipedia collection contains 363,262 information objects (images and documents without considering the facet nodes). A matrix of this size, requires about 983GB RAM for matrix multiplication. In order to make the computation feasible for large collections, our strategy in Astera is to only consider nodes that are potentially reachable after N steps to generate a smaller adjacency matrix. However, this set of reachable nodes depends on the query. Therefore, for different query topics and different number of steps, we work with different subgraphs.

Starting from top ranked nodes for a query topic, we visit next round neighbours in each step. After visiting all neighbours to a specific step in the graph, we create the adjacency matrix W. The cell values of the adjacency matrix are the edge weights between visited nodes. We compute the steps in the graph by matrix multiplication, based on the following formula [SLR14]

$$a^{(t)} = a^{(0)} \cdot W^t \tag{3.6}$$

where the $a^{(0)}$ vector is composed of top ranked nodes of query facets (as non-zero elements), and visited neighbours through traversal (as zero elements). The final vector, $a^{(t)}$, provides the final activation value of all nodes. We select the images and calculate precision and recall based on their scores.

In this version of our model, we use the distance constraint to stop the traversal [Cre97].

Weighting Strategy Each information object (e.g. image, document or any other type of information object) may have many facets. By construction, they can receive at maximum, a score of 1 from facets. The general formula for the combined scoring is: $obj\_score = \sum_{i=0}^{z} w_i.f_i$ where $\sum_{i=0}^{z} w_i = 1$. Variable z is the number of facets, and w_i is the weight of facet f_i. For this collection—having textual and visual facets—we found experimentally the $(0.7 \cdot TextualFacet + 0.3 \cdot VisualFacet)$ as the best combination (Table 5.2). We have three textual facets and four visual facets. Based on this weighting, the combination of scores in a node, with all of these facets is as follows:

$$(0.7 \cdot \sum_{i=1}^{n} \widetilde{f_i} \cdot w_i) + (0.3 \cdot \sum_{j=1}^{m} \widetilde{f_j} \cdot w_j) \tag{3.7}$$

where n is the number of textual facets, $\widetilde{f_i}$ is one of the textual facets with weight w_i, m is the number of visual facets, and $\widetilde{f_j}$ is one of the visual facets with weight w_j.

For images, we have visual facets (CEDD, SURF, CIME and TLEP), and metadata information as textual facet. Mapped to the score formula, it is $(0.7 \cdot TextualFacet + 0.3 \cdot VisualFacet)$. For document objects, if for instance, we have only textual facet TF.IDF, we give (1.0· TF.IDF) as weighting. In this version of Astera, we find different facet weights experimentally. These weights should be learned for a specific query or domain, e.g. via relevance feedback.

CHAPTER 4

Reachability Analysis

The use of different facets in Astera has support in the principles of IR, most notably in the theory of poly-representation [LIK06]. The aim is to leverage different cognitive and functional representations of information objects to improve IR results. There is currently little understanding of how using different representations of the same objects (what we call here facets) affect the reachability of relevant items. Moreover, in a graph model of information objects, different types of links play an important role in reachability to relevant information objects.

In this chapter, we investigate whether the graph structure is conductive to better reachability (RQ2) [SLBR15]. We start with an exploratory data analysis over the collection in Section 4.1. In the second section, we show the effectiveness of our model leveraging different facets. We then elaborate whether using semantic/similarity links helps reaching relevant information objects in Section 4.3. We show the effect of different facets and links in reachability for different categories of topics in Section 4.4. Finally as some facets show the same recall behaviour, in the last section we address the research question whether similarity in the recall behaviour is due to the fact of visiting the same parts of the graph (Section 4.5).

To keep track of which links are present in the graph in different experiments, the following section titles contain the link types (α=semantic, β= containment/part-of, γ=similarity). The like type δ which is used for facets is always included as it connects an information object to its facets.

4.1 Relevant Objects Distribution

As a first experiment, we want to obtain an understanding of the collection in particular of how the relevant images are distributed in the graph. The relation types used in this section are β and δ links. Through these investigations, we want to see how far, and up to how much recall we are able to reach in the graph using only the basic connections (part-of and facet). There are 50 topics in ImageCLEF 2011 Wikipedia collection. We conduct the traversal up to 40 steps for each of these topics. In each step, we check if we visit new relevant images for a specific topic.

Figure 4.1 shows the distribution of relevant nodes in the collection as we start from three facets (document TF.IDF, CEDD and image tags TF.IDF). We will come back to the discussion of the choice of these facets in the following section. The x axis is the number of steps we traverse the graph, and y axis is the Id of the query topics we have. In each step we count the number of new relevant images we visit. Existence of a shape (circle/square/star/triangle) indicates visiting at least a true positive. The size of a shape is the ratio of number of relevant nodes seen in this step, normalized by the total number of relevant nodes for a query topic Id.

We observe a large number of large shapes in the first steps. It indicates visiting more relevant images initiating from different facet results. We observe that easy and medium topics (circles and diamonds) are mostly reachable at the very beginning steps. For hard and very hard topics (squares and triangles) there are more distributed relevant nodes as we continue the traversal. They show almost constant increase as we traverse the graph. This observation demonstrates that we reach a larger fraction of relevant results for hard and very hard topics after 15 steps.

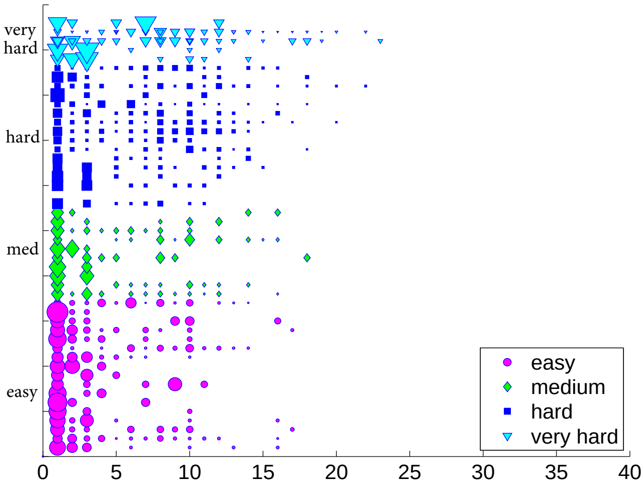


Figure 4.1: Overall recall aggregated per category over steps

4.2 Reachability from Different Facets

Here we investigate the behaviour of potential recall when using not only the basic facets as we have just observed, but rather a variety of combinations of text and image facets, including β and δ links in the graph.

4.2.1 Textual Facets Only

We start with document textual facets (TF.IDF$_D$, BM25$_D$ and LM$_D$). We take top 10 results based on each of these facets for different queries, and traverse the graph starting from them. We calculate recall in each step, i.e. we calculate the recall based on all the nodes visited up to this step. We observe very similar behaviour of different document textual facets in Figure 4.2a, with a high shift in recall in the first step.

We perform the same analysis for images with TF.IDF$_I$, BM25$_I$ and LM$_I$ tag facets. Figure 4.2b shows the recall for image textual facets. We observe that they show also the same behaviour, but LM$_I$ and BM25$_I$ show slightly higher recall than TF.IDF$_I$.

4.2.2 Visual Facets Only

In Section 4.1, the experiment was only based on the CEDD facet, as visual facet. In this section, we add three other image features provided by the collection: SURF, CIME and TLEP. We show their individual effects on recall in Figure 4.2c.

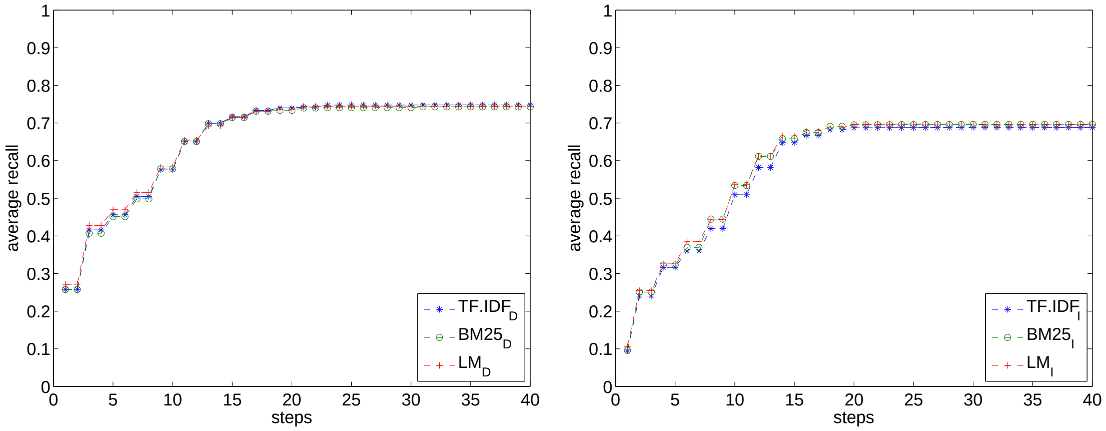
All four image facets demonstrate approximately the same behaviour. We observe that there is a shift in steps between the 2nd and 6th with slightly more increase for CEDD. Also, there is an increase between the 8th and 12th steps for SURF. However, pure visual facets do not start from promising points in the graph.

4.2.3 Different Facet Combinations

According to the analysis so far, each category of facets (textual, visual) shows approximately the same recall behaviour. A question arises whether each of these facets are qualitatively similar too. (i.e. do they visit the same images?) To answer this question, we investigate different information objects visited through each of these facets in Section 4.5.

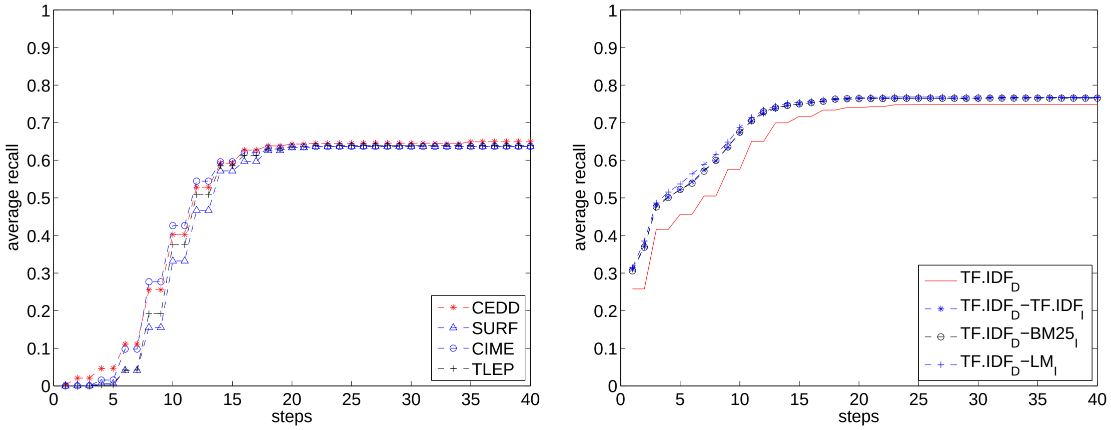
To continue with facet combination, we performed a thorough test on different combinations of document textual facets with image textual facets. For instance, some combinations are TF.IDF$_D$-TF.IDF$_I$, TF.IDF$_D$-BM25$_I$, or BM25$_D$-LM$_I$. The result of different combinations showed similar behaviour. For instance, Figure 4.2d shows the effect of combining TF.IDF$_D$ with image textual facets. In all combinations the recall shifts upwards about 19% in the first steps. The total recall is increased by 4% as well. Therefore, for further investigations on the effect of facet combinations, we define the variable R_D for document textual facets, and the variable R_{IT} for image textual facets. For upcoming particular experiments, we choose $R_D =$TF.IDF$_D$ and $R_{IT} =$TF.IDF$_I$. The value of these variables can be any of TF.IDF, BM25 or LM facets.

Figure 4.2e shows the combination of TF.IDF$_D$ with image visual facets. We observe that the final recall with TF.IDF$_D$-CEDD combination is slightly increased. There is no explicit increase with other visual facets. Comparing Figures 4.2d and 4.2e, we find that with the combination of textual facets, we reach more relevant objects earlier.



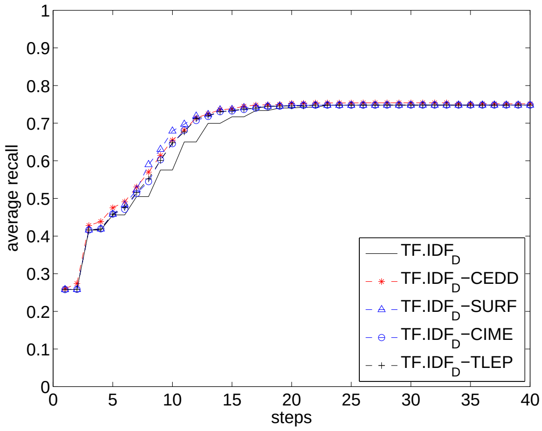
(a) Recall with different **document textual facets**

(b) Recall with different **image textual facets**



(c) Recall with different **image visual facets**

(d) Recall comparison of combining TF.IDF$_D$ with image textual facets



(e) Recall comparison of combining TF.IDF$_D$ with image visual facets

Figure 4.2: Recall with different document or image facets

Similarly for image visual facets, we define the R_{IV} variable that could be any of the visual facets. According to Figure 4.2c, CEDD shows slightly better recall at first steps. CEDD has been shown as the best feature to extract purely visual results on ImageCLEF 2011 Wikipedia Collection [BVO+11]. Therefore, we choose CEDD as the value of R_{IV} in the following experiments.

We perform the experiment for different combinations of facets: R_D, R_D-R_{IV}, R_D-R_{IT}, and R_D-R_{IV}-R_{IT} (Figure 4.3). The upper set of lines shows the average recall obtained through each combination. The lower set of lines indicates the percentage of the graph visited through each combination of these facets. To clarify whether recall increase is the effect of simply visiting more nodes or traversing through a meaningful path in the graph, we added the lower diagram to Figure 4.3.

We observe the changes in the recall values using each combination. The recall results of R_D-R_{IV} are nearer to those of R_D, while R_D-R_{IT} obtains higher recall values, closer to those obtained when using all features (R_D-R_{IV}-R_{IT}). Looking at the lower diagram, we observe that the combination with R_{IV} (CEDD) outpaces the one with R_{IT} (TF.IDF$_I$) in visiting more nodes in the graph. The opposite occurs in the recall diagram - the R_{IT} facet outpaces the R_{IV} facet. This observation confirms the result we had with recall increase through adding textual facet compared to visual facet. Clearly, in this case, we visited fewer nodes, but obtained higher recall.

Another observation is the R_D-R_{IV}-R_{IT} combination, which show higher recall value than the other two combinations. This highlights the importance of different, diverse representations of the query to reach more relevant objects.

Continuing this experiment, we combine separately all document and image textual facets. We compare the result with the diverse facet combination of R_D-R_{IV}-R_{IT}. Figure 4.4 demonstrates that combination of document textual facets outpaces the image textual mixture. However, the R_D-R_{IV}-R_{IT} combination, again outperforms the other two. It confirms the effect of leveraging diverse sets of facets towards better reachability.

4.3 Reachability through Different Links

Up to here, we used various facets in our recall investigation, leveraging only δ and β links. Now, we investigate the role of adding further links to improve reachability. All experiments are started based on the set of top 20 results of standard search on R_D-R_{IV}-R_{IT} facets with TF.IDF$_D$, CEDD and TF.IDF$_I$ values. The reason to not include all facets is to begin with a smaller number of starting points (meanwhile containing at least one facet from each category) to demonstrate the effect of different links.

The baseline recall is what we had in Figure 4.2a. We used the R_D facet to define starting points. As a baseline for comparison, it is visible in Figure 4.5 as "base-graph". We observe that this curve reaches a plateau after step 17, with 76% of recall.

One claim could be that going through Lucene results, we could reach the same recall. In order to compare with Lucene results, we take the same number of results as the

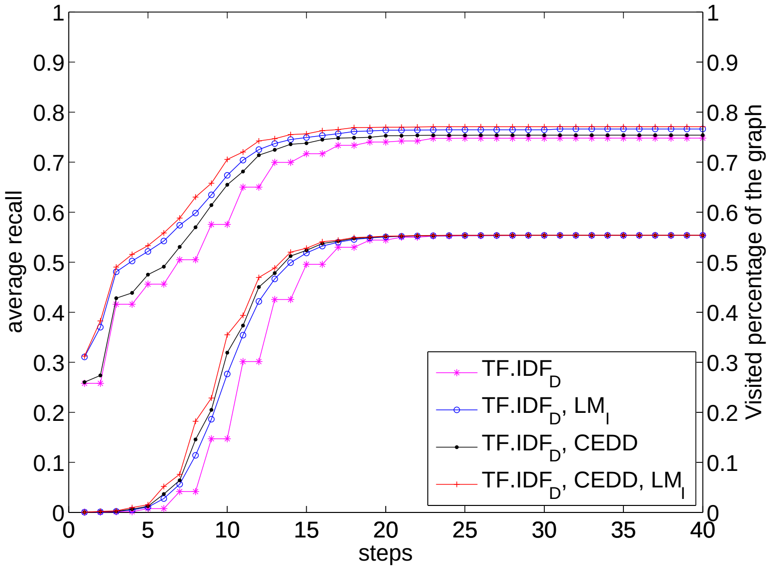


Figure 4.3: Average recall obtained compared to the percentage of the graph seen through different fact combinations of R_D = TF.IDF$_D$, R_{IV} = CEDD, and R_{IT} =TF.IDF$_I$

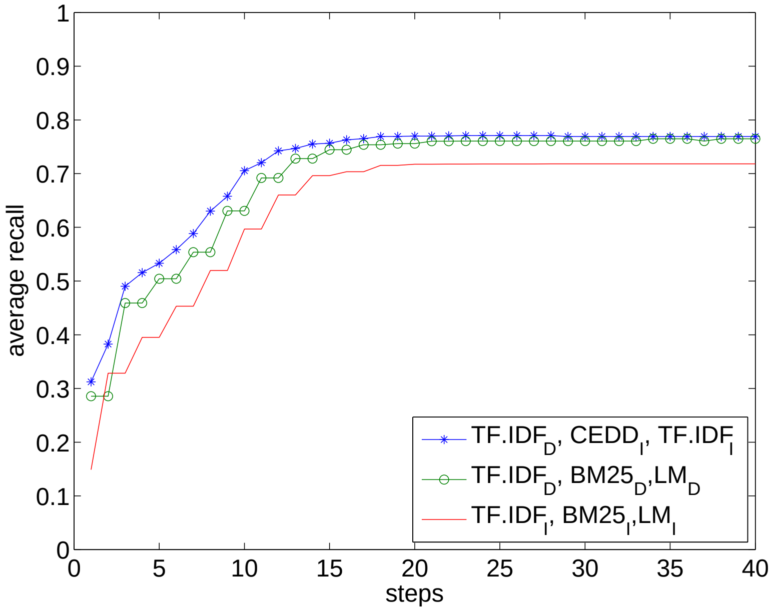


Figure 4.4: Recall comparison of combination of document textual facets with image textual facets with R_D-R_{IV}-R_{IT} (TF.IDF$_D$, CEDD, TF.IDF$_I$) combination

number of new nodes seen in each step in the base-graph. For instance, if we visit 430 new nodes in the third step, we add the subsequent 430 results from the ranked list. We calculated the recall for each of them. We observe that our model holding only β relations (base-graph curve) is more promising than Lucene results which has a plateau at 0.67 recall (Figure 4.5). After the first 3 steps, it outpaces the Lucene curve to the plateau of 0.76.

4.3.1 Semantic Links - β, δ, α

We add intra- and inter-lingual semantic links, and perform the same experiments again (Figure 4.5). We observe a steeper slope in the first steps, up to 0.84 recall in 15th step. We obtain 10% increase in the overall recall. This demonstrates that adding semantic links leads to higher reachability, to a larger number of relevant nodes.

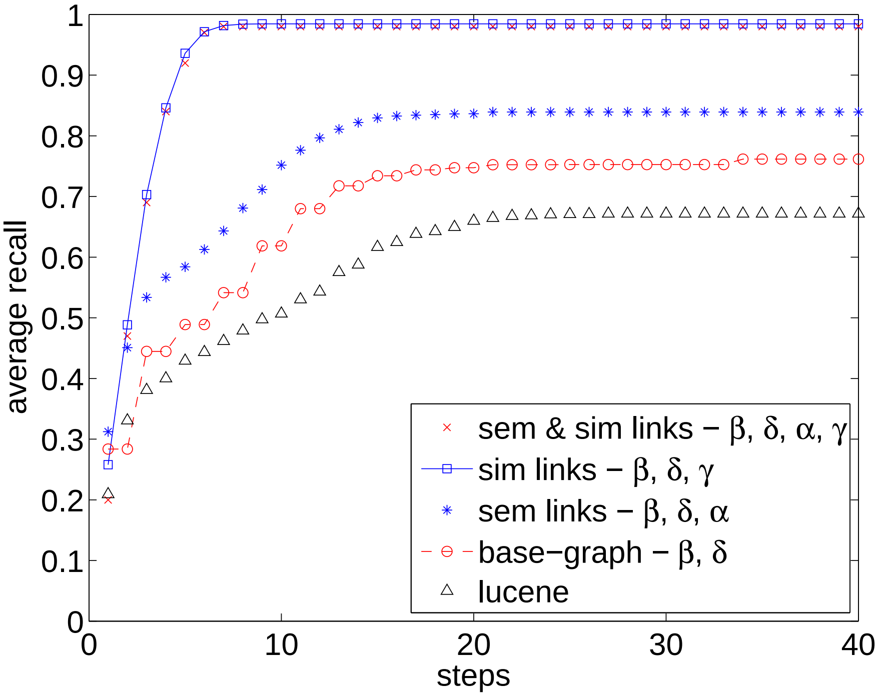
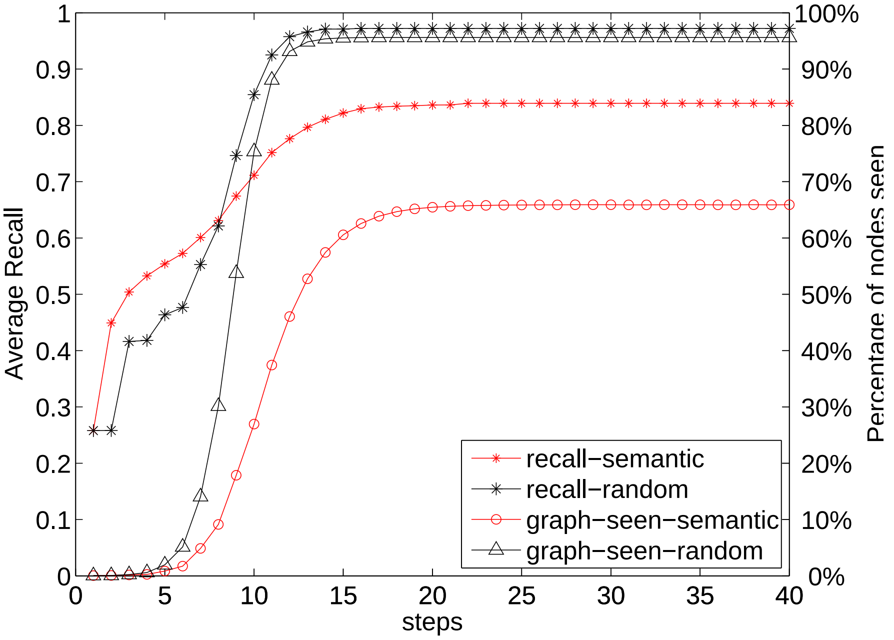
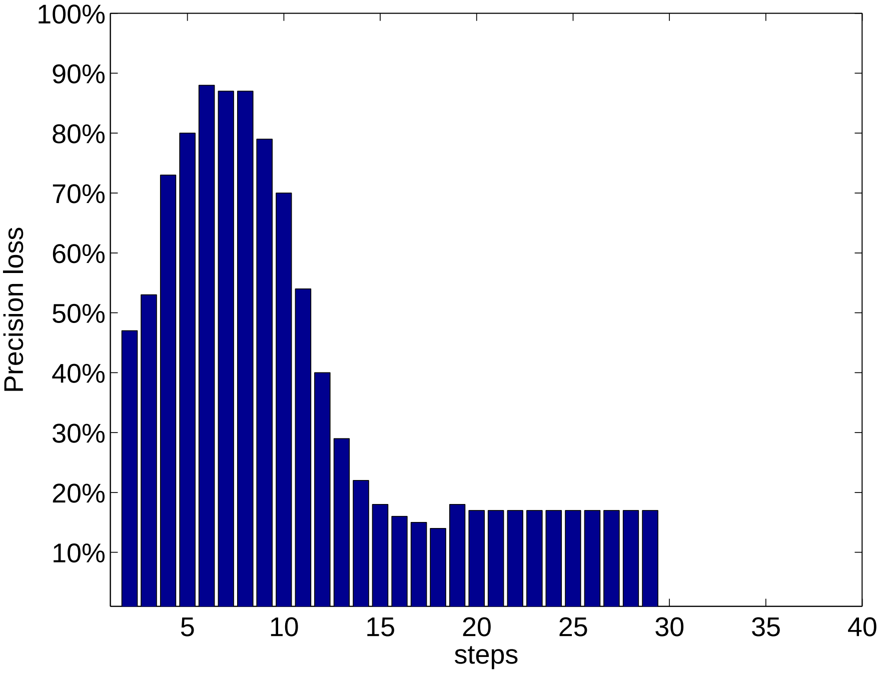


Figure 4.5: Average recall of all topics after 40 steps with different links

After observing high increase in recall by adding semantic links, the question may arise whether *any* additional set of links would lead to such an improvement. Do we reach the same recall value by adding the same number of *random* semantic links? To examine, we compare the result of adding semantic links with adding the same number of pseudo-random semantic links. Based on Table 3.1, we add the same number of random links between EN-DE, EN-FR and DE-FR documents. We run the same scenario on the collection with the new added links. Figure 4.6a compares the recall value of adding semantic links with the one from adding random links. We observe that after 12 steps,



(a) Recall and graph percentage seen, from adding semantic links compared to added randomly semantic links



(b) Precision loss in each step: comparing precision value with random semantic links to the value with real semantic links. For instance, we observe 88% precision loss in the 6th step.

Figure 4.6: Comparing the effect of adding semantic links and random semantic links on recall and precision loss in the graph

with random semantic links, we reach a very high recall of 0.97. We visit 95% of the graph nodes. While we reach 0.84 recall with real semantic links by visiting only 66% of the graph size. We want to investigate this in more detail. We compare the amount of precision we lose, with the increase rate of the number of visited nodes. Figure 4.6b shows that how much adding *random* semantic links deteriorates the precision. In each step, we calculate the precision loss based on:

$$\text{precLoss} = 1 - \left(\frac{\text{precision with random semantic links}}{\text{precision with real semantic links}}\right). \tag{4.1}$$

For instance, in step 6, we lose up to 88% precision compared to using real semantic links. The result of this step shows that the addition of semantic links contributes to higher recall with meaningful links. The precision loss from any of these additional links is lower than if we had added the same number of links randomly.

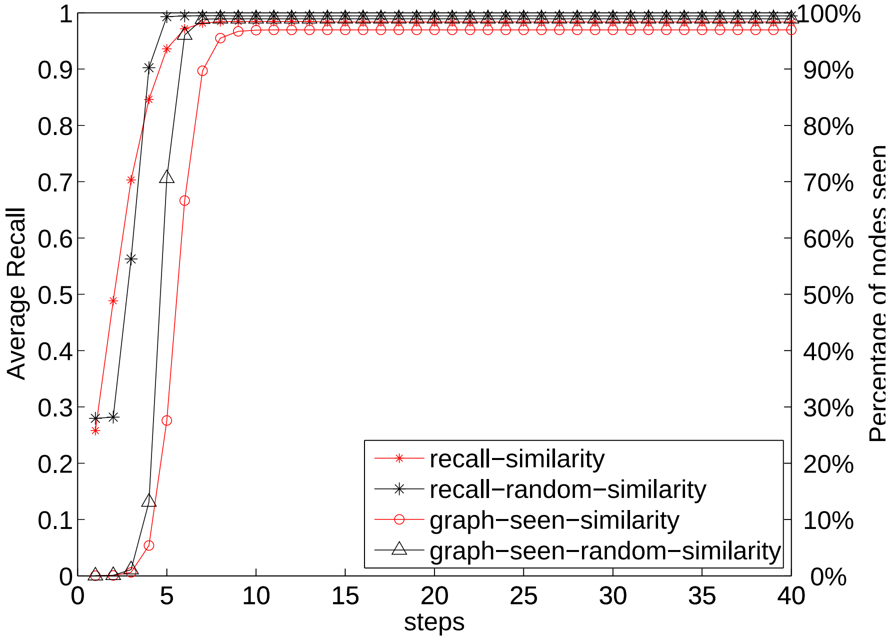
4.3.2 Similarity Links - β, δ, γ

After adding similarity links the graph becomes highly connected. We reach a recall of 98% in the 6th step (Figure 4.7a). The total graph has 363,252 nodes, of which we see 352,208 nodes. This way we reach approximately all nodes in the graph.

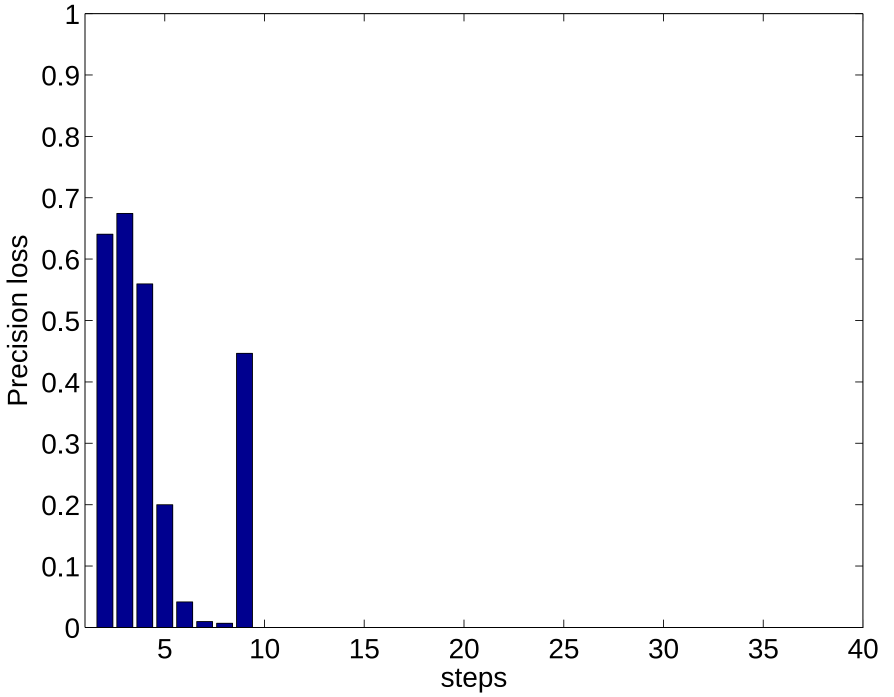
We reached very high recall (98%) in only 6 steps by adding similarity links. The questions is whether we reach the same amount of recall by adding the same number of random links. We perform the same scenario of checking the contribution of added links for recall as in Section 4.3.1 for similarity links. The number of 3,535,437 real similarity links are added in the graph between documents and images. We add the same number of links randomly to the graph. We observe in Figure 4.7a that we reach 98% of recall after 7 steps for real similarity links. The same happens with random similarity links in the 5th step (we reach 99% of recall). After that, there are few relevant images remaining to be reached. However, we should consider the expense of this high recall based on the number of visited nodes in each step. We compare the percentage of the graph seen in the other two diagrams in Figure 4.7a for real and random similarity links respectively. We observe a higher slope in the triangle diagram, which reaches the 99% visit of the graph in the 5th step. Whereas, given real similarity links (circle diagram), we reach the same recall with lower number of visited nodes.

To show the expense of visiting this large number of nodes, we calculate the precision in each step in the graph with real/random similarity links. Figure 4.7b shows the precision loss in each step based on Equation (4.1) as before.

We observe that we lose more than 50% in the first four steps that we reach to 98% of recall. In steps 5, 6, 7, and 8 the precision in both graphs is approximately equal. In step 9 (the last step of visiting relevant nodes in the graph with random similarity links), there are not many relevant nodes remaining to be seen in the graph with random similarity links (Figure 4.7a). This is because of the high connectivity in the graph with random similarity links. However, with similarity links, there are still relevant nodes to be seen



(a) Recall and graph percentage seen, from adding similairty links compared to added randomly similarity links



(b) Precision loss in each step: comparing precision value with random similarity links to the value with real similarity links. For instance, we observe 67% precision loss in the 2nd step.

Figure 4.7: Comparing the effect of adding similarity links and random similarity links on recall and precision loss in the graph

in this step, leading to larger value in the denominator of Equation 4.1. We observe that although we visit a larger number of relevant nods in pursuing random similarity links, the number of visited nodes is too high (e.g. 215588 nodes in the 4th step) that we observe in Figure 4.7b higher percentage for loss of precision in the first steps.

This observation demonstrates the effective contribution of adding similarity links to the collection to increase the reachability.

After elaborating the effect of different links on reachability, we want to zoom in this time, and investigate the effect of different facet combinations on recall behaviour of various topic categories.

4.4 Recall Analysis of Different Topic Categories

In this section, we show the effect of different facet combination on recall behaviour of various topic categories (easy, medium, hard, very hard). We partitioned the results based on the topic categorization done by Tsikrika et al. [TPK11]. The three representative facets R_D (TF.IDF$_D$), R_{IV} (CEDD), and R_{IT} (TF.IDF$_I$) are used in these experiments.

4.4.1 Base Graph - β, δ

We start the experiments on the graph containing δ (facet) and β (part-of) relations.

Text facet - R_D

In this experiment, we include only R_D results to start the search in the graph. Figure 4.8a shows the average recall for different categories. We observe that easy topics meet 0.86 recall after 20 steps and this value remains constant thereafter. For medium topics is the same after the 21st step with maximum value of 0.73. Hard and very hard topics continue increasing the recall value also until the 21st step and up to the values of 0.66 and 0.71, respectively. An interesting observation is the behaviour of very hard topics after the 3rd step: they outpace hard topics. This demonstrates that as we go farther in the graph we cover a higher percentage of recall for very hard topics rather than hard topics.

Another observation is the increase rate in each category. Easy topics show the increase rate of 138% (from 0.36 to 0.86), where it is 128% for medium topics (from 0.32 to 0.73), 266% for hard topics (from 0.18 to 0.66) and 373% for very hard topics (from 0.15 to 0.71). The values show that easy and medium topics are apparently answerable by direct querying, while it is in the hard and very hard topics that the graph model shows most promise.

Further, we observe that recall is increasing up to 21st step and then goes to a plateau for all categories. Two results are obtained from this observation: first is that by conducting the traversal, we can expect increase in recall in the graph to about 21 steps. Because we are still visiting relevant nodes as we go farther every one or two steps. Second is that after the 21st step we do not visit relevant images any more, and recall is still less than 0.86 even for easy topics. This shows that the graph is not connected. Our log files show that no more new nodes are reached after the 40th step for any of the topics. Therefore, the probability of continuing the traversal and seeing new relevant new nodes is zero (when using only the textual facet in this set-up).

All Facets - R_D, R_{IV}, R_{IT}

In this experiment, we use the R_D, R_{IV}, and R_{IT} results to start the propagation (Figure 4.8b). We observe the effect of multiple facets in the first steps (1st to 10th) with higher recall values. In addition, the recall plateau in each category can be reached earlier with all facets. We have the same values between 5th and 10th steps comparable to the recall value for 10th to 15th with only text facets (R_D). Further, recall has increased to 0.71 for hard topics. The final recall for different categories did not show a significant difference. The reason is the connectivity of the graph that after a number of steps, all reachable relevant nodes are visited.

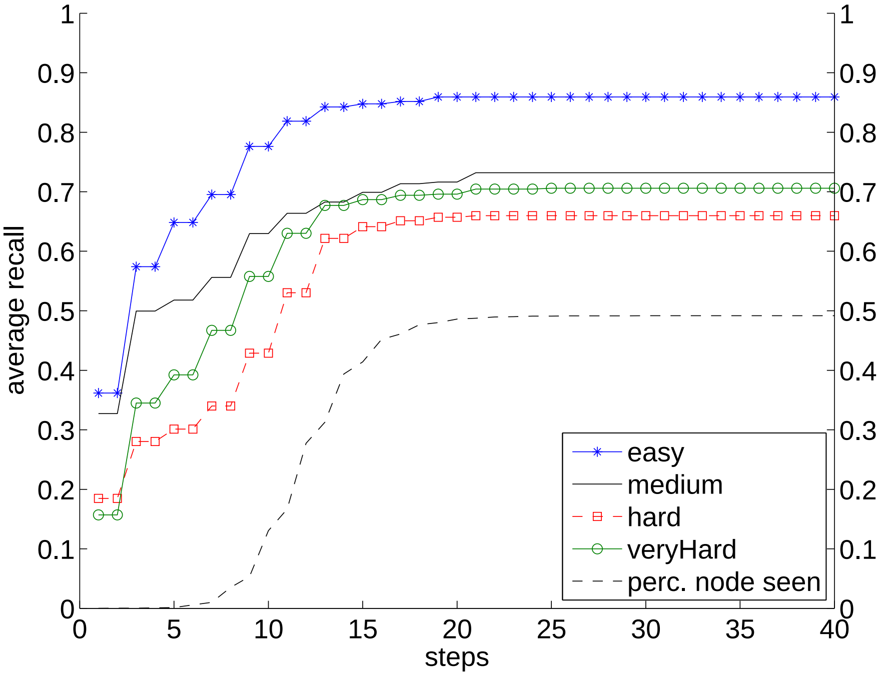
The ImageCLEF 2011 has 363,262 nodes. We included the percentage of the nodes seen in both experiments. The result shows that we visit the majority of relevant nodes by passing less than 10% (about 36000 nodes) of the graph. However, the total number of the nodes seen at plateau is about half of the collection (average of 178,620 nodes). This illustrates that we have access to almost half of the graph. In addition, the convergence of traversal performance at about 20th step for all topic categories is another confirmation that we do not have access to all the graph. To tackle this challenge we add semantic links to the collection.

4.4.2 Semantically Enhanced Collection - β, δ, α

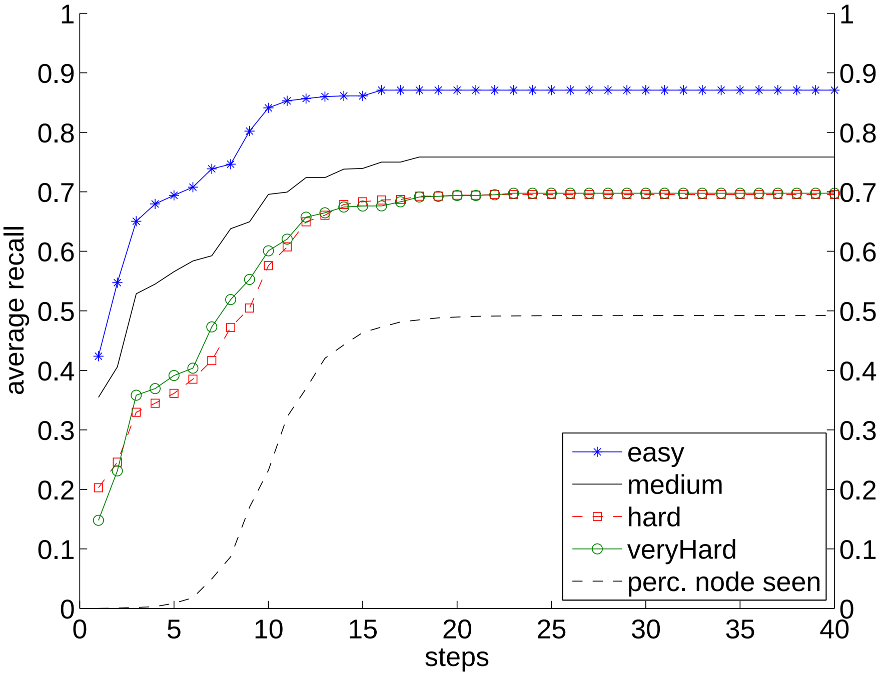
We perform the same experiments for the collection enhanced with semantic links. Again the experiments are based on R_D, R_{IV}, and R_{IT} facets.

Text Facet - R_D

In this experiment, we conduct the test based on R_D facet top results. It is apparent that we obtain a more connected graph and consequently expect higher recall. We show the reachability result starting from the Text facet in Figure 4.9a. We observe that recall in all categories reaches a plateau as it did in the graph version without semantic links, but with a clear shift in recall value for all categories (easy topics from 0.86 to 0.90, medium from 0.73 to 0.81, hard from 0.66 to 0.74, and very hard from 0.71 to 0.77). In this experiment again, hard and very hard topics, with 13% and 8% increase rate compared to the graph without semantic links, outpaced other categories.



(a) Average recall using TF.IDF$_D$ - Base graph (β, δ)



(b) Average recall using TF.IDF$_D$, CEDD, TF.IDF$_I$ - Base graph(β, δ)

Figure 4.8: Average Recall in the base graph on different categories of topics

All Facets - R_D,R_{IV},R_{IT}

By starting from R_D, R_{IV}, and R_{IT} results, we observe approximately the same behaviour in different categories (Figure 4.9b). The two Figures 4.9a and 4.9b, are rather similar. The reason is high connectivity in the graph. It does not show much difference if we start from R_D or R_D-R_{IV}-R_{IT} results. However, there are two obvious differences. First is recall increase in the first steps. Second is the shift in the final recall value of each category. The reason is that we have a highly connected graph, where the starting points for search do not have a major impact. However, starting from different facets has an effect in the initiating steps (1st to 5th step) leading to a steep slope at the beginning.

The same pattern of seen nodes is observed for the traversal in the graph with semantic links. This shows on one side the importance of the starting points in reachability to the relevant information objects. On the other side, we observe the role of added semantic links in reaching higher recall values by visiting the same percentage of 10% of nodes in the graph. Another observation in both Figures 4.8 and 4.9 is that we are visiting the majority part of relevant objects in the first five steps, while visiting about 1% of the graph.

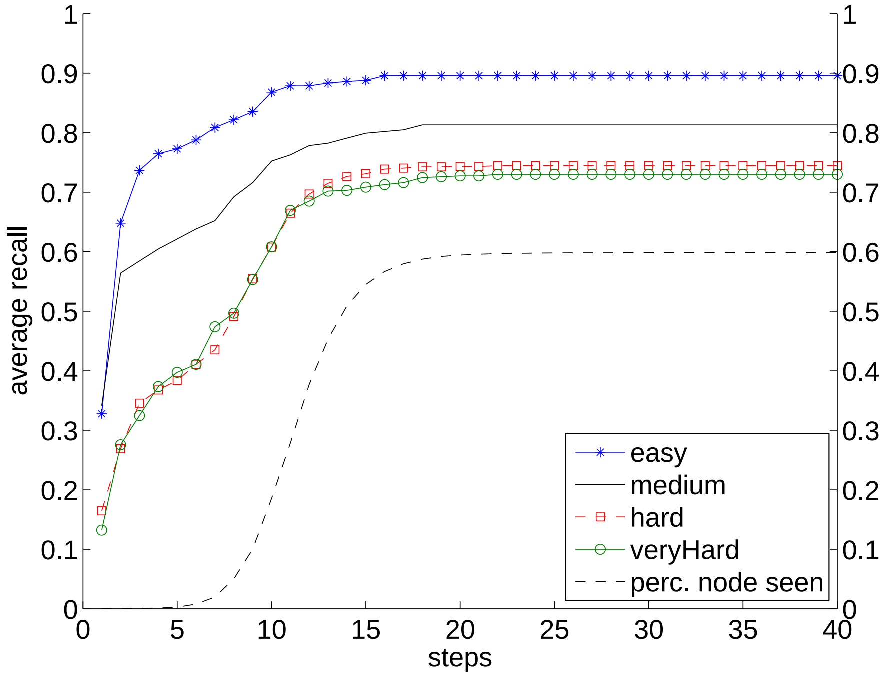
4.4.3 Similarity Links Added - β, δ, γ

In this experiment we add the similarity links (γ) to the base graph. Figures 4.10a and 4.10b show the recall behaviour after adding similarity links for only text facet (R_D) and for all facets. It is clear in both figures that the graph is highly connected – more so than by adding only semantic links. We observe a very high slope at the beginning of each category, and all reach the plateau of 0.98 or 1 after 7 steps. However, it is worth to notice that we visit a large number of nodes in each step (about 30k new nodes), from which is challenging to prioritize the relevant nodes to the top results to improve precision. The detail is described in Section 4.3.2

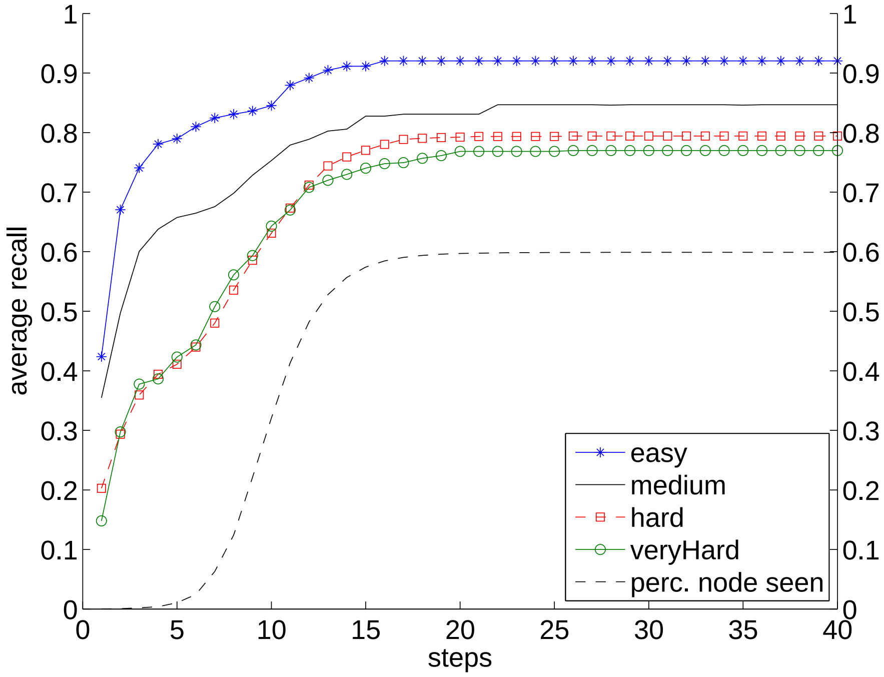
4.4.4 Discussion

From the experiments on reachability in different links and topic categories, we can thus draw the following initial conclusions:

- Adding semantic links increases the potential recall, especially for hard and very hard topic by 13% and 8% whereas easy topics may not need the graph.
- Leveraging multiple facets, we saved at least 5 steps to reach the same potential recall compared to using only one facet.
- Leveraging semantic links shifted highly the recall value already in the first few steps.

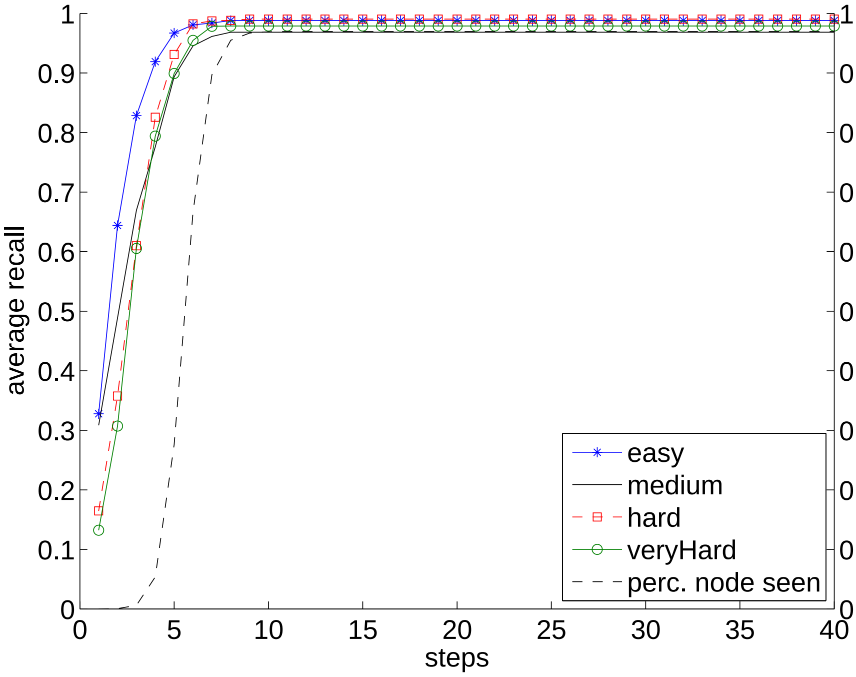


(a) Semantic links added (β, δ, α), average recall using $\mathrm{TF.IDF}_{\mathrm{D}}$

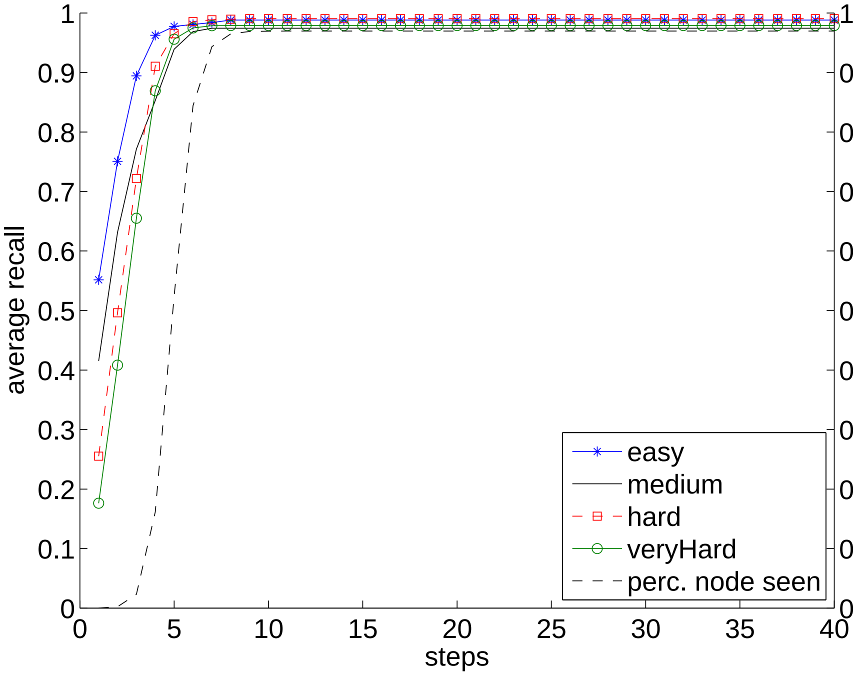


(b) Semantic links added (β, δ, α), average recall using $\mathrm{TF.IDF}_{\mathrm{D}}$, CEDD, $\mathrm{TF.IDF}_{\mathrm{I}}$

Figure 4.9: Average Recall in the graph with semantic links on different categories of topics

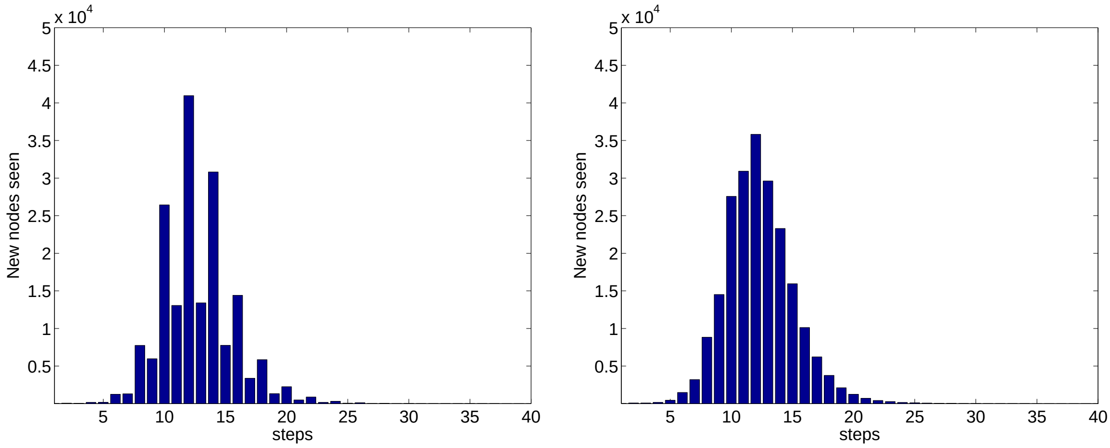


(a) Similarity links added (β, δ, γ), average recall using TF.IDF$_\mathrm{D}$



(b) Similarity links added (β, δ, γ), average recall using TF.IDF$_\mathrm{D}$, CEDD, TF.IDF$_\mathrm{I}$

Figure 4.10: Average Recall in the graph with similarity links on different categories of topics



(a) Number of new nodes seen per step in the collection (β, δ), 178,620 nodes in total

(b) Number of new seen nodes per step, semantic links added (β, δ, α,), 217,395 nodes in total

Figure 4.11: Number of new seen nodes per step in the collection

- We demonstrated the effect of different facets leading to visiting different parts of the collection. This reinforces the importance of the poly-representation idea to identify the relevant objects.

Further, we provide an analysis on the number of nodes visited. Figure 4.11a shows the average number of new nodes seen for all topics in each step. We observe that it starts to increase after the 5th step up to the 25th to the total size of 178,620 nodes. The oscillation of the seen nodes in even steps is because of reaching documents in even steps and images in odd steps. The number of images are more than twice that of the number of documents in the collection.

The same analysis on the collection containing semantic links demonstrates that the number of nodes are mainly increasing in the first steps up to the 25th step again (Figure 4.11b), to the total size of 217,395 nodes. We observe 22% increase in the number of nodes seen compared to the baseline graph. This observation indicates that a lower number of steps is needed to traverse the relevant nodes with semantic links. However, it challenges the precision, as we visit new nodes in the scale of thousands including only few relevant nodes.

4.5 Graph Visit from Different Facets

In Section 4.2, we observed very similar recall behaviour of different visual and textual facets. We want to understand whether with the same recall value, we visit the same relevant nodes as well.

We calculate the percentage of different nodes visited in a step as:

$$d_{f_i} = \frac{|C_{f_i}| - |M|}{|C_{f_i}|} \tag{4.2}$$

where C_{f_i} represents the nodes seen in a step for facet f_i, and $M = \bigcup_{f_j \in F \setminus \{f_i\}} C_{f_i} \cap C_{f_j}$, where F is the set of facets, the set of nodes also seen by other facets. This way, by $|C_{f_i}| - |M|$ we count the nodes only reachable through the facet f_i. The value d_{f_i} is the ratio of nodes reachable only through this facet.

We calculate the same ratio for seen relevant images. In this case, C_{f_i} is all the relevant images seen in a step for facet f_i.

4.5.1 Document Textual Facets

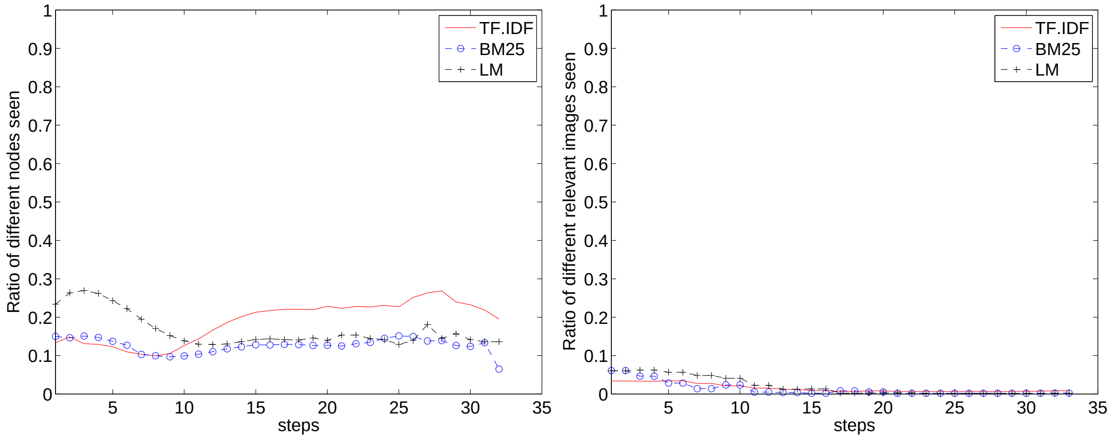
We start with the nodes visited from the result of document textual facets. We perform this experiment with TF.IDF$_D$, BM25$_D$ and LM$_D$ facets. Figure 4.12a shows that in the starting steps (up to 10), TF.IDF$_D$ and BM25$_D$ show the same behaviour. They visit in average 13% different nodes compared to the other two, while LM$_D$ starts from different nodes in the graph. After the 10th step, LM$_D$ and BM25$_D$ show the same behaviour, where TF.IDF$_D$ visits 20% different nodes. However, we observe in Figure 4.12b that all these three facets visit approximately the same relevant images. They show small differences at the beginning, but after 5 steps, there is no difference.

4.5.2 Image Textual Facets

Figure 4.13a shows the ratio of different visited nodes through TF.IDF, BM25 and LM facets of image metadata. We observe that TF.IDF and BM25 start with 20% different points in the graph. Each keep visiting on average 15% different nodes compared to the other two facets. However, LM$_I$ proposes a rather divergent view. It starts with 48% different nodes. It starts with different top results and keeps this different view to the end. Its impact on visiting relevant images is clear in Figure 4.13b. Up to step 15, LM$_I$ visits more different relevant images than the other two facets.

From the results of document and image textual facets (Figures 4.12a, 4.12b, 4.13a, and 4.13b), we observe that the LM facet shows different behaviour compared to TF.IDF and BM25. A probable reason is the structural difference of these facets. Both TF.IDF and BM25 are based on *tf* and *idf* factors, while LM holds a completely different probabilistic view.

Figures 4.12a shows less ratio of visiting different parts of the graph compared to Figure 4.13a. When we start the traversal based on document textual facets, we traverse less divergent parts of the graph, compared to the graph parts visited as we start with image textual facets (Figure 4.13a). The reason is that starting with document textual facets, we touch documents in the first place. In each next step, we see all images inside a document, covering a large number of images in one step. Whereas, starting with image



(a) Ratio of different nodes visited through **document textual** facets

(b) Ratio of different **relevant** images visited through **document textual** facets

Figure 4.12: Ratio of different nodes visited from document textual facets

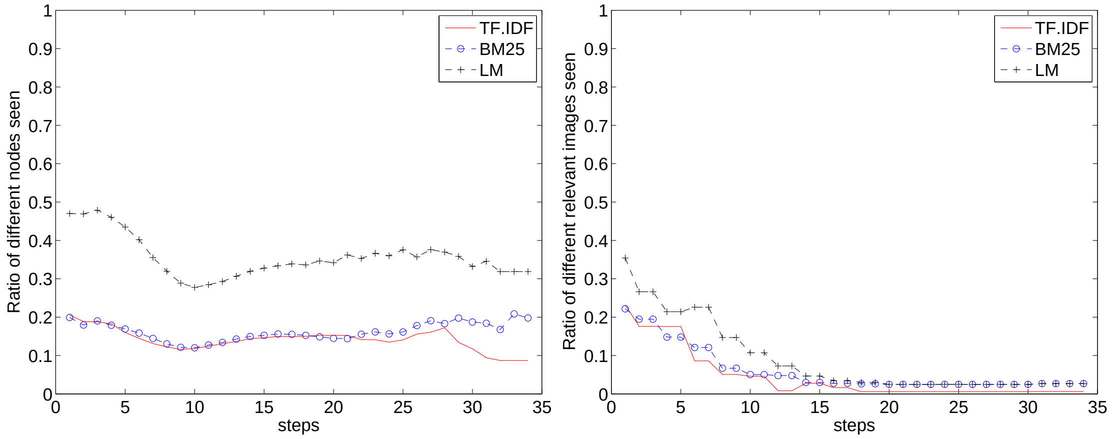
textual facet results, we touch individual images in the first level. These images may be part of different documents which lead to the broader view to the graph. This results in visiting more divergent relevant nodes as well, as we observe large difference in starting steps in Figure 4.12b compared to Figure 4.13b. We observe that, the LM facet not only visits different parts of the graph, but also keeps visiting different relevant images. This observation reinforces the poly-representation principle [LIK06] in using cognitively dissimilar facets.

4.5.3 Image Visual Facets

We calculated the d_{f_i} values (Equation 4.2) in each step for all four image visual facets (Figure 4.14a). We observe that each of these facets starts from totally different parts of the graph ($d_{f_i} = 1$). However, it decreases with a high slope to $d_{f_i} = 0.2$ after 10 steps. Afterwards, they keep visiting different nodes with different ratio (between 20% and 35%). Facets CIME and TLEP show very similar behaviour, while CEDD and SURF are more divergent.

Figure 4.14b shows the ratio of relevant images seen through each of these facets. In the first steps, each facet visits different relevant nodes (correspondent to Figure 4.14a). For instance, in step 7, CEDD has the highest difference with CIME. After step 18, relevant nodes seen through CIME, TLEP and SURF show overlap with seen relevant nodes by CEDD. However, CEDD keeps visiting 2% different relevant nodes.

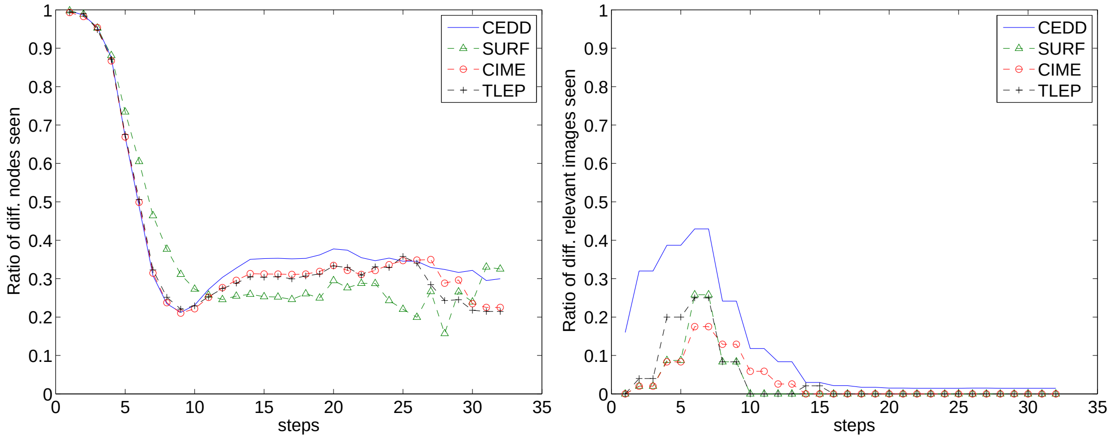
Comparing Figures 4.2c and 4.14a, we observe that although the visual facets demonstrate the same recall value, they visit different relevant nodes at least in the first 15 steps.



(a) Ratio of different nodes visited through **image textual** facets

(b) Ratio of different **relevant** images visited through **image textual** facets

Figure 4.13: Ratio of different nodes visited from image textual facets



(a) Ratio of different nodes visited through **image visual** facets

(b) Ratio of different **relevant** images visited through **image visual** facets

Figure 4.14: Ratio of different nodes visited from image visual facets

4.6 Summary

In this chapter, we designed experiments to evaluate our model in reachability to relevant information objects. The experiments were performed on the ImageCLEF 2011 Wikipedia collection. Astera is able to enrich the modelled collection in two ways: 1) Extracting inherent information of data objects as facets. 2) Adding semantic/similarity links between information objects. We evaluated this model with ImageCLEF2011 Wikipedia test collection. We demonstrated the effect of different facet combination on recall. Further, we observed how connected the collection is. We provided an understanding of reachability to relevant information objects. In addition we found the minimum number of steps that still there is a chance to see relevant information objects.

In the first part of the experiments, we presented a reachability analysis of relevant objects from different facets. We showed the effect of choosing effective diverse facets in reachability, as the combination of document and image textual facets resulted in better recall than document textual and image visual facet. We observed that by leveraging multiple facets, we saved at least 5 steps to reach the same potential recall compared to using only one facet. Further, we showed that not necessarily using larger number of facets leads to higher recall.

In the second part of our experiments, we showed the effect of adding semantic and similarity links in increasing the reachability in the graph. We observed 22% increase rate of the number of nodes seen in the graph, by adding real semantic links, compared to the 10% recall gain. The difference shows that we cannot easily increase recall, when the baseline is already 0.76. This highlights the importance of type and number of the links we choose to add. Further, we showed the contribution of adding similarity links to increase reachability to 98% of relevant information objects. In both cases, we compared the result in the graph with the same number of random links. The rate of precision loss in the graph with random links showed the contribution of real semantic and similarity links to the graph.

In the third part of our analysis, we looked at the effect of different facets and links on different topic categories (easy, medium, hard, very hard). We observed that easy and medium topics are mainly reachable in initial steps. However, the relations in the graph and semantic links helped significantly in reachability to hard and very hard relevant information objects. We observed the recall increase of 266% and 373% for hard and very hard topics, respectively.

In the last part of the reachability analysis, we demonstrated the effect of different facets leading to visiting different parts of the collection. This reinforces the importance of the poly-representation idea to identify relevant information objects. We found that although the image visual facets show the same recall behaviour, they visit totally different relevant images at the beginning steps (up to 10). This encourages to leverage more than one facet to start traversing the graph from, as they help to see relevant nodes earlier. In addition, the analysis on textual facets showed that the Language Model (LM) facet has more divergent view than BM25 and TF.IDF facets.

We conclude that, not necessarily adding more facets leads to a better recall value. We should consider that adding more facets increases the size of the visited graph and increases computational complexity. However, leveraging facets with different aspects of the information object speeds up the time to reach the recall plateau.

From comparing the graph enriched with real semantic links to the one with random semantic links, we learned that adding new links should be chosen meaningfully. We reach higher recall by adding random semantic links, but lost up to 88% precision in some steps compared to using real semantic links.

After accomplishing experiments on recall analysis, we are motivated to consider the role of different facets and link types on precision value. In next chapter, we show experiments on score distribution and precision value in small and large number of steps in the graph.

CHAPTER 5

Precision Analysis

In Chapter 4, we investigated the reachability of relevant information objects and recall. In this chapter, we answer RQ3 which discusses whether the top ranked results based on the graph traversal identify the relevant information objects. We start with the effect of different facets and links on performance in the first section. This analysis is concentrated in the initial steps based on Spreading Activation as traversal method. However, by using this method, we do not reach a stationary distribution state in the graph, as the weighting in the graph does not match the stochastic property. In Chapter 2, we described the two well-known traversal methods in IR: Spreading Activation and Random Walks. We associate Spreading Activation method to not normalized weighting in the graph. We want to leverage the score distribution in the stationary distribution state. Therefore, in the second part of this chapter, we define a normalized weighting on the links in the graph and effectively, this is now Random Walks (as shown in Section 2.1.3). In this part, we also explain the graph behaviour when considering a larger number of steps.

To keep track of which links are present in the graph in different experiments, the section titles contain the link types (α=semantic, β= containment/part-of, γ=similarity). The link type δ for facets is assumed to be always included, as it connects an information object to its facets.

5.1 Precision Analysis with Different Facets

For further investigations on the effect of facet combinations, as we defined in Chapter 4 Section 4.2, we use the variable R_D for document textual facets, and R_{IT} for image textual facets. The value of these variables can be any of TF.IDF, BM25 or LM facets. We performed thorough analysis on the effect of each of these facets in visiting relevant information objects in Chapter 4. We showed that they show a similar recall behaviour, as we used different facet values for R_D and R_{IT} variables. Therefore we take a representative from each category for upcoming particular experiments. As in Chapter 4, we choose

Table 5.1: Results for baseline standard search

doc_w	img_w	*p@10*	*r@10*	*p@20*	*r@20*
1	0	0.311	0.105	0.247	0.129
0.7	0.3	0.34	0.109	0.281	0.133

R_D =TF.IDF$_D$ and R_{IT} =TF.IDF$_I$ for experiments in this section. Further, we define the R_{IV} variable that could be any of the image visual facets such as SURF, TLEP, CEDD and CIME. The CEDD facet has shown better performance result in purely visual results on ImageCLEF 2011 Wikipedia Collection [BVO$^+$11] than the other three visual facets. Again as defined in Chapter 4 Section 4.2, we choose CEDD as the value of R_{IV} in the following experiments.

5.1.1 Baseline

We start the experiments with a baseline test which uses only standard Lucene without leveraging the graph model. We get the top 20 result of standard Lucene search based on the R_D facet. For each ranked document result, we extract its associated images and rank them based on the score of the parent document. This score is constant for the set of images in a document. Secondarily, we rank the images alphabetically based on their names. The result is shown in the first row of Table 5.1. For instance the precision at cut-off 10 (P@10) is 0.311. The variable doc_w is the weight given to the document-based similarity results and img_w is the weight given to image-based similarity results. We show the precision and recall values in the cut-off 10 (p@10 and r@10). In the first part of this experiment, result images are ranked only based on their parent score, we have $doc_w = 1$ and $img_w = 0$.

In the next step, we additionally refine the baseline by computing the similarity between each of the query images (q_{img_i}) and each of the result list images res_{img}. We compute the maximum similarity value (SV) as reference. It is shown in the following formula:

$$SV_{q_{imgs},res_{img}} = max(Sim(q_{img_i}, res_{img})), 1 \leq i \leq 5 \tag{5.1}$$

where the index i ranges between 1 and 5 indicating the 5 image examples of each query.

Now each image result has two scores: the text score and the image similarity score. We apply a range of different weightings for linear combination of text and image score. As shown in Table 5.2 the best result is obtained by assigning weight 0.7 to the text score and weight 0.3 to the image score.

Results obtained from image-only searches (using LIRE) had very low recall and are not presented here. There are other experiments that confirm the poor result of using only visual features [AOB$^+$11, ZB11]. As stated in the review of the ImageCLEF 2011 Wikipedia collection [TPK11], none of the top runs are based on pure visual features.

Table 5.2: Different combinations of weightings on document and image similarity results

doc_w	img_w	*p@10*
1.0	0.0	0.311
0.9	0.1	0.329
0.8	0.2	0.337
0.7	**0.3**	**0.34**
0.6	0.4	0.331
0.5	0.5	0.322
0.4	0.6	0.310
0.3	0.7	0.305
0.2	0.8	0.294
0.1	0.9	0.282

The result based on $R_{IT} = TF.IDF_I$ as the result of image metadata indexing showed lower performance (with p@10 of 0.26 and p@20 of 0.20). Therefore, we choose the R_D based results, refined with $R_{IV} = CEDD$ as the baseline for our experiments.

5.1.2 Different Facets Combination - β, δ

From this experiment, we start the graph-based search for precision on the collection. We design several experiments based on R_D, R_{IV}, and R_{IT} facets, as well as various self-transitivity values. Link types β and δ are used to focus on the effect of different facets in the graph.

In these experiments, we set positive values for the self-transitivity parameter. With no self-transitivity value, we end up with zero value for precision in the even steps. The reason is that the graph is bipartite and we visit no images in the even steps (Figure 5.1). Another reason is that we would only count the images visited in a particular step in the precision, as all of the energy is shifted in each step to the next. Therefore, we set a non-zero value for the *st* parameter. This way, all visited images up to this step participate in the final ranking. Any positive value for *st* would have this effect. We set $st = 0.9$ to slow down the pass of energy to the neighbours.

To choose the number of steps to traverse in the graph, we designed experiments to take many steps in the graph. The results for different facets showed the decreasing trend of precision, which ended up with 0.03 value at step 24 (Table 5.3). On the other hand, as the weighting in the graph does not satisfy the stochastic property, we generate energy in each step, such that in the 87th step we reached the Double max value (more detail is described in Appendix A). We begin by analysing the result of starting steps up to the 9th step in this section, and focus on the effect of facet combinations in the initial steps in this section.

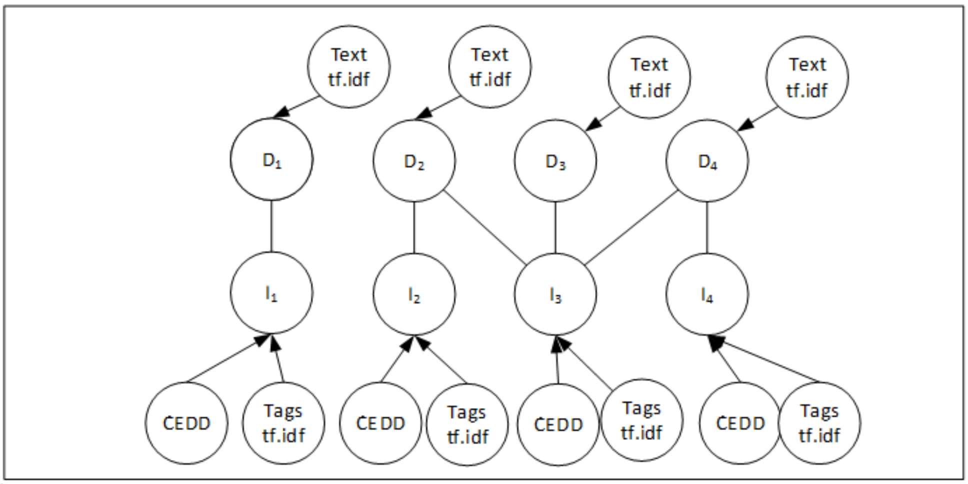


Figure 5.1: Bipartite graph, in odd steps we visit only images and in even step only documents

Document Textual Facet - R_D

In this experiment, we use the TF.IDF$_D$ facet results as initiating points in the graph. We do not include any visual or textual facet of the images. From Table 5.4, we observe 0.34 for P@10 by using the graph structured data. Compared to the baseline result (Table 5.1, row 1), we obtain 8.5% increase in the precision value.

As the activation is propagated further up to 9 steps, we observe a decrease in the precision. We obtain almost the same precision in even steps compared to their prior odd steps. The reason is that self-transitivity holds the value of 0.9, and we count all images visited up to the current state in each step. In even steps, we visit only documents, and we have the same (number of) images as in odd steps.

Document Textual and Image Visual Facets - R_D, R_{IV}

In this experiment, in addition to the TF.IDF$_D$ results, the top images (based on CEDD similarity) are added to the list of activation points. The activation vector is therefore a combination of indexed documents and images.

First, we consider no self-transitivity value. This way, in each step, only the images seen in this step are included in the final ranked list. We receive worse results compared to the R_D result, especially in the even steps (Figure 5.2). The reason is that starting from top image nodes, we visit more images in even steps and they are mostly non relevant. In these steps we have the results from both document textual facets and CEDD with no preference. This result confirms that pure image-based retrieval is not informative.

In order to remove the high influence of top image results in the propagation, we weight the document and image score results in the initial vector. To compare with the best

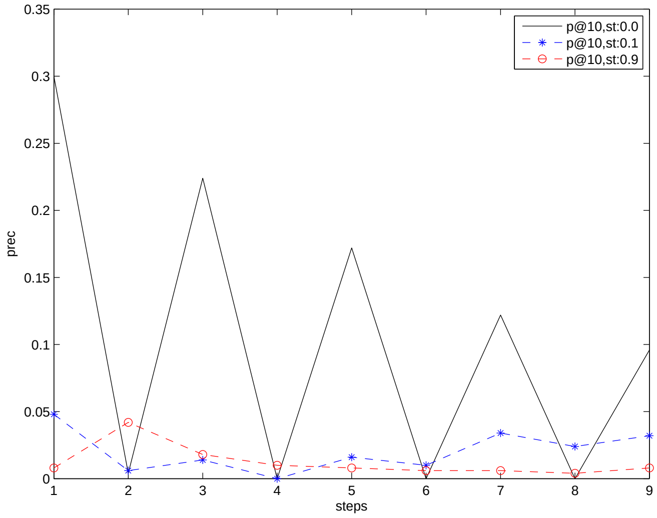


Figure 5.2: Prec@10 using TF.IDF_{D}, CEDD, not weighted

result we had in the baseline search, we perform the same weighting here (with 0.7 weight to the R_D and 0.3 to R_{IV}). We observe in Table 5.5 that the weighted result is much better in even steps than Figure 5.2. The reason is that the scores of visual facet (R_{IV}) are reduced to match their perceived importance for retrieval. The precision results in Table 5.5 are approximately the same as the results in Table 5.4. However, we obtain better recall in the first four steps. Going further in the graph, we see a higher number of images, which decrease the precision of the system.

Document and Image Textual Facets - R_D, R_{IT}

In this experiment, we perform the search based on TF.IDF_D and TF.IDF_I facet results. We see an increase of 6% in the first step (Table 5.6). Also in the third step, we have better precision. This shows that we visit relevant documents, not only after one step, but also after three steps. We observe that by using R_{IT}, we have better recall in the first three steps as well. In these experiments the *st* value is 0.9, which means that energy is predominantly retained in the nodes. Therefore, all image nodes visited in the traversal are included in our calculations.

All Facets - R_D, R_{IV}, R_{IT}

We included all three result sets of TF.IDF_D, CEDD, and TF.IDF_I in this experiment (Table 5.7). Precision in the first step is the same as combination of R_D and R_{IT} results. This means that CEDD top ranked nodes did not help. In the second step, starting from document top results (R_D), we visit documents again (no new images) which do not affect the result of this stage. As we use only δ and β links, the graph is bipartite. However, starting from images top results (R_{IV} and R_{IT}), we visit new images in the

Table 5.3: The results of 24 steps in the graph starting from TF.IDF$_D$. We observe the descending order of precision and recall.

steps	*p@10*	*r@10*	*p@20*	*r@20*
1	0.34	0.136	0.25	0.161
2	0.34	0.136	0.25	0.161
3	0.286	0.114	0.208	0.158
4	0.28	0.112	0.206	0.149
5	0.252	0.104	0.188	0.144
6	0.244	0.104	0.18	0.138
7	0.218	0.095	0.176	0.138
8	0.194	0.081	0.158	0.124
9	0.19	0.08	0.148	0.115
10	0.146	0.126	0.065	0.1
11	0.134	0.122	0.053	0.088
12	0.13	0.11	0.049	0.081
13	0.116	0.102	0.045	0.077
14	0.116	0.1	0.045	0.076
15	0.104	0.094	0.042	0.073
16	0.104	0.093	0.042	0.072
17	0.1	0.087	0.038	0.063
18	0.088	0.081	0.032	0.055
19	0.088	0.069	0.026	0.048
20	0.076	0.063	0.027	0.05
21	0.066	0.051	0.024	0.042
22	0.048	0.04	0.016	0.033
23	0.048	0.039	0.016	0.03
24	0.032	0.031	0.01	0.024

second step. Precision increase to 0.372 demonstrates that leveraging diverse facets (R_{IV} and R_{IT}) lead to visiting more relevant images. We gain 9% increase compared to using only TF.IDF$_D$ facet. The precision gain is statistically significant based on t-test with p value of 0.05; the same holds for the previous experiment.

5.2 Precision Analysis by Adding Semantic and Similarity Links

In this section, we test the performance with the collection enriched with semantic (α) and similarity links (γ). We investigate the role of these links on improving the precision.

Table 5.4: Results with R_D, st:0.9, links: β, δ

steps	*p@10*	*r@10*	*p@20*	*r@20*
1	**0.34**	0.136	0.25	0.161
2	0.34	0.136	0.25	0.161
3	0.286	0.114	0.208	0.158
4	0.28	0.112	0.206	0.149
5	0.252	0.104	0.188	0.144
6	0.244	0.104	0.18	0.138
7	0.218	0.095	0.176	0.138
8	0.194	0.081	0.158	0.124
9	0.19	0.08	0.148	0.115

Table 5.5: Results with R_D, R_{IV}, st:0.9, links: β, δ

steps	*p@10*	*r@10*	*p@20*	*r@20*
1	0.34	0.135	0.25	**0.188**
2	0.338	0.133	0.257	**0.193**
3	0.29	0.115	0.207	**0.163**
4	0.266	0.101	0.175	**0.131**
5	0.234	0.093	0.165	0.122
6	0.136	0.052	0.098	0.068
7	0.11	0.039	0.077	0.055
8	0.094	0.032	0.065	0.042
9	0.084	0.056	0.062	0.038

Table 5.6: Results with R_D, R_{IT}, st:0.9, links: β, δ

steps	*p@10*	*r@10*	*p@20*	*r@20*
1	**0.362**	0.139	0.265	**0.189**
2	**0.346**	0.132	0.259	**0.175**
3	**0.308**	0.119	0.224	**0.165**
4	0.24	0.088	0.187	0.135
5	0.212	0.081	0.164	0.118
6	0.158	0.06	0.133	0.097
7	0.164	0.06	0.128	0.091
8	0.144	0.56	0.113	0.085
9	0.084	0.027	0.062	0.038

Table 5.7: Results with R_D, R_{IV}, R_{IT}, st:0.9, links: β, δ

steps	*p@10*	*r@10*	*p@20*	*r@20*
1	**0.358**	0.14	0.27	**0.195**
2	**0.372**	0.137	0.272	**0.193**
3	**0.308**	0.12	0.22	**0.166**
4	0.25	0.093	0.186	0.127
5	0.218	0.083	0.162	0.113
6	0.114	0.037	0.088	0.06
7	0.114	0.037	0.085	0.056
8	0.138	0.056	0.113	0.085
9	0.068	0.055	0.107	0.083

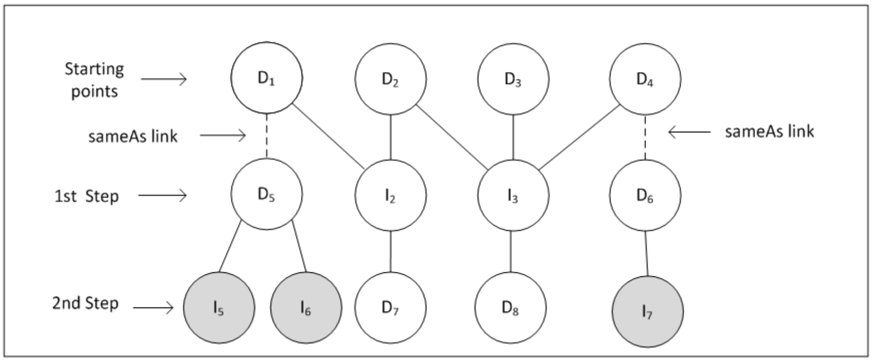


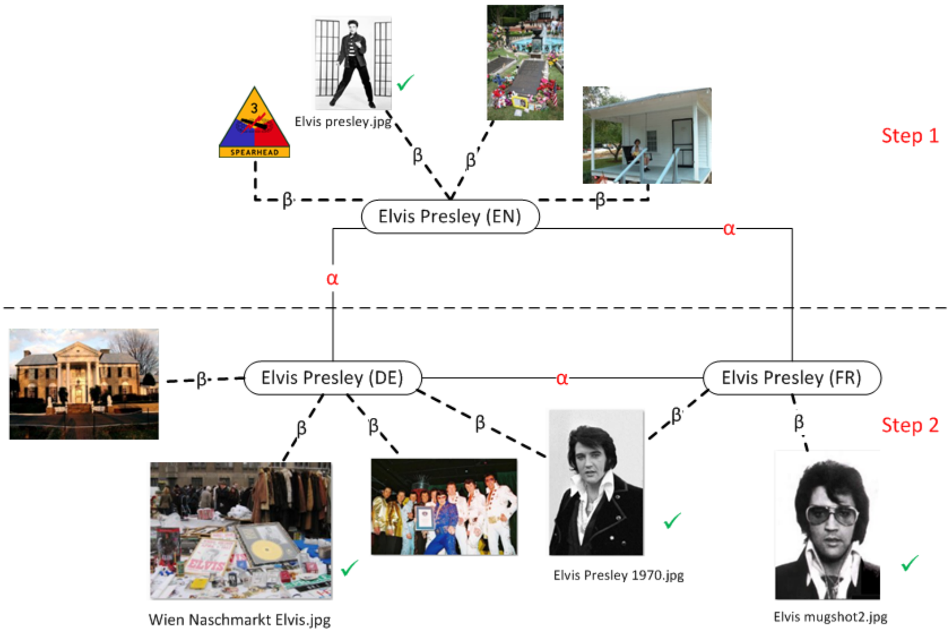
Figure 5.3: In the second step with $st = 0$, reachable images are the ones through `sameAs` links

5.2.1 Semantic links - α, β, δ

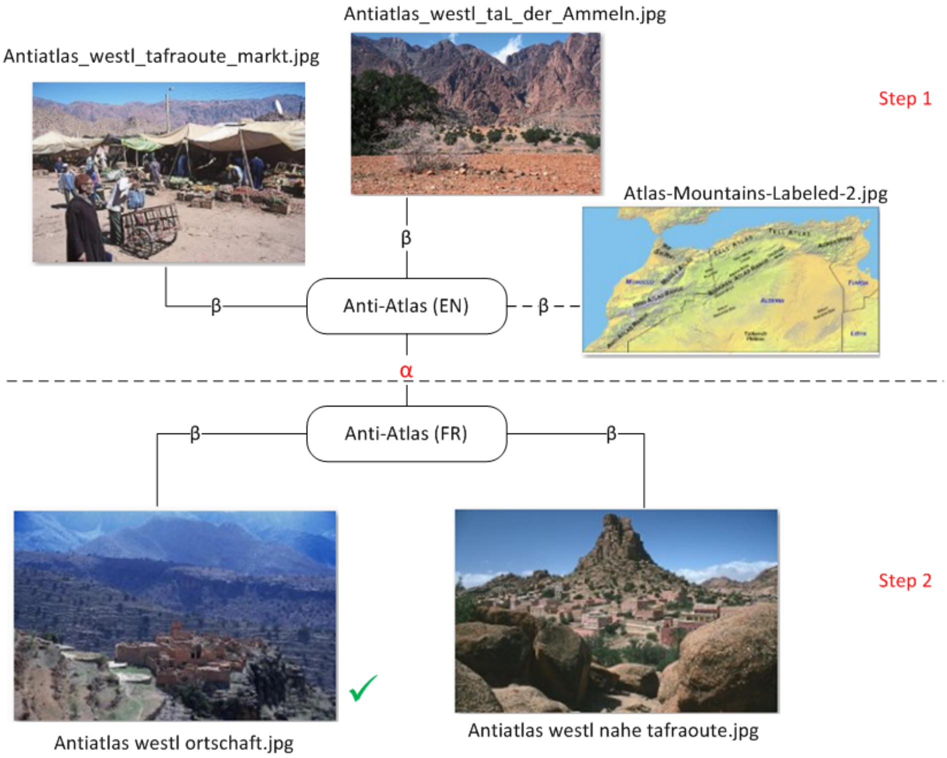
We described in Chapter 3 that we enrich the collection with two types of semantic links: inter-lingual and intra-lingual. As we consider only English metadata in our experiments, therefore, only the English documents of the collection and their images are connected. We utilize the inter-lingual links to reach the German and French documents as well. The only inter-lingual link used is the `sameAs` link. The reason for choosing this link is described in Section 3.2.2. Adding a semantic link like `sameAs` also has the side effect of removing the bipartite nature of the graph. The energy is no longer completely transferred between the images and documents, as we have new document neighbours for the documents.

Here, we compare reachability to relevant images in each step with/without semantic links. Figure 5.3 shows the graph when we use the TF.IDF$_D$ facet result as starting points with $st = 0$. Without any semantic link, all the energy goes to the neighbours in each step and we visit images only in odd steps. However, if we consider the `sameAs` links in the first step, the energy goes not only to the images, but to the semantically related documents as well. In the second step, the energy shifts from images to their parent documents, and from new semantically related documents to their images. In this step, we see images only reachable through `sameAs` links.

We designed two experiments in this section. As shown in Table 5.8, without `sameAs` links we have precision zero in even steps. The reason is that we visit only documents in this step. In Table 5.9, we see the result of adding `sameAs` links. In the second step we see the images reachable only through `sameAs` links. We observe the amount of 0.22 increase in the precision value in the second step, and 0.2 in the fourth step. Two examples are shown in Figures 5.4a and 5.4b, one as a sample of easy topics and the second one as an example of very hard topics. We observe from these figures that we can reach more relevant images following the semantic links.



(a) For the query of "Elvis Presley" (as an easy topic), we visit one of the relevant images in the first document in the first step. In the second step, focusing on semantic links, we reach three more relevant ones through German and French documents.


Antiatlas_westl_taL_der_Ammeln.jpg
Antiatlas_westl_tafraoute_markt.jpg
Step 1
Atlas-Mountains-Labeled-2.jpg
β
β
Anti-Atlas (EN)
β
α
Anti-Atlas (FR)
β
β
Step 2
Antiatlas westl ortschaft.jpg
Antiatlas westl nahe tafraoute.jpg

(b) For the query of "house in mountain" (as a very hard topic), we visit no relevant images in the first document (Anti-Atlas) in the first step. In the second step, we reach a relevant image through the French document.

Figure 5.4: Two examples of following inter-lingual semantic link

Table 5.8: Result without semantic links, st = 0, β, δ

steps	p@10	r@10	p@20	r@20
1	0.34	0.13	0.25	0.16
2	0	0	0	0
3	0.28	0.11	0.20	0.15
4	0	0	0	0

Table 5.9: Result with semantic links (sameAs), st = 0, β, δ, α

steps	p@10	r@10	p@20	r@20
1	0.34	0.13	0.25	0.16
2	**0.22**	0.1	0.2	0.14
3	0.28	0.11	0.22	0.14
4	**0.2**	0.07	0.16	0.11

5.2.2 Similarity links - $\alpha, \beta, \delta, \gamma$

In this experiment we include the similarity links added to the collection. We design experiments with/without similarity/semantic links. By adding any of these links, the graph is no more bipartite. We set the $st = 0.9$ in these experiments to include all the visited images up to this step in the performance of the current step. In Table 5.10, we show the results without adding any similarity/semantic links. Table 5.11 shows the results by adding only similarity links. In this table, we observe almost no difference in the results. It seems that adding similar documents and images does not help with improving the precision. The results of Table 5.12 show that in the absence of similarity links and existence of only inter-lingual semantic links and $st = 0.9$, we have the best precision of 0.37 in the second step. By leveraging semantic links we gain 9% increase in the precision value in this step, as the images form other relevant documents (French, German) are added to the result list.

Table 5.13 shows the result of adding both similarity and semantic links. We observe less increase in the precision than adding only semantic links. The reason is that, by adding both links we reach a large number of images in each step. In addition, with semantic links new images from German or French documents are added to the final ranking list.

5.2.3 Discussion

With Astera we can search the collection from different points of view by using different facets. We showed the effect of using textual facets in combination with visual facets on precision value. We were able to improve the results by using a weighted combination of textual and visual facets. Utilizing image textual facet increased precision by 6%. Further, combination of two different facets of images (R_{IV} and R_{IT}) with document textual facet (R_D) leaded to better result than the sum of their individual results. This demonstrates the positive effect of the combination of different facets and poly-representation in Astera. Adding inter-lingual semantic links increased the precision value by 9%. However, adding similarity links did not help with the performance.

Table 5.10: Result without semantic/similarity links, st = 0.9, β, δ

steps	p@10	r@10	p@20	r@20
1	0.34	0.13	0.25	0.16
2	0.34	0.13	0.25	0.16
3	0.28	0.11	0.20	0.15
4	0.2	0.09	0.20	0.14

Table 5.11: Result with similarity links, st = 0.9, β, δ, γ

steps	p@10	r@10	p@20	r@20
1	0.34	0.12	0.25	0.16
2	0.34	0.12	0.24	0.16
3	0.27	0.09	0.24	0.15
4	0.24	0.08	0.22	0.13

Table 5.12: Result with semantic links (sameAs), st = 0.9, β, δ, α

steps	p@10	r@10	p@20	r@20
1	0.34	0.12	0.25	0.16
2	**0.37**	0.12	0.25	0.16
3	**0.35**	0.10	0.22	0.14
4	0.26	0.09	0.20	0.13

Table 5.13: Result with both semantic and similarity links, st = 0.9, β, δ, α, γ

steps	p@10	r@10	p@20	r@20
1	0.34	0.11	0.25	0.16
2	0.34	0.11	0.24	0.16
3	0.27	0.09	0.24	0.15
4	0.24	0.08	0.19	0.12

5.3 Score Analysis in the Graph - $\alpha, \beta, \delta, \gamma$

In previous section, we investigated the role of different starting points and their combinations in improving the precision value. In addition, we performed experiments to examine the role of enriching the collection with similarity and semantic links. The results showed that we obtain better performance by adding semantic links. However, adding only similarity links or combination of the two, degraded performance. With these findings, we are interested to find what is happening in the score distribution in the graph [SLR15]. Why do we not see a significant increase in the precision values? To answer this question, we designed experiments to observe the score distribution in very large steps in the graph. We normalize the weights in the graph to examine graph score distribution in the convergence state. Practically, this is now Random Walks (as shown in Chapter 2).

5.3.1 Experiment Design

We described in Section 3.2.4 that for efficiency reasons we create query-dependent sub-graphs, as we start from top results of that query as starting points. For each topic we start from top 20 results based on one or more facets. According to the hardware feasibility analysis of our resources in Appendix A, we are limited to have sub-graphs of 70,000 nodes. This size of a sub-graph does not influence our model and traversal method. We check the total number of visited nodes after each step and if it is exceeding the limit, we remove randomly from the neighbors of the last visited nodes. For each topic, we trim the graph to 70,000 nodes.

Based on the size of the neighbour list of each node (N), we simply calculate the weight of $1/N$ for each neighbour. This way, we create a sub-graph with normalized weighting on the edges, satisfying the stochastic property. Now, the sub-graph is ready to perform the iterations. We use the term *step* as theoretical multiplication of the graph matrix. From an *iteration*, we mean each mathematical matrix multiplication, which may include one or many steps.

Number of Iterations It usually takes more than 1000 steps that we reach the stationary distribution state [Wal04]. We choose steps of power of 2 to create big jumps. It helps recognizing convergence in the graph. To determine the number of steps needed, we check the convergence in each step. At each iteration, we calculate the difference between the score vector of the previous iteration $a^{(t-1)}$ and the current one $a^{(t)}$. We check the convergence in each iteration based on the absolute error and relative error. The absolute error shows the difference of the value of each node in the two vectors $a^{(t-1)}$ and $a^{(t)}$ as $abs_{err} = |a^{(t)} - a^{(t-1)}|$. Based on the precision we want to obtain, we define the relative error. This error shows the magnitude of the difference between each value in $a^{(t-1)}$ and $a^{(t)}$ vectors as $rel_{err} = abs_{err}/|a^{(t-1)}|$.

We consider the score values larger than $1e - 3$ in our ranked list. Therefore, we choose $1e - 4$ as the absolute error value of convergence, indicating no difference between a node score in two iterations. We set the accuracy value of $5e - 2$ for relative error as enough accuracy between two iterations. According to our checking for different topics, we reach the convergence after step 2048 (12 iterations).

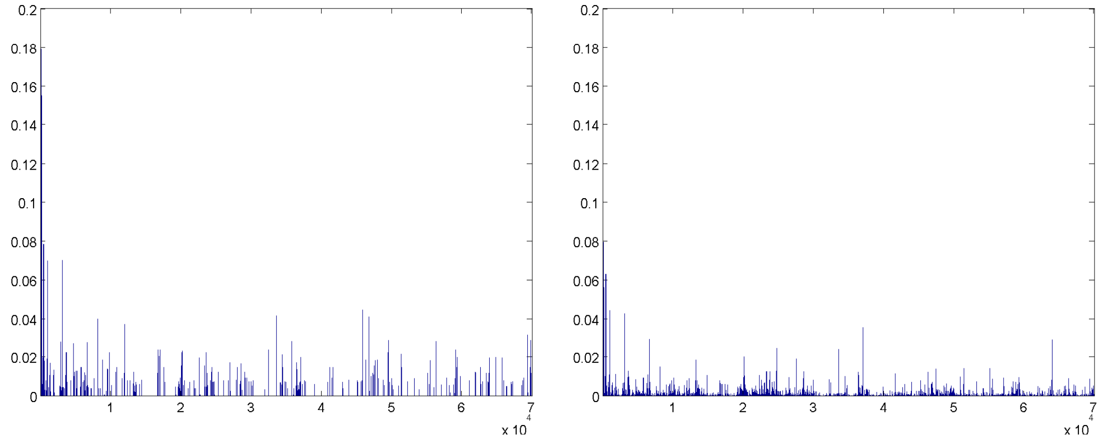
5.3.2 Score distribution in the graph

To start the experiments, we take snapshots from Random Walks steps to obtain an understanding of score distribution in the graph. We show the results for a sample topic (Topic 83). The score distribution is shown in steps of $1, 2^3, 2^5, 2^7, 2^9$, and 2^{11} (Figure 5.5).

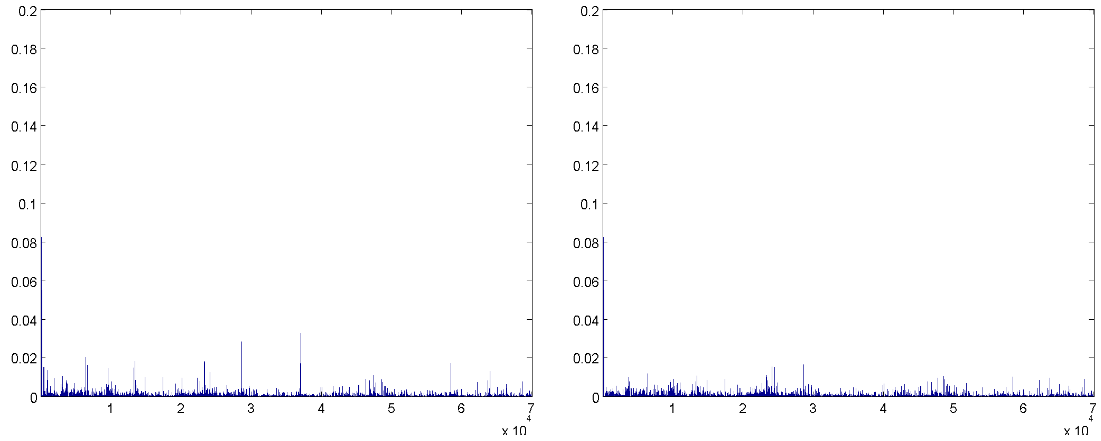
We observe that in the starting steps, fewer nodes have received the energy/score from starting points with higher values. As we traverse further in the graph, the energy is more evenly distributed. After many iterations, we have the same amount of energy spread over the nodes. This energy is the sum of normalized scores of the top results of a query. However, there are some nodes which have still higher energy than the others, and are returned as ranked list.

5.3.3 Query-dependent and Query-independent Routing

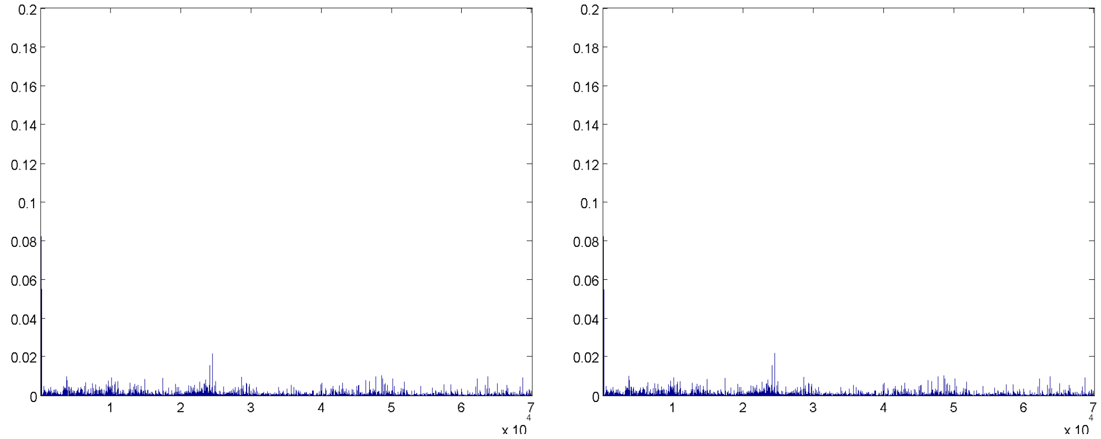
By query-independent routing, we mean that the traversal in each step is independent from the relevance to a query. It is only based on the pre-defined weighting in the graph. We use Markov Chain Random Walks for this traversal.



(a) Step 1 (b) Step 8



(c) Step 32 (d) Step 128



(e) Step 512 (f) 2048

Figure 5.5: Score distribution in the graph in different steps for topic 83. The x axis is all the nodes in the graph which is 70,000. Units are in the scale of 10^4.

In the literature, Random Walks and its stationary distribution have been mostly used to benefit from the graph structure. The question arises whether we obtain different results if we use a query-dependent routing [SLR16]. To answer this question, we employ one of the query-dependent Random Walks methods named Metropolis-Hastings [CG95]. This algorithm is described in detail in Section 2.1.4. By query-dependent routing we mean that in each step we consider the relevance of each node and that of its neighbour to the query.

Metropolis-Hastings properties in Astera

Here we check how our instantiated model satisfies Metropolis-Hastings restrictions.

- **Irreducibility**: To check irreducibility we should prove that our graph is connected. By adding different relations of β, γ and α, we have a connected graph. For this purpose, starting from top ranked results for a sample query we traverse the graph. In each step we visit new neighbours and continue until we see no more new nodes. In addition, we add similarity links between the top results of a query topic. The number of nodes seen in this traversal is the whole graph size. This construction and observation, even for one query, demonstrates the connectivity of our graph.

- **Aperiodicity**: There is no rhythm in moving from one state to another. The number of states in each move is not multiple of some integer. We satisfy this constraint by construction.

- **Stochastic property**: According to the weight definition in Astera for β links, the sum of weights on a row may be more than one. However, semantic (α) and/or similarity (γ) links can be used in a normalized form, complying with stochastic property. In addition, in experiments with Random Walks as traversal method, we define the weight of $1/N$ for N neighbours of a node to satisfy the stochastic property.

Transition Matrix

According to the Metropolis-Hastings algorithm defined in Section 2.1.4 and Equation 2.21, we sample from $W(x, y)$ and accept the move with probability $\lambda(x, y)$. This has implications on how we define high-order transition probabilities after t steps:

$$Pr_q^{t+1}(x, y) = \sum_{i=1}^{k} Pr_q^t(x, z_i) \cdot Pr_q^t(z_i, y) \tag{5.2}$$

where q is the query, Pr^t is the transition probability of starting from x and moving t steps further, z_i can be any node in the matrix, and k is the number of the nodes. The new matrix with updated weight is called the *transition matrix.* Leveraging Metropolis-Hastings, the edge weights are affected by λ in each step (Eq. 2.21). We repeat the λ

formula here:

$$\lambda(x,y) = min\left[\frac{\widetilde{\pi}(y).W(y,x)}{\widetilde{\pi}(x).W(x,y)}, 1\right] \tag{5.3}$$

Updating each edge in each iteration slows down the matrix multiplication. The reason is that, in each iteration, the matrix W is affected by λ. It will be $W \cdot \lambda$ in the first iteration and $W \cdot \lambda \cdot \lambda$ in the second iteration.

However, Hlynka et al. [HC09] proved that the transition matrix Pr does not change in further steps. Therefore, we need to compute only once the matrix of $Pr(x,y) = W(x,y) \cdot \lambda(x,y)$ for all the nodes and their links. We can use this matrix in further multiplications.

We compute the final score as $a^{(t)} = a^{(0)} \cdot Pr^t$ after t steps. This computation is needed for intermediary steps, since in ideal case the multiplication is performed many times until the matrix converges. In the stationary distribution, the nodes' probabilities are independent of starting scores in the graph.

Metropolis-Hastings Algorithm in Astera

To apply Metropolis-Hastings in Astera, first we need to define what we we are looking for in the stationary distribution of $\pi(x)$. As a stationary distribution over the set of nodes, we would like to approach the *true* relevance probability distribution by the indexing ranked results. It will be a query dependent stationary distribution such that the probability in node x is proportional to the probability that this node is relevant to the query, and at any other node (non-relevant) the probability is zero. This is the $\pi(x)$ distribution from which we cannot directly sample. Instead, we have the $\widetilde{\pi}(x)$ which can be a relevance scoring function (e.g. a BM25 score between the information object x_i and the query). Metropolis-Hastings formally provides a method to sample from the probability distribution, if the approximate probability $\widetilde{\pi}(x)$ is properly chosen. Further, we need a probability which suggests the neighbour y, when we are in the node x. This is in the form of the matrix W, which plays the role of *jumping distribution.* Mapped to Astera, the proposed matrix W is our stochastic transition matrix. We use this transition matrix as jumping distribution to find the next neighbour.

Now, we have the graph of different relations in the adjacency matrix W. To generate $\widetilde{\pi}(x)$ values, we need the relevancy of each node to the query in each step. For this reason, we provide the relevancy of each node based on their facets to the query. Assuming the true relevancy of the nodes to the query as $\pi(x)$, we define the $\widetilde{\pi}(x)$ as the relevance score value function (RSV) in our model, as defined in Section 3.1.1. To perform any jump in Metropolis-Hastings, we need to compute λ (Equation 5.3), so we need RSV values of the source and destination node. Suppose that we start from similarity with TF.IDF$_I$ results, we will have a set of images as starting points to do the traversal. Each image is connected to at least one parent document (D) through a β link. To compute the $Pr(I, D) = W(I, D) \cdot \lambda(I, D)$, we need the λ value, which is:

$$\lambda(I, D) = \left[\frac{RSV(Q, D)}{RSV(Q, I)} \cdot \frac{W(D, I)}{W(I, D)}, 1\right] \tag{5.4}$$

where

$$\begin{aligned} RSV(Q, I) = norm(sim(Q_{TF.IDF}, I_{TF.IDF})) \cdot w_{TF.IDF} + \\ norm(sim(Q_{CEDD}, I_{CEDD})) \cdot w_{CEDD} \end{aligned} \tag{5.5}$$

where an image object (I) has two facets of $\{TF.IDF, CEDD\}$. The common set of facets of l between the query and image is $l = \{TF.IDF, CEDD\}$. For $RSV(Q, D)$ we have

$$RSV(Q, D) = norm(sim(Q_{TF.IDF}, D_{TF.IDF})) \cdot w_{TF.IDF} \tag{5.6}$$

The RSV value is computed based on the normalized Lucene and LIRE similarity score for TF.IDF and CEDD facets respectively. The w_{CEDD} and $w_{TF.IDF}$ are facet weights for this query.

Uni-modal IR and Metropolis-Hastings

We can easily satisfy the stochastic property with one modality in our model. For example, we can include only the relations between the documents and get their images in the final list as they are ranked. This way, the relations in the graph consist of δ relation between R_D facet and the document, and γ relation between the same R_D facets of two documents. To compute the weights on γ edges, we find similar documents for each top document result. It can be based on its TF.IDF facet, using standard search. We normalize the similarity values for each document.We add these similarity links for each document. This way, the stochastic property is met: $\sum_{i=1}^{N} W(x, y_i) = 1$, where N is the number of the neighbours. Our graph is asymmetric, i.e., $W(d_1, d_2)$ may have different value than $W(d_2, d_1)$. This means that two neighbour documents do not have the same ranked similarity for each other. Mapped to the Equation 5.3 and our model, λ is:

$$\lambda(d_1, d_2) = \left[\frac{RSV(Q, D_2)}{RSV(Q, D_1)} \cdot \frac{W(D_2, D_1)}{W(D_1, D_2)}, 1\right] \tag{5.7}$$

where $RSV(Q, D_1) = norm(sim(Q_{TF.IDF}, D_{TF.IDF}).w_{TF.IDF})$. The RSV value is computed based on normalized Lucene result value for this facet. The relevancy is computed between document TF.IDF and the query TF.IDF facet. We take this value as relevancy value of each document for this specific query, and the probability of going from d_1 to d_2 is $Pr_q(d_1, d_2) = W(d_1, d_2) \cdot \lambda(d_1, d_2)$.

Multimodal IR and Metropolis-Hastings

In multimodal IR, the nodes in the graph may be of any modality. This way, having a node x, the next node may be based on any of its relations: α, β, γ or δ. One approach to satisfy the stochastic property is to define the same weighting of $1/N$ for N neighbours of a node. However, it remains as a challenge to define different weightings for various link types with the constraint of stochastic property.

For example, assume that the query is a combination of text and image (a multimodal query). As we reach a node with Text or Image modality, we compute the relevancy of that node to the query. In each node, we consider facets of the same type with the query. For instance, if the object is an image, we find the relevancy based on CEDD and TF.IDF$_I$ facets, and only based on TF.IDF$_D$ facet if it is a document.

5.3.4 Experiments

To compare the query-dependent and query-independent routing in our model, we design different experiments in this section. From here, we compare the two methods of Random Walks and Metropolis-Hastings as query-independent and query-dependent Random Walks in our experiments.

Precision Analysis

We compare the performance of Random Walks and Metropolis-Hastings in this experiment. We calculate the precision and recall using both methods. We choose TF.IDF$_D$ and TF.IDF$_I$ facets for computing relevancy in each step. As before, we start with top 20 results based on these facets for each query. We traverse the graph from these 40 starting points. We consider all three languages (English, German and French) for finding relevancy of an information object to the query. This is especially needed for image metadata indexes, as many images do not contain metadata information in all languages. We consider the relevancy of an image in the three languages. This way, the relevance value is influenced by the relations of that node.

The result in Table 5.14 shows that by leveraging Random Walks, in the steps higher than 16, the precision value is zero. This indicates that the top ranked results in stationary distribution are not relevant to the query. With these two algorithms, we are more interested in graph behaviour in steps higher than 16. We observe from Table 5.15 that with Metropolis-Hastings we obtain positive values for precision and recall in all iterations. Comparing the two tables, we observe lower values only in the first step for Metropolis-Hastings. The reason is that our relevancy function considers only metadata indexes and all images do not have enough text in this part. However, from the 2nd step, Metropolis-Hastings shows higher values in precision and recall.

In the stationary distribution with Random Walks, the only influencing parameter in the result vector is the graph structure. However, with Metropolis-Hastings, we include in addition the relevancy to the query of each information object in each step. The precision values could increase by using better relevancy functions for images and documents.

Correlation Analysis

In previous experiments on precision at cut-off, we found that in higher steps, the top ranked results are rarely relevant to the query. Why have these images obtained high scores?

We perform rank correlation analysis based on the top ranked results and the number of their neighbours. We want to investigate the result bias to the number of neighbours of a node. In each step, we get the top ranked images. For each image we find the number of its neighbours and create another ranked list. We calculate the correlation between these two lists. We use Spearman correlation, as it does not make any assumption about the distribution of data. It is an appropriate choice when the variables are measured on an ordinal scale. We compute the correlation values in each step for both algorithms.

We observe from Figure 5.6a that both methods show low correlation value between the top ranked results and the number of incoming links. However, Metropolis-Hastings as a query-dependent traversal method, shows lower correlation values than Random Walks up to the step 2^9. We performed the same analysis with the number of outgoing links of top ranked results (Figure 5.6b). We observe that the correlation between the rank and the number of outgoing links in both methods is less than 0.5. However, both algorithms show higher correlation value to the number of outgoing links. This shows that nodes with high fan-outs in the graph influence both algorithms.

Further, we investigate the rank correlation of top ranked images to the sum of the weights around a node. In Figure 5.6c and 5.6d, we show the correlation result between the top ranked nodes and the sum of the weights on the incoming and outgoing edges of each top result node. We observe that Metropolis-Hastings holds higher value for correlation value in both cases. We find that this method is more influenced by the weighting in the graph. According to Metropolis-Hastings algorithm, to find the next jumping destination, besides the relevancy of the destination node to the query, the incoming weight of potential destination node influences the choice of next neighbour (Equation 2.21). Therefore, the weights on incoming links of a node influences the jump decision to the next neighbour in Metropolis-Hastings algorithm. However, the number of incoming links is not a factor.

Query influence versus graph structure

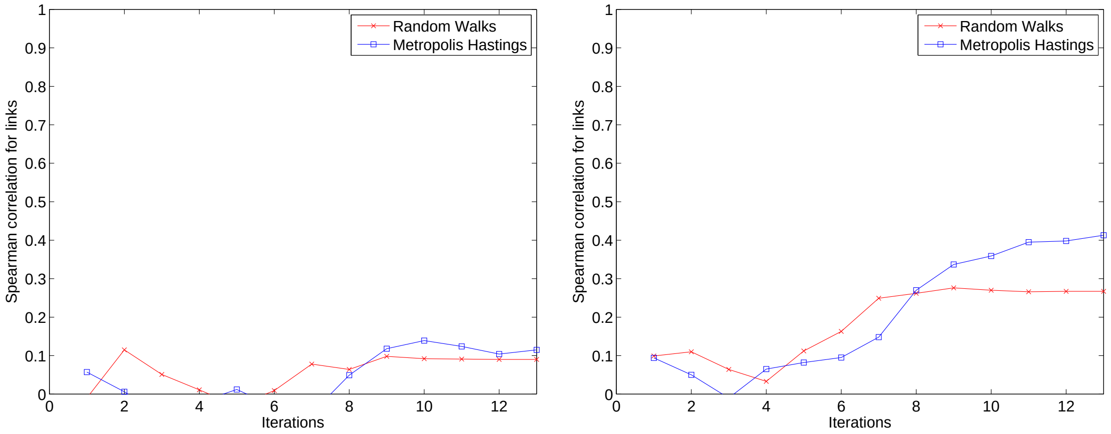
After correlation analysis, we want to investigate how fast the initial score influence is overriden by the graph structure using these methods. We compare two result vectors $(a_1^{(t)}, a_2^{(t)})$ in each iteration: one computed based on initial vector $(a_1^{(0)})$ composed of Lucene results; one computed based on equal score of $1/N$ for all nodes $(a_2^{(0)})$. This way, there is no preference between initiating points and we have only the effect of the

Table 5.14: Performance result with Random Walks

iter	*steps*	*p@10*	*r@10*	*p@20*	*r@20*
1	1	0.2267	0.0815	0.1767	0.1111
2	2	0.1571	0.0478	0.15	0.0866
3	4	0.1	0.0293	0.1067	0.0523
4	8	0.0857	0.0209	0.0643	0.0364
5	16	0.0133	0.0027	0.0333	0.0175
6	32	0.0	0.0	0.0	0.0
7	64	0.0	0.0	0.0	0.0
8	128	0.0	0.0	0.0	0.0
9	256	0.0	0.0	0.0	0.0
10	512	0.0	0.0	0.0	0.0
11	1024	0.0	0.0	0.0	0.0
12	2048	0.0	0.0	0.0	0.0
13	4096	0.0	0.0	0.0	0.0

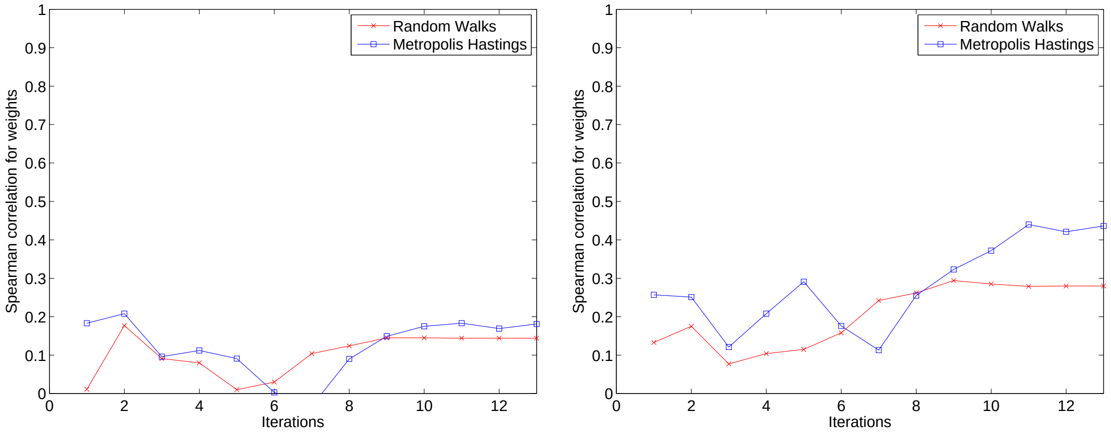
Table 5.15: Performance Result with Metropolis-Hastings

iter	*steps*	*p@10*	*r@10*	*p@20*	*r@20*
1	1	0.1958	0.0501	0.1479	0.0711
2	2	**0.216**	**0.065**	**0.16**	**0.0836**
3	4	0.175	0.0432	0.1562	0.0787
4	8	0.1333	0.029	0.1292	0.0582
5	16	0.1125	0.0284	0.0833	0.0354
5	32	0.068	0.0145	0.056	0.0189
7	64	0.032	0.0079	0.018	0.0081
8	128	**0.0042**	**0.0038**	**0.0021**	**0.0038**
9	256	0.0042	0.0038	0.0021	0.0038
10	512	0.0042	0.0038	0.0021	0.0038
11	1024	0.0042	0.0038	0.0021	0.0038
12	2048	0.0042	0.0038	0.0021	0.0038
13	4096	**0.0042**	**0.0038**	**0.0021**	**0.0038**



(a) Correlation between the top ranked results and the number of **incoming links**

(b) Correlation between the top ranked results and the number of **outgoing links**



(c) Correlation between the top ranked results and the **incoming weights**

(d) Correlation between the top ranked results and the **incoming weights**

Figure 5.6: Correlation between top results and the links/weights of these nodes

graph structure. We compute the result vector in each step based on Equation 5.3. With the second initial vector ($a_2^{(0)}$), we remove the effect of scores of starting points in the traversal.

For each topic, in steps of power 2 (1, 2, 4,..., 4096), we calculated the $a_1^{(t)} = a_1^{(0)} \cdot W^t$ and $a_2^{(t)} = a_2^{(0)} \cdot W^t$ vectors. We perform this analysis for both methods. We calculate the Cosine similarity between $a_1^{(t)}$ and $a_2^{(t)}$ (Figure 5.7). We observe that, in the first and second iterations in both methods, the similarity is very low—the results are highly dependent to the starting point scores. Surprisingly, after step 2^6, the result vectors are converging. We observe that the result vectors of both methods approach to highly similar values of $a_2^{(t)}$. This is exactly what the stationary distribution means that the

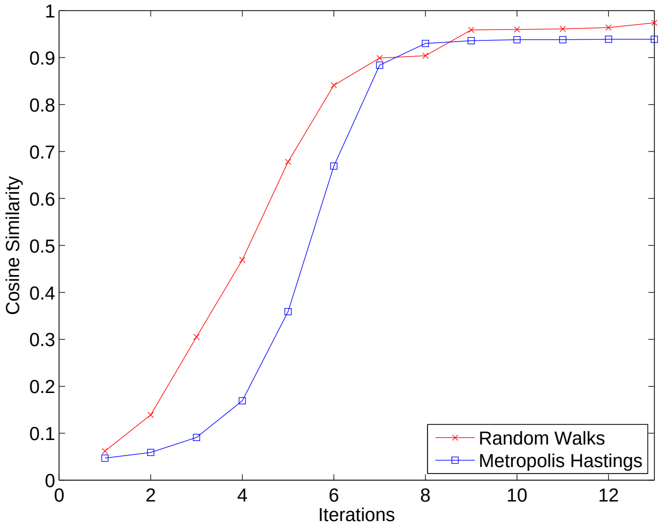


Figure 5.7: The rate that the initial score is overriden by the graph structure with Random Walks and Metropolis-Hastings

stabilized values in each node is independent of initial values. However, we observe that with Metropolis-Hastings, we have lower similarity values in the initial steps. This observation is consistent with the results in Tables 5.14 and 5.15 that we obtain better precision with Metropolis-Hastings algorithm as a query-dependent traversal method.

5.3.5 Discussion

We formulated a Random Walks problem on our proposed model for graph-based multimodal IR. We have the opportunity to examine query dependent traversal, as weights in the graph are affected by relevancy of source and target nodes to the query. We compared the performance of the model leveraging query-dependent and independent Random Walks in our graph model. The results show that by considering query relevancy in each step, we obtain higher precision results. This result can be improved by using better relevance functions. Moreover, we showed that query-dependent results are more influenced by the graph structure. Finally, we found that with Metropolis-Hastings as a query-dependent method, we can expect more relevant results not only in the first 32 steps.

5.4 Summary

We investigated the role of different facet combination on precision in the first part of this chapter. We showed that leveraging different facet types supports the polyrepresentation idea in IR. We obtained higher precision value by using both textual and visual facets. Further, we showed that by using semantic links, we can reach to a new set of relevant information objects and increase precision as well. However, the increase in precision values was mostly in the initial steps. This motivated us to investigate the score distribution in very large steps. We designed query-dependent and query-independent Random Walks in the graph. We showed that using query-dependent traversal leads to more promising results. However, the value of precision is low in the stationary distribution state.

We performed rank correlation analysis between the top ranked results and the number of incoming/outgoing links of these nodes. The results showed that with Metropolis-Hastings we observe less correlation to the number of incoming links. Further, the correlation analysis results with the sum of the weights on incoming/outgoing links showed that weighting in the graph influences the Metropolis-Hastings results.

CHAPTER 6

Conclusion

In this chapter, we present the conclusion of our research by highlighting the significance of this thesis in terms of summarizing the contributions and their implications for the performance analysis in graph modelled multimodal collections. After giving a summary, we revisit the research questions in Section 6.2. Section 6.3 states the ongoing trends and open topics in related research areas for future research to build upon the contributions presented in this work.

6.1 Summary

The work within this thesis is to answer the question: How can we benefit from different modalities to improve performance in the task of multimodal Information Retrieval?

To address this question we proposed a generic model for multimodal Information Retrieval. It is a graph-based model, in which the nodes are the modalities and different relations between information objects are the links. However, there are challenges in such a model such as how to model this heterogeneous data, how to define different relations between information objects, how to traverse the proposed graph, how much this graph model helps with reaching relevant objects, and how to affect different modality features in the final result.

We tackled these challenges in this thesis by describing the model in Chapter 3 and investigating the reachability to relevant information objects in Chapter 4. Finally we analysed the score distribution in the graph in Chapter 5. We evaluated this model with ImageCLEF2011 Wikipedia 2011 test collection as a multimodal collection.

6.2 Research Questions Revisited

The research questions introduced in Section 1.3 are revisited here. In the following, we summarize how the research questions were addressed within this work and what limitations still remains.

6.2.1 RQ1:Can we define a graph-based model for multimodal multi-faceted information retrieval?

We defined a generic model, named Astera, for multimodal Information Retrieval based on a graph of information objects. It is a generic model as the nodes can be from any modality (Text, Audio, Video, Image, etc), and the links include both similarity and semantic. One characteristic of this model is a multi-faceted view to an information object. From facet we mean an inherent feature, a property or metadata of an information object. It is based on the poly-representation principle in IR [Ing96, IJ06] to leverage different representations of a modality to perform a better search. We see an information object as a set of facets, so there in no difference of how to deal with an image, a document or a video clip file. By adding a separate node for each facet of an information object, we provide the possibility of utilizing each of them in the retrieval process. We used our defined faceted-search in this graph. The standard indexed results of facets are used to find the starting points in the graph, from which the traversal begins.

In Astera we can model different types of data collections with various modalities and different link types. We can enrich the modelled collection by extracting features of information objects as facets. Further, we can add semantic/similarity links between information objects. The relevancy of an information object to a query is calculated based on the similarity between their common facets. The search in the graph is performed in two phases. We start from the result of an indexed search to find good starting points in the graph, then continue with a graph traversal method. Two well-known graph traversal methods, Spreading Activation and Random Walks, are utilized in this thesis.

6.2.2 RQ2: In such a graph model, can the relevant nodes be reached?

To tackle this question we designed a series of experiments to test the reachability of relevant information objects. The experiments are based on different facets and links to explore the effect of each separately on reachability to relevant information objects.

First, we examined the effect of leveraging multiple facets. We designed experiments to start from the top results based on textual facets of the documents and textual and visual facets of images. The results show that the combination of document and image textual facets shows better results than only document textual facet. Another observation was that easy and medium topics are mainly reachable in initial steps. However, we observed an increase in recall of 266% and 373% for hard and very hard topics, respectively by traversing the graph. Furthermore, combining textual facet from documents and images led to higher recall than using different textual facets of only one modality as document or image. These observations confirm the effect of poly-representation to reach a higher number of relevant information objects.

In addition, enriching the collection with semantic links from corresponding DBpedia dump helped with 13% and 8% in reachability of relevant information objects for hard and very hard query topics. We compared the result of graph enriched with real semantic links to the one with random semantic links. We learned that in the process of enriching the graph with more links, these links should be chosen meaningfully. We lost 88% precision by adding random semantic links compared to adding real semantic links.

In these experiments, we observed similar recall behaviour in each category of document textual, image textual and image visual facets. We designed experiments to investigate whether we visit the same relevant nodes starting from different facet results. We found that facets with the same recall behaviour do not visit necessarily the same relevant nodes. This motivates to use diverse facets as starting points to traverse the graph. The comparison results about three textual facets (TF.IDF, BM25 and LM), both for documents and images, showed that the Language Model (LM) facet has more divergent view than the other two textual facets. In addition, in image visual facets, CEDD passed through parts of the graph that the other three visual facets did not. These findings encourage to utilize different facets. Mapped to poly-representation principle, cognitively different facets (e.g. LM and CEDD) lead to visit more parts of the graph than only functionally different ones (e.g. TF.IDF and BM25).

We conclude that although starting from different facet results helps reaching more parts of the graph, which is a potential to obtain a higher recall, we should consider the much higher computational complexity that is required for reaching more nodes. As we saw in the experiments, although some facets show low overlap in the visited nodes in the graph, the recall difference is low. Therefore, finding optimal set of facets to traverse the graph with minimum overlap is a challenge ahead to be tackled in our future work. Further, we observed how connected the collection is. We found the minimum number of steps of 30 that still there is a chance to see relevant information objects.

6.2.3 RQ3: In such a model can scores identify the relevant nodes?

Besides finding relevant nodes, we were curious about the score distribution in the graph. First, we started with the role of different facets in precision. The results demonstrate the positive effect of the multi-faceted view (poly-representation) on precision. The best result was obtained by combination of document textual, image textual, and image visual facets. We obtained 9% increase in precision compared to using only document textual facets. In addition, we showed that using semantic links helps reaching relevant images which have only German or French metadata.

However, the precision increase was only at the starting steps. We designed experiments to traverse the graph in very large steps to reach the stationary distribution state. We formulated query-dependent and query-independent Random Walks. We used Metropolis-

Hastings as query-dependent Random Walks. In this method, weights in the graph are affected by relevancy of the source and target nodes to the query. We found that with Metropolis-Hastings as a query-dependent method, we can expect more relevant results and a higher precision not only in the first 32 steps. We were curious how much the final ranking is affected by the graph structure. Hence, we investigated the contribution of the graph structure (quantified by the number and weights of incoming and outgoing links) to the final ranking in both types of Random Walks. As correlation of both Random Walks and Metropolis-Hastings to the number of incoming links was lower than 0.2, we conclude that the result of none of these methods is dependent to the number of incoming links. Based on the weight correlation analysis, we found that Metropolis-Hastings, as query-dependent Random Walks, is more influenced by the weighting in the graph. The reason stems in the definition of this method which may change the weighting of the graph in each step of traversal.

6.3 Future Work

While this thesis presented solutions to a number of important issues, there are still some questions that could not be fully addressed. These questions define the starting point for the future research and the further improvement of Astera as a model for multimodal IR.

Performance and Scalability We had to limit the graph size to traverse large number of steps in the graph. We map the graph to a matrix in Matlab to perform matrix multiplications. Although we use sparse matrices, in higher number of multiplications, the matrix is not sparse any more. One approach to keep the matrix sparse is to define a threshold to keep values higher than this threshold in the matrix and to assign zero value to cells with values less than it. This way, we can manage larger size of the graph in its sparse matrix format. As a solution we used Vienna Scientific Cluster[1] (VSC) to target parallel running of the program with high hardware configuration. However, our bottleneck is RAM of the system. The ImageCLEF 2011 Wikipedia collection requires high RAM configuration in the scale of 300GB. The VSC nodes have at most 128GB of RAM.

Probability In order to satisfy the stochastic property in Random Walks, we defined simple normalized weighting on edges of a node based on the number of its neighbours. This affects the meaning of the weights in our model. For example, if a document has 10 images as neighbours, each of them receives 0.1 weight, and if 2, they receive 0.5. We see that in this case we are biased to the number of images in a document. In this example, the equal values on probabilities give the same chance to any image in the document as the next step in traversal. However, as the scores propagate in the graph, the value of the weights on edges play role as well. This is the challenge ahead that should be addressed when Random Walks method is applied in the graph. As another

---

[1] http://vsc.ac.at/

example, if a document has 5 images and is semantically related to 2 other documents and has 10 similar neighbours, the questions is how do we properly define the probability of moving from this document to each of its neighbours with different relation types. This is in our future work to define the probability in a multimodal graph with different link types to satisfy stochastic property.

Stationary Distribution Cost We used two approaches in considering the score distribution in the graph. First, we considered the scores in the initial steps; second, in the stationary distribution state. However, it takes more than 1000 steps normally to reach to the stationary distribution state [Wal04]. We used matrix multiplications in the power of two to reduce the computational time in reaching high number of steps. However, we miss the results of the steps in between. How expensive is our approach regarding the need of high number of transitions until the matrix converges? One future work is to find a compromise between computational time requirement and snapshot results that we require.

Further Semantic Analysis We leveraged DBpedia semantic links only inside the collection. Another approach to explore is to further explore the semantic relations between the ImageCLEF 2011 Wikipedia collection and DBpedia. For example, we can traverse the graph starting from the collection, go through DBpedia links until we return to the collection again. The semantic length between the source and destination nodes can be used as semantic distance of these two objects.

Further, we can create semantic facets for the images in the collection. We can use concept detection methods [BBB^{+}14] to extract image concepts. The concepts can be seen as a document and be added to the images as textual semantic facet. In this case, we extract the semantic facet of the query image example, and the search can be performed based on this facet as well.

Learning the Weights In the current version of Astera, we set the weights of different facets based on our analysis of their best combination. Further, we define weighting strategies for different relation types in our model. One future direction is to apply machine learning algorithms in both cases. For example, we can use Relevance-Feedback methods to leverage user clicks in learning the weights on the edges of a clicked result, or use SVM method to learn the optimal weight of each facet for different queries.

APPENDIX A

Computational Complexity Analysis

The ImageCLEF 2011 Wikipedia collection contains a total of 363,262 information objects (images and documents). We show the number of nodes needed in the graph for adding different parts in Table A.1. In the basic model of the collection (without adding facet, similarity, and semantic links), we have images and documents. To simulate the graph traversal, we make an adjacency matrix out of this graph (2.11). The full matrix is of size $363,262 \times 363,262 \times 8B$ (matrix elements are saved in double with 8 bytes memory). With matrix in this size, we need about 1TB RAM to perform matrix multiplication, which is not practical.

As mentioned in Section 3.2.4, we leverage a subgraph strategy to minimize the RAM requirement. Our strategy is to only consider the set of nodes that will be potentially reachable after N steps, and generate a smaller adjacency matrix only for them. With this subgraph approach, the average number of seen nodes after 21 steps is 178,620, requiring 121.7GB RAM with Matlab single precision. This is still a large amount of memory. However, the matrices we make are highly sparse and we take advantage of Matlab sparse matrices. It reduces the required memory down to 47GB.

Parallel Threads Configuration Our system is easily parallelizable. To run the experiments efficiently in time, we run a thread for each topic separately. In each run, we visit a different subgraph, needing a large amount of RAM. For this reason, we used the Vienna Scientific Cluster configuration. The VSC clusters contain nodes with 32GB to 128GB RAM. Based on memory requirement, we configure different threads to run on nodes of the cluster.

Table A.1: Different categories of nodes in the graph

node type	*number*
doc	125,828
img	237,434
total	363,262

A.1 Matlab Matrix Sizes and Memory Requirement

In this section, we analyse different sub-graph sizes mapped to matrices in Matlab based on both single and double precision. The purpose is to find the limit on size of the sub-graph which can fit into memory to perform the matrix multiplications.

For any matrix size we need double of its size to fit into memory. The reason is that we have the matrix itself and also $b = b * a$ or $b = a^2$ where a is the matrix and b is the result of multiplications. In Table A.2, we show the memory needed in each case. In any case the memory should not be used up and we need about 10GB for the Java program itself to run.

For single matrices we need the memory of size of a double size matrix in addition to a single size matrix. The reason is that we cannot directly create a single matrix in Matlab. We should first create a double matrix and then apply *single* function on that matrix. Of course we can delete the double matrix after creating the single matrix, but this does not change the memory requirement.

Table A.2: Memory needed to create a matrix in Matlab. The available memory of our machine is 100GB. The real free memory is about 90GB. If we touch the border of available memory it needs to swap and performance will degrade substantially. We work with matrix sizes that need less than 90GB.

matrix size	**precision**	**memory**	**real need** (program requirement)	**total**	**practical**
50000	double	19GB	38GB + 10GB	48GB	yes
60000	double	27.5GB	55GB+ 10GB	65GB	yes
70000	double	38GB	75GB + 10GB	85GB	no
50000	single	9.5GB	19GB+19GB+ 10GB	48GB	yes
60000	single	14GB	27.5GB + 28GB+ 10GB	66GB	yes
70000	single	19GB	38GB+ 38GB+ 10GB	90GB	no

A.1.1 Matrix Multiplication

In this section we compare the time taken for double and single precision matrix multiplication for different matrix sizes. With double precision, the matrix of size 50,000

Table A.3: The time needed for one iteration (one matrix multiplication) in Matlab

matrix size	**precision**	**one iteration**
50,000	single	18 m
60,000	single	32 m
70,000	single	59 m
50,000	double	47.5 m
60,000	double	4h 37m

takes 47.5min for each multiplication (Table A.3)). This takes 3.125 days for 100 steps for just one topic, more than 5 months for all topics. Then we calculated the matrix multiplication with power operator. We compared the time needed to multiply iteratively with using mpower operator of Matlab (e.g. for m^4 we have $m \cdot m \cdot m \cdot m$ as iterative multiplication, and $mq = m \cdot m, mqq = mq \cdot mq$ using power operator). We use the *mpower* function of Matlab, which is an efficient way to multiply a matrix to itself.

Table A.4: Matlab matrix multiplications with different matrix sizes and operators

matrix size	steps	precision	**multiply** for one topic	**power** for one topic	50 topics (**iteratively**)	50 topics (with **mpower**)
50,000	50	**single**	15h	2h 24m	31d 5h	**5d**
60,000	50	**single**	1d 1h	4h 16m	52d 2h	**9d**
70,000	50	single	2d 2h	6h 37m	104d 4h	14d
50,000	50	**double**	1d 153h	5h 48m	83d 7h	**12d 2h**
60,000	50	double	9d 5m	20h 30m	450d 4h	42d 17h
50,000	100	single	1d 6h	2h 55m	62d 10h	6d
60,000	100	single	2d 2h	4h 31m	104d 4h	9d 11h
70,000	100	single	4d 4h	7h 23m	208d 8h	15d 7h
50,000	100	double	3d 3h	6h 40m	166d 14h	13d 18h
60,000	100	double	18d 10m	22h 18m	900d 8h	50d

The results for a sample topic and 50/100 steps are shown in Table A.4. As shown in the table, multiplications with single precision using *mpower* function is much faster than what we have for double precision, or in cases that we perform the multiplications iteratively. Therefore, we use the mpower to perform the multiplications. This way, we have the results in binary steps (2,4,8,16,...). In addition, we can reach high number of steps like 128 or 512 with only 6 or 7 times of multiplications. The pseudo code is shown as below:

```
n = number of iterations;
m = matrix;
a = activation_vec
for i = 1 to log(n) do
        z = mpower(m)
```

```
            res = a.z
           m = z
end
```

The question arises whether we lose precision in case of using matrices with single precision. Therefore, we verify the result of matrix multiplication of single and double precision values in the next section.

A.1.2 Results of Single and Double Multiplication

First, let us have a look at max/min values in Matlab for single and double precision. The MATLAB functions realmax and realmin return the maximum and minimum values that can be represented with the double data type: The range for double is:

```
    -1.79769e+308 to -2.22507e-308 and
     2.22507e-308 to  1.79769e+308
```

Numbers larger than realmax or smaller than -realmax are assigned the values of positive and negative infinity, respectively:

```
realmax + .0001e-308 = Inf

-realmax - .0001e-308 = -Inf
```

The MATLAB functions realmax and realmin, when called with the argument 'single', return the maximum and minimum values that you can represent with the single data type:

```
ans = The range for single is:
-3.40282e+38 to -1.17549e-38 and
 1.17549e-38 to  3.40282e+38
```

In the single precision interval (1.17549e-38 to 3.40282e+38), the results of multiplication with single or double precision are completely the same. We tested with Matlab *rand* function. For the numbers bigger than this interval it goes to Infinity value and create *Inf* values in the matrix. For numbers less than the min value, it rounds to 0.

A.1.3 Test in Astera

To analyse what happens in practice, we performed the real matrix multiplications in Astera. These matrices are not normalized and they are generated as before. We compare the result of using Matlab file with double precision matrices with the result of single precision matrices. In small number of iterations, the results (the final calculated precision) are the same. When it reaches higher multiplications, it may cross the margins. Once it happened in the 88th step; as in step 87 all numbers were greater than $1.0e + 38$.

In the next iteration, some cells had Inf value, since it goes beyond the single max value. Another time it happened in the 110th step - depending on the topicID that we start from, we have different sub-graphs and matrices. We cannot identify a step number that all matrices from different topics will cross the upper margin. This happens since we generate energy, and this value explodes in larger steps. We did not cross the lowest margin, as in the case of Spreading Activation, the generated value is always getting larger and larger.

A.2 Conclusion

The goal of this appendix is to find the maximum size of the matrix that can fit into memory, meanwhile to find a method to perform the matrix multiplications faster. We considered different matrix sizes, two multiplication methods (iteratively and using Matlab power function), with single and double precisions in Matlab. The findings are listed as follows:

- **Matrix size**: for 60,000 and 70,000 graph size, both single and double matrices fit into our memory.
- **Matrix precision/Time needed**: with single precision the time needed for power multiplications are acceptable. For double precision and also iterative multiplications the time is too long. Therefore, we create matrices with single precision.
- **Snapshots to the iterations**: we will have the results only in 2nd, 4th, 8th, 2^nth steps.

APPENDIX B

Astera Software Architecture Document (SAD)

B.1 Introduction

Astera is a generic model for multimodal information retrieval[1]. Finding useful information from large multimodal document collections such as the WWW is one of the major challenges of Information Retrieval (IR). The many sources of information now available - text, images, audio, video and more - increases the need for multimodal search. Particularly important is also the recognition, that each information item is inherently multimodal (i.e. has aspects in its information character that stem from different modalities) and forms part of a networked set of related information items. We proposed a model for multimodal IR named Astera. This model works on multimodal collections (like image, patent or medical collection). This model is under test with ImageCLEF 2011 multimodal collection. The system design is combined of a core which performs the main functionality and plug-ins. These plug-ins can be in different categories like different indexers, parsers, for new features.

This appendix provides a comprehensive architectural view of the system. It contains different architectural views to picture different aspects of the Astera system. It is envisioned to convey the important architectural decisions, which are made on the system. These are views on an underlying Unified Modeling Language (UML) model developed using Rational Rose.

---

[1] Astera is open-source and available at http://www.ifs.tuwien.ac.at/∼sabetghadam/Astera.html

B.2 Architectural Representation

This document presents the Astera architecture as a series of views: logical, process, deployment, data, and implementation.

B.3 Logical View

The Astera architecture follows the principle of three layer architecture design. It consists of presentation, business and data layer (Figure B.1).

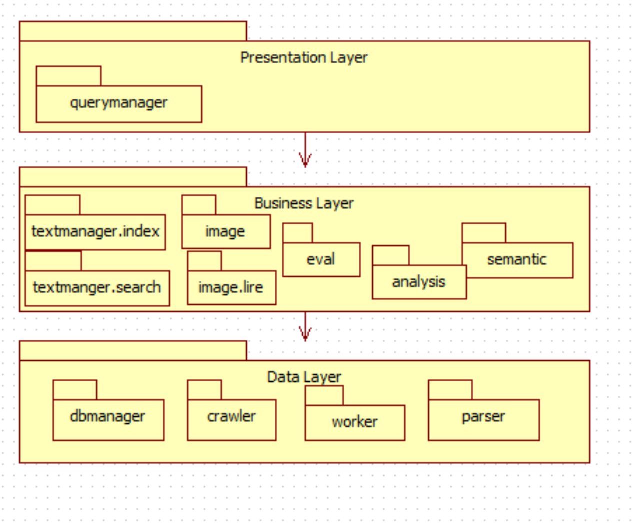


Figure B.1: Astera three layer architecture

B.3.1 Presentation layer

This layer is the interface with the user. It receives the query from the user and passes it to the business layer. After the query is processed and results are ready, it shows the results back to the user. The only package in this layer is *querymanager*.

B.3.2 Business layer

As in the three layer model of architecture, this layer interacts with the presentation and the data layer. It contains packages for managing the logic of the system, as follows:

- textmanager.index: Indexing of all the data is performed in this package. We can index multiple collections through classes of this package, e.g., a text collection or a metadata collection.
- textmanager.Search: This package receives the query and performs the standard search task. We have separate classes for each indexed collection.

- image: We perform image similarity computations in this package. Similarity computation can be based on the facets provided by the collection for both query and information objects. We create a query based on available facets from the collection, and compute similarity with the same facet of all information objects in the collection (managed through image/*ImageSimilarityManager.java* class). In addition, it can be performed by standard search engines such as LIRE for image similarity (managed through *image/lire/Searcher.java* class).
- image.lire: We handle the image search task by LIRE in this package.
- eval: This package conveys the logic of different graph traversal methods and calculation of the final result.
- semantic: This package manages the data retrieved from DBpedia.
- analysis: This package holds all the logic needed for preparing the output files of recall analysis.

B.3.3 Data Layer

We read the information about different types of information objects and their relationships in this layer. This procedure may differ for each new collection. After the metadata is parsed, we save the information objects in the database. The design is independent of underlying database type, which could be a graph DB like *Neo4j* or *Jena*, or other types of databases like MySQL. Packages in this layer are:

- crawler: This package contains the main class of *Crawler.java*. This class runs a thread which is responsible for reading data from different resources such as file or database. Crawler is responsible to work with different types of data and parse them to add to the Astera database. It gets help from Parser package to parse different file types, and then adds them to a queue.
- parser: This package is responsible for parsing different types of input files. For example, the information object can be in *csv* format file, or list of feature values in the format of *arff*. In addition, a new file format with its defined parser can be added to the system.
- worker: This package is responsible for getting the parsed ready data from queued data and add them to the graph database.
- dbmanager: This package is responsible for interaction with the database. It provides generic interface for business layer classes. For any new database, we should implement this generic interface to be able to interact with business layer. In the current version, we have implemented the interface to work with the Neo4j graph database.

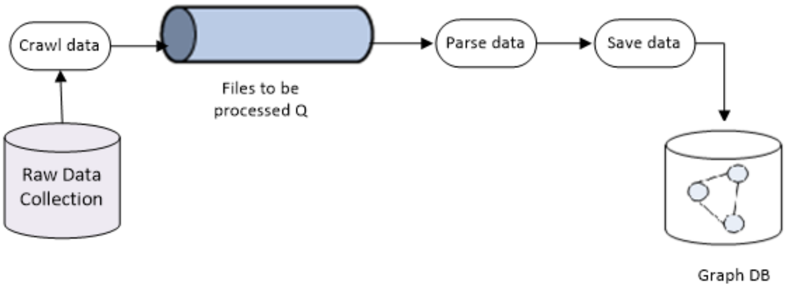


Figure B.2: Crawling data, parsing and saving in the Graph DB. These stages are managed through packages of crawler, worker, parser and data.

- data.dbpedia: This package is responsible for interaction with DBpedia for fetching related information and adding semantic links.

B.4 Process View

The process view describes the processes and threads involved in the execution of the system, their interaction and configuration (Figure B.3).

B.4.1 Data Crawling Process

The tasks of crawling and handing data to be saved in the database are handled in *crawler*, *worker*, and *parser* packages. Raw data for Astera can be in any format, as long as the appropriate parser is defined. As is shown in Figure B.2, the data is fetched (crawled) from a pre-specified folder address. Data can be in different files or be read from database. The files are read and put in a queue for the *data/Worker.java* class to fetch and process. Whenever a file is dequeued, it is given to *data/parser/ParserProvider.java* class to find the appropriate parser for this file according to its extension. As is shown in the configuration part (Table B.2), there is a configuration variable for defining parsers. All parsers should be defined in this field. If we receive a file with a new extension, the new file type should be added to the *ext* parameter (Table B.2). After a data file is parsed, it is given to an instantiation of *IDBManager.java* interface to store in the database.

B.4.2 Query Process

In order to test the system, query information is read from the test collection. This is managed in the Query package by *QueryManager.java* class. We create queries from the test collection through *QueryFactory.java.* This class reads the query facet files and creates a query object out of them..

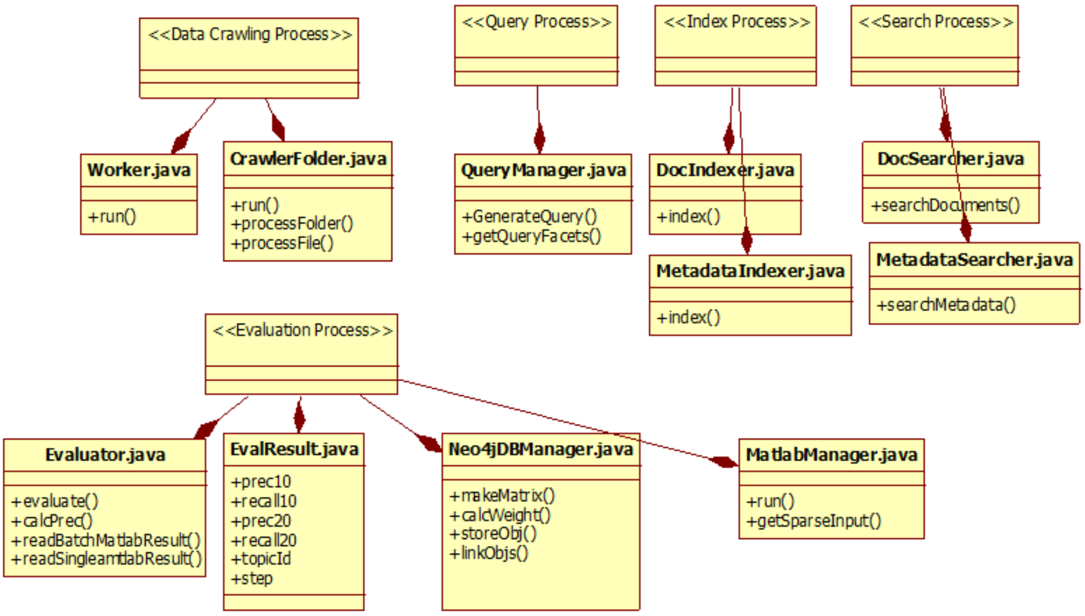


Figure B.3: Different processes in Astera

B.4.3 Indexing Process

We create two types of indexes in Astera: Standard indexing and Graph indexing. Any collection that is given to the system as input is indexed by standard search engines like Lucene. Afterwards when the query is given or parsed from the given query files, *QueryManager.java* gives the query objects to the *TextSearchManager.java*, and provides the first level results of information objects.

We use different facets of the query modalities to perform standard indexing. For example, if the query is a combination of text and image, we can use textual facets of the keywords and query image example tags, and visual facets of the image example. The result is a ranked list of information objects. We use Lucene for textual facets (TF.IDF, BM25, LM), and LIRE for visual facets.

To provide the graph indexing of the data, the data collection is crawled. The relation between information objects is created based on the given metadata. As a result, we obtain a relational indexing of the given data.

B.4.4 Search Process

The search procedure in Astera is composed of two steps of standard search and graph search. First, the top results of the standard search (from any defined facet) form the starting points in the graph. Second, based on the result list, the starting points in the graph is determined and the graph traversal begins. In each traversal step, we fetch the neighbours of current nodes with the weights on edges. We create a matrix out of visited nodes and edges. Each step in the graph is simulated as a matrix multiplication. The result vector in each step is calculated based on Equation 2.17, which we repeat here:

$$a^{(t)} = a^{(0)} \cdot W^{t} \tag{B.1}$$

where $a^{(0)}$ is the initial score vector, $W^{(t)}$ is the matrix in step t, and $a^{(t)}$ is the result vector. We receive a ranked list in each step based on the node scores in this step ($a^{(t)}$).

For the specific collection of ImageCLEF2011 Wikipedia, we have the task of image retrieval. We filter the images in each step and calculate the precision. The final ranked list is checked with the ground truth data and precision and recall is calculated. This filtering may differ for other collections based on the retrieval task.

B.4.5 Evaluation Process

The evaluation task in Astera, starts with receiving the top ranked nodes from the search result. The *Evaluator.evaluate()* method calls *Neo4jDBManager.makeMatrix()* to create the output files needed for matrix multiplication (Figure B.3). Then, it calls *MatlabManger.run()* to calculate the result of each step. Finally, it calls *Evaluator.calcPrec()* to calculate the performance of this run in different steps.

B.5 Implementation View

The design of Astera to support different collections and combinations of search procedures is shown in Figure B.4. As shown, a query is received from the user, or from some input files in the case of a test collection data. The QueryManager module provides the query objects, and delegates to the Indexed Search Manager module, which provides the first level results. The result is given to the Evaluator module to perform the search through Graph Search Manager module. This module provides the result for Ranking step which provides the final output result.

B.5.1 Important Classes

Here, we list some important classes in Astera.

AsteraManager.java This is the starting class which calls different configurations to run. The primary.properties is called through this class, from which we read the configuration of different runs. Different configurations are loaded from this class to run in each iteration. We check first if it is an index, search, or evaluate run and call the correspondent methods from *business/Evaluator.java*.

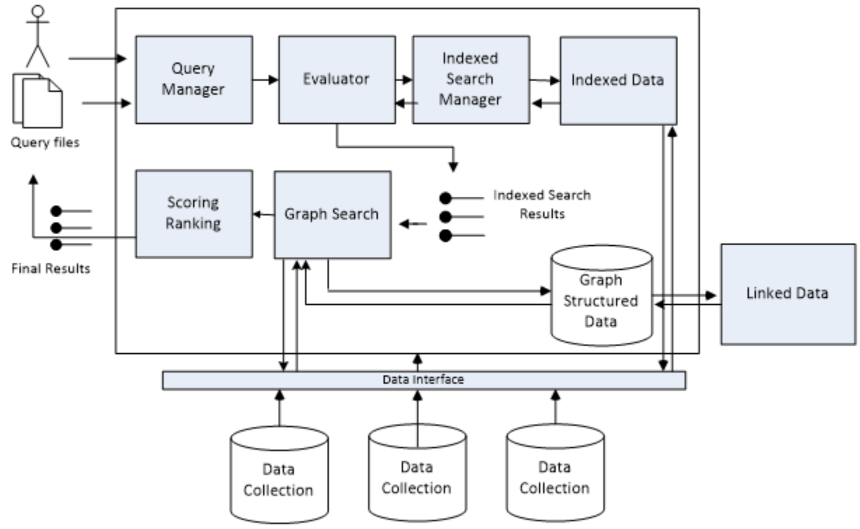


Figure B.4: Design of Astera architecture

Table B.1: Some methods of *Neo4jDBmanager.java*

Method	**Description**
saveInfObj()	This method saves an information object with its properties in the graph DB. This information object can be a document, an image or any other type of information object.
addRelationship(infObj, infObj)	This method adds relationship with specific type (α, β, γ, δ) between two information objects.

business/Evaluator.java This class is responsible for performing the evaluation. It starts from the top results of the standard search to traverse the graph. It calls *Neo4jDBManager.makeMatrix()* method to perform the traversal to the number of defined steps in the configuration of this run. Then, it calls Matlab function via *MatlabManager.run()* to perform the matrix multiplications. It reads the result vector files (row 7 of Table B.12), calculates the precision, and saves in the database.

data/Neo4jDBManager This class acts as an interface to the *Neo4j* graph database. It traverses the graph and generates the output files needed in Matlab (rows 1 to 6 in Table B.12). All methods needed to save node information and relationships to the neighbours are included in this class. Some of the methods are listed in Table B.1

data/Neoj4QueryManager This class contains libraries and methods to query directly the *Neo4j* graph database. For example, if we want to check the number of semantic

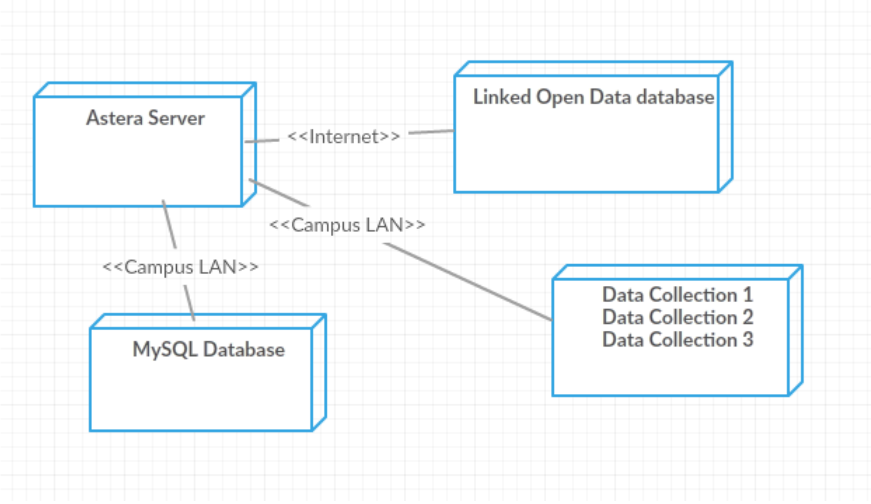


Figure B.5: Deployment diagram of Astera

or similarity links in the graph database, we can simply write a query and call it via *execQueryManyCol()* method of this class.

B.6 Deployment View

Here, we show how to run Astera as a standalone application. The design of Astera allows to run it on multiple servers as well. As shown in Figure B.5, the Astera program is deployed on an Astera server. Different data collections are read and processed through this system. External (semantic) sources such as Linked Open Data can be used to enrich the collection. Evaluation results are saved into a MySQL database.

B.7 Data View

We design database tables to store the required information (Table B.3). To speed up the evaluation process for different runs, we store the search results in the Evaluation table. We keep the similarity information of different information objects in different tables (e.g. we keep the similarity between documents in Doc2doc table). In the current version, we have documents and images, and keep their similar results in different tables. The final evaluation result (performance of the system) is saved in the result table.

B.8 Configuration

Astera is generic in its basic architecture, however equipped with a flexible configuration mechanism to support collection-specific evaluation tasks. Astera's configuration is divided into two levels of configuration:

Table B.2: Configuration variables in secondary.properties file

Variable	**Description**
do.index	true to build the indexes, false otherwise
do.search	true to perform the search, false otherwise
do.evaluate	true to evaluate the results, false otherwise
add.semantic.relation	true, if linking to DBPedia is included.
facets	Astera can work with any number of facets. When a new facet is added to the system, its name should be added here. In this case when system starts up, reads the allowed facets like CEDD and TF.IDF.
parser.arff	New features/facets may be encoded in different way. Either the encoded format of this feature is parsable by existing parsers, or a new parser should be added to the system. When Astera reads a facet file, it asks for appropriate parser from the *ParserProvider.java* class. When a new parser e.g for *arff* file is added, the line in the configuration file should be like this: Parser.arff = data.parser.Arffparser which adds an *arff* parser to the system. The same scenario holds for , or Parser.csv = data.parser.CSVParser.
exts	Accepted extensions of different files like txt, arff, csv.
text.weight img.weight meta.data.weight	In order to give different weights to different types of nodes, related parameters are defined. The value given is between 0 and 1 and also the sum of them should not be more than 1.
prec.at	To determine the precision@ value, e.g. 10, 20.
img.on	True if image similarity is included in the ranking procedure, false otherwise
clean.eval.db	True if the evaluation DB should be cleaned, false otherwise
db.type	Database could be of any type, graph, file or sql. In this parameter the exact type of the DB is determined, which for example Neo4j is used in the current version.
similarity.function	The method to compute the similarity between vectors
normalize.start.range normalize.end.range	Normalization start and end range values
doc.dir meta.dir doc.index.dir meta.index.dir	Addresses of the directory paths
topics.file	The collection topics file path, which includes queries
ground.truth.file	The collection ground truth file

Table B.3: Database tables

Table	Description
Evaluation	Top similar document results of the standard search engine, Lucene
Doc2doc	Similarity information between documents
Img2img	Similarity information between images
LIREimgsimiinfo	Top similar image results from LIRE
Result	The evaluation result and graph performance in different steps

1. Primary Configuration: Astera requires basic settings that are needed to generally operate the system independent from what is done specifically with respect to collection(s). Such basic settings are kept in the primary configuration file called "primary.properties". An example for such primary properties are the database settings to store Astera-wide evaluation results.

2. Secondary Configuration: Astera also manages additional configurations that is optional and specific to particular collections and particular evaluations on these collections. These secondary configurations are linked with the primary configuration file. Multiple secondary configurations can be listed and thus chained into sequences. This introduces a range of flexibilities to the system:

    - Specificity: Configurations can be made specific to collections, evaluations, researchers and tasks and can define different qualities (e.g. testing versus production-level).
    - Partitioning: Configuration settings can be clearly separated from each other and exchanged easily without touching the logic of the source code.
    - Batch Mode and Workflows: Astera performs in batch mode on its configurations and executes a configuration as a so-called "Run". Multiple secondary configurations can be listed which means that Astera chains them into workflows and executes them one after another.

B.8.1 Design of configuration files

Figure B.6 shows an example configuration infrastructure within Astera that consists of five configuration files. The primary configuration provides the basic settings (e.g. information of how to connect to the Astera evaluation database). It either triggers two test runs (e.g. trying out a experimental weighting scheme on two local collections) or two production runs (e.g. two search evaluations on a collection that are compared for a publication).

These features enable that every collection can be configured with Astera and that available tools can be adapted to process these collections. Furthermore, it allows creating sequences of runs each operating with multiple collections and alternative settings to test different variables in the hypotheses space. The list of runs can be

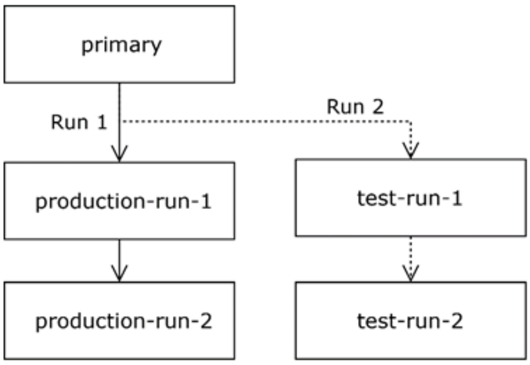


Figure B.6: The configuration of primary and secondary runs in Astera

arbitrarily long thus enabling to design and run complex research agendas in batch mode. Each configuration contains all decisions that have been made in addition to the Astera code. This means, each configuration file is a descriptive research document that can be useful for later and it worth keeping. In the following two sections, we highlight the internal structure of the two levels of configuration — the primary and the secondary configuration file.

B.8.2 Structure of the Primary

In the configuration design of Astera, the primary and secondary configurations are separate now and we can specify different second configuration for different collections. These configurations can be run separately. The step to run them in parallel is in the next design steps of the system. The primary configuration (filename: “primary.properties”) contains two types of information:

- A set of variables that define the absolute core of the Astera system configuration the is required to run its basic functionality.
- A list of secondary configurations as an IR Evaluation Process for a unit of research (e.g. a paper, deliverable or even a test procedure). Each secondary configuration file is also called a “Run”.

The following list (Table B.4) shows variables that are part of the primary with a description and its default: The variable “runs.properties” can list a number of additional secondary configuration files. This list represents the IR Evaluation batch process. Each configuration file from this list is loaded in isolation from each other, in that sequence, and executed as a run. Each run loads its variables on top of the variables defined in the primary. When switching to the next run, the variables from the previous run are removed which means that each run is executed in its own environment.

Table B.4: Configuration variables in primary.properties file

Variable	Description	Default
run.properties	Ordered list of file names with run configurations	secondary.properties e.g. run1.properties, run2.properties
driver	JDBC Database driver for MySQL with reference to Astera database name	jdbc:mysql://localhost:3306/astera
user	Astera database user	root
pass	Astera database password. Must be set.	N/A
clean.eval.db	When set true, removes everything for evaluation DB	true

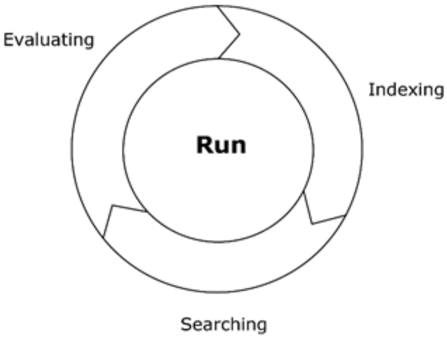


Figure B.7: Run cycle in Astera: Each collection is indexed, searched, and then evaluated

B.8.3 Structure of a Secondary (Run)

Each secondary configuration is a potential run. It is potential in the sense that it only becomes a run when listed in the primary configuration. A run is a single iteration through the three stages of indexing, searching and evaluating (Figure B.7):

- Indexing: This configures the indexing fields and where the content for this fields resides (e.g. extracting the authors of documents with a XPath statement and putting it in a field called “author”). This configuration in this section highly depends on the collection.
- Searching: This part determines which queries are used for the search and how they are generated, how they are processed, how many results are extracted per query, and the scoring algorithms used.

Table B.5: Different configurations in Astera to index a collection or run the search and evaluate the indexed collection

id	Description	Indexing	Searching	Evaluating
1	Pure indexing: For testing or preparing an index	X		
2	Pure searching: For testing a search algorithm on an existing index		X	
3	Pure evaluating: For testing an evaluation algorithm on an existing index, and use searched result			X
4	Evaluated Search: Allows evaluating a search algorithm on an existing index		X	X
5	Index and search: For testing a search on a newly created index	X	X	
6	Full Run: A full IR evaluation from scratch	X	X	X

- Evaluating: This section defines a set of measures and their parametrization. These depend highly on the kind of research that is created.

The cycle of all three stages represents a complete IR evaluation. However, it is also possible to disable some of its parts. Table B.5 shows all sensible combinations that are supported within Astera:

Most arrangements are used when developing or testing a search strategy with a new collection. However, the arrangements without indexing (2 and 4) are generally useful when a complex index is previously built, to save system resources. The following table shows the variables that are required for a secondary configuration. Some parameters are specific for CLEF2011 collection as a sample collection like the address of the directories.

The class property points to the full path of a Java class that is used for this run. It allows connecting the execution with Java code that understands additional properties on top of the basic ones listed above. The three properties *do.index*, *do.search* and *do.evaluate* can be used to control the evaluation cycle.

B.9 Similarity Links

We define the classes and configuration variables involved in adding similarity links between information objects. To add similarity links, first we find similar information objects based on the defined facets. For instance if we define the text.facet = LM, we find similar information objects for each document in the collection. Then we add similarity links between these similar objects. Related configuration variables are listed in Table B.6.

Table B.6: Configuration variables related to similarity links in the graph DB

Field	Description
Similiarity_on	True, if we include similarity links in the traversal, false otherwise
SimilarityNo	The maximum number of similar neighbours to be added for an information object
Weight_threshold	If the weight on a link (the similarity value of two nodes) is higher than this value we create the similarity link
similarity.function	The type of the similarity function like Cosine.

Table B.7: Configuration variables related to semantic links in the graph DB

Field	Description
Semantic_on	If we include semantic links in the traversal
Semantic_weight	If we set a default weight for semantic links, this field can have a value between (0,1)
Allowed_semantic_links	To put constraints in the traversal and limit the semantic links which are allowed to pass through
Not_allowed_semantic_links	To put constraints in the traversal and limit the semantic links which are not allowed to pass through

We can use different functions such as Cosine or Euclidean distance for similarity computation. Both of these methods are implemented in the current version of Astera. The configuration variable *similarity.function* determines which method to be used. The result of similarity computations are normalized to combine different facets as one single score of a node.

We can weigh the effect of each facet in the final score of a node. This weighing can be used or defined via configuration variables. We have defined text.weight, img.weight and meta.data.weight parameters for weights.

B.10 Semantic Links

Astera has the possibility to enrich the collection by adding semantic links. We can add semantic links with the help of external sources such as Linked Open Data. For example, as we work with the Wikipedia collection in this thesis, we use the corresponding DBpeida dump to add semantic links. We read all semantic information in the format of <subject, predicate, object> from the DBpedia dump. For each, we check if both subject and object are in the collection. If so, we add the corresponding semantic links. The semantic links are added through *Neo4jDBManger.addRelation()* method with relation type α

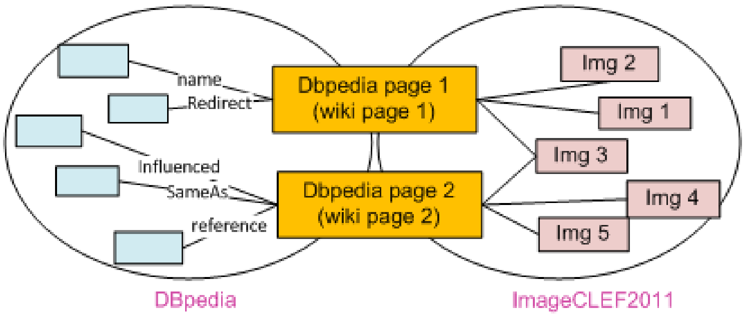


Figure B.8: We add sameAs links between the Wikipedia pages that have correspondent relation in DBPedia

Table B.8: Fields needed to be changed if we change the traversal method. The first field exists only in the Astera_randomWalks branch.

Field	**Description**	**Sample value**
do.metropolis.hastings	If we set the traversal method to Metropolis-Hastings	true
node.id.list.file	This is a sample configuration variable that changes if we change the traversal method. This file contains the visited node Ids.	output_MH/NodeIdListFile.txt output_RW/NodeIdListFile.txt output_SA/NodeIdListFile.txt
nodes.no.file	This is another sample variable that should change, which logs the number of the nodes seen in the traversal.	output_MH/NodesNoFile.txt output_RW/NodesNoFile.txt output_SA/NodesNoFile.txt

B.11 Implementation Strategies

B.11.1 Different Traversal Methods Configuration

We can configure three different traversal methods Spreading Activation, Random Walks, or Metropolis-Hastings in Astera. We use Spreading Activation when we set our defined weighting on the edges, or apply constraints on the traversal. We use Random Walks or Metropolis-Hastings settings in the configuration file when we want to see the graph behaviour in stationary distribution. With these two methods, the weighting on the edges satisfy the stochastic property—we use the weighting of $1/N$ on all edges of a node. Metropolis-Hastings methods is applied based on the algorithm described in Section 2.1.4. In each of these methods, a different weighting method is used. This decision is made in the Neo4jDBManager.calcWeight() function. The output of any of these methods are saved differently. The path to save the results is changed accordingly in the configuration file (secondary.properties). We show sample configuration variables in Table B.8 for Metropolis-Hastings algorithm.

B.11.2 Different Implementation Branches

In the current implementation, we have three implementation branches of Astera. Two branches are based on the traversal methods, and one branch is for doing the recall analysis part. The details of each branch are as follows:

Astera Spreading Activation The setting and weight definition on edges for Astera with Spreading Activation traversal method are located in this branch. Two projects are created here: Astera_breadthFirst and Astera_depthFirst. In Astera_breadthFirst, we go through all topics in each step, and accordingly calculate the performance. In Astera_depthFirst project (current version of Astera for Spreading Activation method), we go through all steps for each topic. This way, we calculate the performance in all steps for each topic. The three phases of interaction between Java and Matlab program (Section B.11.3) is implemented in this project. Further, we can run different threads for each topic to run in Parallel. We do not have any configuration variable to set the traversal method in *secondary.properties*, as the default method here is Spreading Activation.

Astera Random Walks This implementation branch is based on Random Walks method for graph traversal. In addition, the configuration and weighting method for Metropolis-Hastings goes into this branch. The reason is that Metropolis-Hastings is a version of Random Walks with different weighting method during traversal. As shown in Table B.8, in the Astera_randomWalks path, we add a configuration variable (*do.metropolis.hastings*) to set the traversal method. If it is set to *false*, Random Walks as the default method in this branch is used.

AsteraAnalyser Branch We created another branch in the implementation for recall and reachabiltiy analysis. Two of the important classes in this branch are *business/eval/DistManager.java* and *business/eval/DistResult.java*. The class *business/eval/DistManager.java* is responsible for calculating recall in each step for each topic. Important methods of this class are listed in Table B.10. Output of the DistManager.logRelImgs() method is the recall analysis information in each step for different topics. The output files are shown in Table B.9.

In our recall analysis, we can configure Astera to start from different facet results or consider semantic or similarity links and calculate recall in each step. Another part of this analysis is to check if we visit the same (relevant) information objects if we obtain the same recall of the facets. This analysis is performed in *DistManager.java* class. Two important methods of this class are shown in Table B.10. For each facet, we check the relevant information objects and calculate the ratio of the nodes visible only to a specific facet. This calculation is based on Equation 4.2. The output files of this analysis are shown in Table B.11.

Table B.9: Output files of the recall analysis in AsteraAnalyser branch, saved in output/-analysis/ path

Output	**File path**	**Description**
recall.log.file	recallLogFile.txt	Logs the recall in each step for each topic
step.rel.count.file	stepRelCount.txt	The number of relevant information objects in each step for each topic
total.rel.count.file	totalRelCount.txt	The cumulative number of relevant information objects in each step for each topic
img.seen.file	imgSeen.txt	Log of the image IDs seen in each step for each topic
rel.img.seen.file	relImgsSeen.txt	Log of relevant image IDs seen in each step for each topic

Table B.10: Important methods of DistManger.java class

Method	**Description**
logRelImgs	This method checks the number of relevant nodes in each step and logs it.
logRatioDiffFacets	This method logs the ratio of information objects seen for different textual facets.
logRatioDiffFacetsDocImg	We log the ratio of information objects seen for different visual facets. This method specifically compares different images seen for a doc and a img facet.

Table B.11: Graph visit configuration variables and output files in the secondary.properties, save in output/analysis path

Variable	**Value**	**Description**
cedd.step.seen.img.file	stepSeenImg.txt	Log of the image Ids seen from CEDD facet in each step. This type of output file is generated for each facet separately.
cedd.ratio.file	ceddRatio.out	Log of the ratio of relevant images visible only by CEDD facet compared to other visual facets. This type of output file is generated for each facet separately.

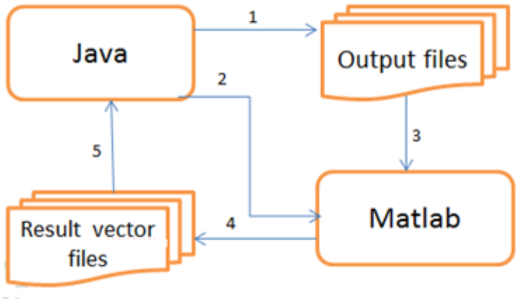


Figure B.9: Data flow between Java and Matlab programs

B.11.3 Java and Matlab

As mentioned in Section B.5, we call a Matlab program from Java to perform matrix multiplications. In this procedure, we design a three phase implementation (Figure B.9). In the first phase (in the Java program), we traverse the graph from starting points of a topic to the number of defined steps. The files needed for the Matlab program to do the matrix multiplications are generated. In practice, we visit a sub-graph of the whole graph. The output files shown in Figure B.9 are shown in Table B.12.

The nodes and the edge weights are logged in an output file like MatlabGraphWeightLogFile.txt-83. One sample line in this file is like: 1,2,0.5,3,0.5: node with ID 1 has neighbours node 2 with weight 0.5 on the link to this neighbour; the same holds for neighbour node 3. This is a map of the part of the graph seen. This map will be used as input of the Matlab file to perform Matlab multiplications (step 3 in Figure B.9). The activation vector consists of all nodes initial scores. We use the score of the starting nodes, which are the result of standard search (e.g. Lucene search), plus zero value for other nodes in this vector.

In the second phase, the row.txt-83, col.txt-83, and val.txt-83 are loaded in Matlab to create the sparse matrix (step 3). We perform the matrix multiplications in Matlab based on the Equation 2.15. The $a^{(0)}$ is created based on the NodesLogFile.txt-83 file. In each step, the $a^{(t)}$ result is calculated and saved in the result vector file Res-s*step*-t*topicid*.txt, e.g. Res-s2-t83.txt in step 2 for topicid 83.

In the third phase, we read the result vector file. This is performed in the *business/Evaluator.java* class, *readBatchResult()* method. To calculate the precision in each step, we need the correspondent result vector files from Matlab, and the *nodeIdList* file to map the result scores to the node Ids in the graph. We filter the images from this result map and calculate the recall and precision.

Different settings for small and large graphs If the generated graph size fits the memory size of the Java program, we call Matlab program from Java and all these

Table B.12: The output files from Java or Matlab program. All files are saved in the output folder, e.g., output/res_vec/res-s2-t77.txt.

File name	**Description**	**Generated from**
GraphWeightLogFile.txt-*topicid*	Log of each node seen and its neighbors and the edge weight to the noighbours, e.g., 335748,I-164945, 1.0, I-116763,1.0, means that node Id 335748 has two neighbours of I-164945 and I-116763, each with weight of 1.0. This file is created for a specific topic *topicid*.	Java program
MatlabGraphWeightLogFile.txt-*topicid*	Input for Matlab program. Log of each node seen and its neighbors and the edge weight to the neighbours. The nodeIds start from 1 in this file for simplicity in the Matlab program, e.g., 1,2,1.0,3,1.0, means that node Id 1 has two neighbours of 2 and 3, each with weight of 1.0. This file is created for a specific topic *topicid*.	Java program
col.txt-*topicid*	Information about the values of the matrix of the visited graph seen for a specific topic. This file is needed for creating a sparse matrix in Matlab.	Java program
row.txt-*topicid*	Information about the rows of the matrix of the visited graph for a specific topic. This file is needed for creating a sparse matrix in Matlab.	Java program
val.txt-*topicid*	Information about the columns of the matrix of the visited graph for a specific topic. This file is needed for creating a sparse matrix in Matlab.	Java program
NodesLogFile.txt-*topicid*	This file contains the initial score of all the nodes for a specific topic	Java program
nodeId_list/NodeIdList-t*topicid*.txt	This file contains the nodeId (e.g. 345000, or I-547778) of all the nodes for a specific topic.	Java program
res_vec/Res-s*step*-t*topicid*.txt	This file contains the results from the Matlab program in step *s* for a specific topic *t*. It contains the scores of all nodes in this step of multiplication. The score order is mapped to the correspondent NodeIdList file. For example, NodeIdList-t83.txt is mapped to res-s2-t83.txt.	Matlab program

three phases are performed in one thread of implementation. In this case, we set the configuration variable *matlab.on=true* in the secondary.properties. The Matlab function is called from the *business/Evaluator.java* class. In larger steps, e.g. after step 30, we visit a large part of the graph leading to large matrices. As we call Matlab from Java, the Matlab Java library reads the graph output files and transfers the data to Matlab via marshaling.

When the matrix size is larger than the memory limit of Java program, we have problem with transferring large matrices. When the graph size mapped to a Matlab matrix (requires nodesize * nodesize * 8 bytes) exceeds the memory limit of the Java program, then the marshaling cannot be performed successfully. In this case, we perform the three phases in three steps, separately. In the first phase, the output result files are written to disk (files 1 to 7 in Table B.12). In the second phase, we do not call Matlab function from Java, but call it directly from the Matlab program. This way, the Matlab program uses the system memory directly. The result of this phase is the result vector files (file row 8 in Table B.12). In the third phase, we call the *Evaluator.readBatchResult()* to read the results, and then *Evaluator.saveResults()* to save in DB.

B.11.4 Matlab Files

In this section, we list the Matlab functions called from Java, or called directly from a function in Matlab environment (Table B.13). To handle the memory limit with large matrices, we use the sparse matrix option in Matlab. In addition to speed up later computations, we use Mat files in Matlab to save snapshots from the matrices in different steps. Later, we can load this Mat files in memory and save the time needed to reach this step of multiplication.

B.11.5 Executing Astera

There is a bash file named *run.bat* in the *src* directory. This file compiles the project, and runs *AsteraManager.java* class loading *primary.properties* file. The run(s) to be called respectively are configured in *primary.properties* file in *runs* variable. There is another bash file, named *run_backup.bat* in the *src* directory, which is responsible to perform backup process of output files generated. This file is called as the last step in Astera project before the last run finishes.

B.11.6 Mapping different collections

First we need the path of the files of a new collection to be indexed. Second is to add the information objects and their defined relationships from the collection to the graph DB. This is performed via *data/Crawler.java*, *data/Worker.java*, and *data/Neo4jDBManager.java*. The features of the query and information objects can be provided by the collection in e.g. *csv* formats, or we can extract features. If the collection provides a new feature format, there is the possibility to define its own parser and set it in the configuration file in our model. Astera loads that specific parser, when it reaches files with the defined suffix.

Table B.13: Matlab functions to be called from Java or Matlab in a run of Astera

Id	Function name	Description
1	calcMatrix1050.m	This function reads the output files from java, creates the matrix and performs the multiplications. This function works on full matrix rather than sparse matrices. This was to work with Mat files in Matlab.
2	calcMatrix1050fullmatrix.m	The same as the function in row 1, This function works on full matrix rather than sparse matrices. This was to work with Mat files in Matlab.
3	calcMatrix1050NoMemory.m	The same as the function in row 1, we use the normal Spreading Activation algorithm, which does not use any memory.
4	calcMatrix1050WithMemory.m	The same as the function in row 1, this time we use the Spreading Activation algorithm with memory based on the algorithm described in Section 2.1.3.
5	runMatlab.m	As mentioned in Section 2.1.1, in case we run the algorithm in three phases. This function reads the Java output files for defined topics in the function and calls one of the functions from row 1, 2, or 3.
6	calcMatrixFullMatrixWithMatRW.m	The same as the function in row 1, This function works on full matrices, and also saves the result of matrix multiplications in each step in Mat files. Paths are customized for Random Walks run and output location.
7	calcMatrixFullMatrixWithMatMH.m	The same as the function in row 1, This function works on full matrices, and also saves the result of matrix multiplications in each step in Mat files. Paths are customized for Metropolis-Hastings run and output location.
8	runMatlab70000rw.m	Manages to load the results with Random Walks traversal from Java output and then call calcMatrixFullMatrixWithMat
9	runMatlab70000metHas.m	Manages to load the results with Metropolis-Hastings traversal from Java output and then call calcMatrixFullMatrixWithMatMH

Table B.14: Technologies used in Astera

Programming language	Java
Indexing library	Lucene & LIRE
Graph database	Neo4j
Semantic relation library	Jena
Database for evaluation data	MySQL

B.11.7 Technologies

Different technologies used in Astera are listed in Table B.14.

List of Figures

2.1 The extended structural poly-representation continuum ([LI05]) 32

3.1 Different types of relations in the model 37
3.2 Topics categories based on their difficulty [TPK11] 41

4.1 Overall recall aggregated per category over steps 48
4.2 Recall with different document or image facets 50
4.3 Average recall obtained compared to the percentage of the graph seen through different fact combinations of R_D = TF.IDF$_D$, R_{IV} = CEDD, and R_{IT} =TF.IDF$_I$ 52
4.4 Recall comparison of combination of document textual facets with image textual facets with R_D-R_{IV}-R_{IT} (TF.IDF$_D$, CEDD, TF.IDF$_I$) combination . 52
4.5 Average recall of all topics after 40 steps with different links 53
4.6 Comparing the effect of adding semantic links and random semantic links on recall and precision loss in the graph 54
4.7 Comparing the effect of adding similarity links and random similarity links on recall and precision loss in the graph 56
4.8 Average Recall in the base graph on different categories of topics 59
4.9 Average Recall in the graph with semantic links on different categories of topics 61
4.10 Average Recall in the graph with similarity links on different categories of topics 62
4.11 Number of new seen nodes per step in the collection 63
4.12 Ratio of different nodes visited from document textual facets 65
4.13 Ratio of different nodes visited from image textual facets 66
4.14 Ratio of different nodes visited from image visual facets 66

5.1 Bipartite graph, in odd steps we visit only images and in even step only documents 72
5.2 Prec@10 using TF.IDF$_{\mathrm{D}}$, CEDD, not weighted 73
5.3 In the second step with $st = 0$, reachable images are the ones through `sameAs` links 76
5.4 Two examples of following inter-lingual semantic link 77
5.5 Score distribution in the graph in different steps for topic 83. The x axis is all the nodes in the graph which is 70,000. Units are in the scale of 10^4. . . . 81
5.6 Correlation between top results and the links/weights of these nodes 88

5.7 The rate that the initial score is overriden by the graph structure with Random Walks and Metropolis-Hastings 89

B.1 Astera three layer architecture 104
B.2 Crawling data, parsing and saving in the Graph DB. These stages are managed through packages of crawler, worker, parser and data. 106
B.3 Different processes in Astera 107
B.4 Design of Astera architecture 109
B.5 Deployment diagram of Astera 110
B.6 The configuration of primary and secondary runs in Astera 113
B.7 Run cycle in Astera: Each collection is indexed, searched, and then evaluated 114
B.8 We add sameAs links between the Wikipedia pages that have correspondent relation in DBPedia 117
B.9 Data flow between Java and Matlab programs 120

List of Tables

3.1 Top 7 most frequent semantic links after adding DBpeida links to ImageCLEF 2011 Wikipedia collection 44

5.1 Results for baseline standard search 70
5.2 Different combinations of weightings on document and image similarity results 71
5.3 The results of 24 steps in the graph starting from TF.IDF$_D$. We observe the descending order of precision and recall. 74
5.4 Results with R_D, st:0.9, links: β, δ 75
5.5 Results with R_D, R_{IV}, st:0.9, links: β, δ 75
5.6 Results with R_D, R_{IT}, st:0.9, links: β, δ 75
5.7 Results with R_D, R_{IV}, R_{IT}, st:0.9, links: β, δ 75
5.8 Result without semantic links, st = 0, β, δ 78
5.9 Result with semantic links (sameAs), st = 0, β, δ, α 78
5.10 Result without semantic/similarity links, st = 0.9, β, δ 79
5.11 Result with similarity links, st = 0.9, β, δ, γ 79
5.12 Result with semantic links (sameAs), st = 0.9, β, δ, α 79
5.13 Result with both semantic and similarity links, st = 0.9, β, δ, α, γ 79
5.14 Performance result with Random Walks 87
5.15 Performance Result with Metropolis-Hastings 87

A.1 Different categories of nodes in the graph 98

A.2 Memory needed to create a matrix in Matlab. The available memory of our machine is 100GB. The real free memory is about 90GB. If we touch the border of available memory it needs to swap and performance will degrade substantially. We work with matrix sizes that need less than 90GB. 98
A.3 The time needed for one iteration (one matrix multiplication) in Matlab . . . 99
A.4 Matlab matrix multiplications with different matrix sizes and operators . . . 99

B.1 Some methods of *Neo4jDBmanager.java* 109
B.2 Configuration variables in secondary.properties file 111
B.3 Database tables 112
B.4 Configuration variables in primary.properties file 114
B.5 Different configurations in Astera to index a collection or run the search and evaluate the indexed collection 115
B.6 Configuration variables related to similarity links in the graph DB 116
B.7 Configuration variables related to semantic links in the graph DB 116
B.8 Fields needed to be changed if we change the traversal method. The first field exists only in the Astera_randomWalks branch. 117
B.9 Output files of the recall analysis in AsteraAnalyser branch, saved in output/analysis/ path 119
B.10 Important methods of DistManger.java class 119
B.11 Graph visit configuration variables and output files in the secondary.properties, save in output/analysis path 119
B.12 The output files from Java or Matlab program. All files are saved in the output folder, e.g., output/res_vec/res-s2-t77.txt. 121
B.13 Matlab functions to be called from Java or Matlab in a run of Astera 123
B.14 Technologies used in Astera 124

Bibliography

[AFJG06] Asad Awan, Ronaldo A Ferreira, Suresh Jagannathan, and Ananth Grama. Distributed uniform sampling in unstructured peer-to-peer networks. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS)*, volume 9, 2006.

[AHESK10] Pradeep K Atrey, M Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Journal of Multimedia Systems*, 16(6):345–379, 2010.

[AHK01] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of Database Theory Conference*, pages 420–434, 2001.

[AIL$^+$03] WH Adams, Giridharan Iyengar, Ching-Yung Lin, Milind Ramesh Naphade, Chalapathy Neti, Harriet J Nock, and John R Smith. Semantic indexing of multimedia content using visual, audio, and text cues. *EURASIP Journal on Advances in Signal Processing*, 2003(2):1–16, 2003.

[AOB$^+$11] Adil Alpkocak, Okan Ozturkmenoglu, Tolga Berber, Ali Hosseinzadeh Vahid, and Roghaiyeh Gachpaz Hamed. DEMIR at imageclefmed 2011: Evaluation of fusion techniques for multimodal content-based medical image retrieval. In *Proceedings of Conference and Labs of the Evaluation Forum (CLEF)*, 2011.

[AR11] Avi Arampatzis and Stephen Robertson. Modeling score distributions in information retrieval. *Information Retrieval*, 14(1):26–46, 2011.

[BBB$^+$14] Petra Budikova, Jan Botorek, Michal Batko, Pavel Zezula, et al. Disa at imageclef 2014: The search-based solution for scalable image annotation. In *CLEF (Working Notes)*, pages 360–371, 2014.

[BBK$^+$09] Michael R. Berthold, Ulrik Brandes, Tobias Kotter, Martin Mader, Uwe Nagel, and Kilian Thiel. Pure spreading activation is pointless. In *Proceedings of Conference on Information and Knowledge Management (CIKM)*, 2009.

[Ber05] Pavel Berkhin. Survey: A survey on pagerank computing. *Journal of Internet Mathematics*, 2(1):73–120, 2005.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Journal of Computer Vision and Image Understanding*, 110(3), 2008.

[BGRS99] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of International Conference on Database Theory (ICDT)*, pages 217–235. 1999.

[Bin15] Bing homepage, 2015.

[BVO$^+$11] Tolga Berber, Ali Hosseinzadeh Vahid, Okan Ozturkmenoglu, Roghaiyeh Gachpaz Hamed, and Adil Alpkocak. Demir at imageclefwiki 2011: Evaluating different weighting schemes in information retrieval. In *Proceedings of Conference and Labs of the Evaluation Forum (CLEF)*, 2011.

[CB08] Savvas A Chatzichristofis and Yiannis S Boutalis. Cedd: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In *Proceedings of International Conference on Computer Vision Systems*, 2008.

[CBGM02] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[CC03] Ya-Chun Cheng and Shu-Yuan Chen. Image classification using color, texture and regions. *Journal of Image and Vision Computing*, 21(9), 2003.

[CDVR10a] Maarten Clements, Arjen P. De Vries, and Marcel J. T. Reinders. The task-dependent effect of tags and ratings on social media access. *ACM Transactions on Information Systems (TOIS)*, 28(4), November 2010.

[CDVR10b] Maarten Clements, Arjen P De Vries, and Marcel JT Reinders. The task-dependent effect of tags and ratings on social media access. *ACM Transactions on Information Systems (TOIS)*, 28(4):21, 2010.

[CG95] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *Journal of The American Statistician*, 49(4), 1995.

[CGD92] William S Cooper, Fredric C Gey, and Daniel P Dabney. Probabilistic retrieval based on staged logistic regression. In *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*, pages 198–210, 1992.

[Chr04] Mikkel Christoffersen. Identifying core documents with a multiple evidence relevance filter. *Journal of Scientometrics*, 61(3):385–394, 2004.

[CLM$^+$13] Jianliang Chen, Yuting Liu, Dipanwita Maulik, Linda Xu, Hao Zhang, Craig A Knoblock, Pedro Szekely, and Miel Vander Sande. Lodstories: Learning about art by building multimedia stories. Technical report, 2013.

[CMS07] Shih-Fu Chang, Wei-Ying Ma, and Arnold Smeulders. Recent advances and challenges of semantic image/video search. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages IV–1205, 2007.

[Cre97] Fabio Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6):453–482, 1997.

[CS07] Nick Craswell and Martin Szummer. Random walks on the click graph. In *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*, 2007.

[CSZ09] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[CTC05] Kevyn Collins-Thompson and Jamie Callan. Query expansion using random walk models. In *Proceedings of Conference on Information and Knowledge Management (CIKM)*, 2005.

[CYK$^+$05] Pei-Cheng Cheng, Jen-Yuan Yeh, Hao-Ren Ke, Been-Chian Chien, and Wei-Pang Yang. Comparison and combination of textual and visual features for interactive cross-language image retrieval. In *Multilingual Information Access for Text, Speech and Images*. 2005.

[DJLW08] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval: Ideas, influences, and trends of the new age. *Journal of ACM Computing Surveys*, 40(2):5, 2008.

[DLTX11] Lixin Duan, Wen Li, Ivor W Tsang, and Dong Xu. Improving web image search by bag-based reranking. *IEEE Transactions on Image Processing*, 20(11), 2011.

[DMDH02] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th international conference on World Wide Web (WWW)*, 2002.

[DTCT10] Renaud Delbru, Nickolai Toupikov, Michele Catasta, and Giovanni Tummarello. A node indexing scheme for web entity retrieval. In *Proceedings of Extended Semantic Web Conference (ESWC)*, 2010.

[EB11] Shady Elbassuoni and Roi Blanco. Keyword search over RDF graphs. In *Proceedings of Conference on Information and Knowledge Management (CIKM)*, 2011.

[FFFPZ05] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from Google's image search. In *Proceedings of International Conference on Computer Vision*, 2005.

[FPZ04] Robert Fergus, Pietro Perona, and Andrew Zisserman. A visual category filter for google images. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 242–256. 2004.

[FRA+05] Ronaldo A Ferreira, Murali Krishna Ramanathan, Asad Awan, Ananth Grama, and Suresh Jagannathan. Search with probabilistic guarantees in unstructured peer-to-peer networks. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2005.

[Goo15] Google homepage, 2015.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[Has70] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Journal of Biometrika*, 57(1):97–109, 1970.

[HC09] Myron Hlynka and Michelle Cylwa. *Observations on the Metropolis-Hastings Algorithm.* University of Windsor, Department of Mathematics and Statistics, 2009.

[HCY08] Alexander G Hauptmann, Michael G Christel, and Rong Yan. Video retrieval based on semantic concepts. *Proceedings of the IEEE*, 96(4):602–622, 2008.

[HG10] Sung Ju Hwang and Kristen Grauman. Accounting for the relative importance of objects in image retrieval. In *Proceedings of British Machine Vision Conference (BMVC)*, pages 1–12, 2010.

[HKC07] Winston H Hsu, Lyndon S Kennedy, and Shih-Fu Chang. Video search reranking through random walk over document-level context graph. In *Proceedings of the 15th ACM International Conference on Multimedia*, 2007.

[HLZ+04] Jingrui He, Mingjing Li, Hong-Jiang Zhang, Hanghang Tong, and Changshui Zhang. Manifold-ranking based image retrieval. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 9–16, 2004.

[HM09] Gilles Hubert and Josiane Mothe. An adaptable search engine for multimodal information retrieval. *Journal of the American Society for Information Science and Technology*, 60(8):1625–1634, 2009.

[HNS04] Hartwig Holzapfel, Kai Nickel, and Rainer Stiefelhagen. Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3d pointing gestures. In *Proceedings of the 6th International Conference on Multimodal Interfaces*, pages 175–182, 2004.

[HZ04] Xian-Sheng Hua and Hong-Jiang Zhang. An attention-based decision fusion scheme for multimedia information retrieval. In *Proceedings of Pacific-Rim Conference on Multimedia*, pages 1001–1010, 2004.

[HZ08] Tim Hussein and Jürgen Ziegler. Adapting web sites by spreading activation in ontologies. In *Proceedings of International Workshop on Recommendation and Collaboration*, 2008.

[IJ06] Peter Ingwersen and Kalervo Järvelin. *The turn: Integration of information seeking and retrieval in context*, volume 18. Springer Science & Business Media, 2006.

[IN03] Giridharan Iyengar and Harriet J Nock. Discriminative model fusion for semantic concept detection and annotation in video. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, pages 255–258, 2003.

[Ing96] Peter Ingwersen. Cognitive perspectives of information retrieval interaction: elements of a cognitive ir theory. *Journal of Documentation*, 52(1):3–50, 1996.

[INN03] Giridharan Iyengar, Harriet J Nock, and Chalapathy Neti. Audio-visual synchrony for detection of monologues in video archives. In *Proceedings of International Conference on Multimedia and Expo (ICME)*, pages I–329, 2003.

[JB08] Yushi Jing and Shumeet Baluja. Visualrank: Applying pagerank to large-scale image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1877–1890, 2008.

[JNR05a] Anil Jain, Karthik Nandakumar, and Arun Ross. Score normalization in multimodal biometric systems. *Journal of Pattern recognition*, 38(12):2270–2285, 2005.

[JNR05b] Anil Jain, Karthik Nandakumar, and Arun Ross. Score normalization in multimodal biometric systems. *Journal of Pattern Recognition*, 2005.

[JWL06] Dhiraj Joshi, James Z Wang, and Jia Li. The story picturing engine—a system for automatic text illustration. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):68–89, 2006.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[KC07] Lyndon S Kennedy and Shih-Fu Chang. A reranking approach for context-based concept fusion in video indexing and retrieval. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, pages 333–340, 2007.

[KDF05] Diane Kelly, Vijay Deepak Dollu, and Xin Fu. The loquacious user: a document-independent source of terms for query expansion. In *Proceedings of Special Interest Group On Information Retrieval (SIGIR)*, pages 457–464, 2005.

[KF07] Diane Kelly and Xin Fu. Eliciting better information need descriptions from users of information search systems. *Journal of Information Processing & Management*, 43(1):30–46, 2007.

[KKW$^+$14] Ashnil Kumar, Jinman Kim, Lingfeng Wen, Michael Fulham, and Dagan Feng. A graph-based approach for the retrieval of multi-modality medical images. *Journal of Medical Image Analysis*, 18(2):330–342, 2014.

[KSI$^+$08] G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. Naga: Searching and ranking knowledge. In *Proceedings of IEEE 2008 Conference Data Engineering (ICDE)*, 2008.

[KWJ06] Mohan S Kankanhalli, Jun Wang, and Ramesh Jain. Experiential sampling in multimedia systems. *IEEE Transactions on Multimedia*, 8(5):937–946, 2006.

[Lar02] Birger Larsen. Exploiting citation overlaps for information retrieval: Generating a boomerang effect from the network of scientific papers. *Journal of Scientometrics*, 54(2):155–178, 2002.

[Lar04] Birger Larsen. *References and citations in automatic indexing and retrieval systems-experiments with the boomerang effect.* PhD thesis, Københavns University, Faculty of Humanities, School of Library and Information Science, 2004.

[LARD13] Michalis Lazaridis, Apostolos Axenopoulos, Dimitrios Rafailidis, and Petros Daras. Multimedia search and retrieval using multimodal annotation propagation and indexing techniques. *Journal of Signal Processing: Image Communication*, 28(4):351–367, 2013.

[LGG04] Ai Poh Loh, Feng Guan, and Shuzhi Sam Ge. Motion estimation using audio and video fusion. In *Proceedings of International Conference on Control, Automation, Robotics and Vision (ICCARV)*, pages 1569–1574, 2004.

[LI05] Birger Larsen and Peter Ingwersen. Cognitive overlaps along the polyrepresentation continuum. In *New Directions in Cognitive Information Retrieval.* 2005.

[LIK06] Birger Larsen, Peter Ingwersen, and Jaana Kekäläinen. The polyrepresentation continuum in IR. In *Proceedings of Information Interaction in Context (IIiX)*, 2006.

[LIL09] Birger Larsen, Peter Ingwersen, and Berit Lund. Data fusion according to the principle of polyrepresentation. *Journal of the American Society for Information Science and Technology*, 60(4):646–654, 2009.

[LLH$^+$07] Jingjing Liu, Wei Lai, Xian-Sheng Hua, Yalou Huang, and Shipeng Li. Video search re-ranking via multi-graph propagation. In *Proceedings of the 15th ACM International Conference on Multimedia*, pages 208–217, 2007.

[LSC01] Simon Lucey, Sridha Sridharan, and Vinod Chandran. Improved speech recognition using adaptive audio-visual fusion via a stochastic secondary classifier. In *Proceedings of the International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 551–554. IEEE, 2001.

[LSW10] Xirong Li, Cees GM Snoek, and Marcel Worring. Unsupervised multi-feature tag relevance learning for social image retrieval. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 10–17, 2010.

[LYS02] Ren C Luo, Chih-Chen Yih, and Kuo Lan Su. Multisensor fusion and integration: approaches, applications, and future research directions. *IEEE Sensors Journal*, 2(2):107–119, 2002.

[MCN06] Einat Minkov, William W Cohen, and Andrew Y Ng. Contextual search and name disambiguation in email using graphs. In *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*, pages 27–34, 2006.

[MDS05] Kieran Mc Donald and Alan F Smeaton. A comparison of score, rank and probability-based fusion methods for video shot retrieval. In *Image and Video Retrieval*, pages 61–70. 2005.

[MM99] Wei-Ying Ma and Bangalore S Manjunath. Netra: A toolbox for navigating large image databases. *Journal of Multimedia Systems*, 7(3):184–198, 1999.

[MPC$^+$10] Enrico Minack, Raluca Paiu, Stefania Costache, Gianluca Demartini, Julien Gaugaz, Ekaterini Ioannou, Paul-Alexandru Chirita, and Wolfgang Nejdl. Leveraging personal metadata for desktop search: The Beagle++ system. *Journal of Web Semantics: Science, Services and Agents on the WWW*, 8(1):37–54, 2010.

[MRLT14] Tao Mei, Yong Rui, Shipeng Li, and Qi Tian. Multimedia search reranking: A literature survey. *Journal of ACM Computing Surveys (CSUR)*, 46(3):38, 2014.

[MS07] Jean Martinet and Shin'ichi Satoh. An information theoretic approach for automatic document annotation from intermodal analysis. In *Proceedings of the Workshop on Multimodal Information Retrieval*, 2007.

[MSBT11] D. Magatti, F. Steinke, M. Bundschus, and V. Tresp. Combined Structured and Keyword-Based Search in Textually Enriched Entity-Relationship Graphs. In *Proceedings of the Workshop on Automated Knowledge Base Construction*, 2011.

[MU49] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[NF03] Henrik Nottelmann and Norbert Fuhr. From retrieval status values to probabilities of relevance for advanced ir applications. *Journal of Information Retrieval*, 6(3-4), 2003.

[NNT05] Apostol (Paul) Natsev, Milind R. Naphade, and Jelena TešiĆ. Learning the semantics of multimedia queries and concepts from a small number of examples. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, 2005.

[OP99] Timo Ojala and Matti Pietikäinen. Unsupervised texture segmentation using feature distributions. *Journal of Pattern Recognition*, 32(3):477–486, 1999.

[PKPM06] Vassilis Pitsikalis, Athanassios Katsamanis, George Papandreou, and Petros Maragos. Adaptive multimodal fusion by uncertainty compensation. In *Proceedings of INTERSPEECH Conference*, 2006.

[PPS96] Alex Pentland, Rosalind W Picard, and Stan Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, 1996.

[RB82] S. E. Robertson and J. D. Bovey. Statistical problems in the application of probabilistic models to information retrieval. Technical Report Report No. 5739, Microsoft Research, January 1982.

[RD02] Mathew Richardson and Pedro Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2002.

[Red07] Bakkama Srinath Reddy. Evidential reasoning for multimodal fusion in human computer interaction. 2007.

[RG99] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Journal of Neural Computation*, 11(2):305–345, 1999.

[RG03] Ali Rashidi and Hassan Ghassemian. Extended dempster–shafer theory for multi-system/sensor decision fusion. In *Proceedings of the Commission IV Joint Workshop on Challenges in Geospatial Analysis, Integration and Visualization II*, pages 31–37, 2003.

[Rob97] S. E. Robertson. Readings in information retrieval. chapter The Probability Ranking Principle in IR, pages 281–286. 1997.

[Rob07] Stephen Robertson. *On score distributions and relevance.* Springer, 2007.

[RSA04] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi Aragao. A hybrid approach for searching in the semantic web. In *Proceedings of World Wide Web Conference (WWW)*, 2004.

[SB88] G. Salton and C. Buckley. On the use of spreading activation methods in automatic information. In *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*, 1988.

[SBR14] Serwah Sabetghadam, Ralf Bierig, and Andreas Rauber. A hybrid approach for multi-faceted IR in multimodal domain. In *Proceedings of Conference and Labs of the Evaluation Forum (CLEF)*, 2014.

[SC96] John R Smith and Shih-Fu Chang. Automated image retrieval using color and texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[SJ01] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2001.

[SLBR14] Serwah Sabetghadam, Mihai Lupu, Ralf Bierig, and Andreas Rauber. A combined approach of structured and non-structured IR in multimodal domain. In *Proceedings of International Conference on Multimedia Retrieval (ICMR)*, 2014.

[SLBR15] Serwah Sabetghadam, Mihai Lupu, Ralf Bierig, and Andreas Rauber. Reachability analysis of graph modelled collections. In *European Conference on Information Retrieval (ECIR)*, 2015.

[SLI08] Mette Skov, Birger Larsen, and Peter Ingwersen. Inter and intra-document contexts applied in polyrepresentation for best match IR. *Journal of Information Processing & Management*, 44(5):1673–1683, 2008.

[SLR13] Serwah Sabetghadam, Mihai Lupu, and Andreas Rauber. Astera - a generic model for multimodal information retrieval. In *Proceedings of Integrating IR Technologies for Professional Search Workshop*, 2013.

[SLR14] Serwah Sabetghadam, Mihai Lupu, and Andreas Rauber. Which one to choose: Random walk or spreading activation? In *Proceedings of Information Retrieval Facility Conference (IRFC)*, 2014.

[SLR15] Serwah Sabetghadam, Mihai Lupu, and Andreas Rauber. Leveraging metropolis-hastings algorithm on graph-based model for multimodal IR. In *Proceedings of the First International Workshop on Graph Search and Beyond (GSB)*, 2015.

[SLR16] Serwah Sabetghadam, Mihai Lupu, and Andreas Rauber. Random walks analysis on graph modelled multimodal collections. In *Second International KEYSTONE Conference*, 2016.

[SNF02] Renato O Stehling, Mario A Nascimento, and Alexandre X Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *Proceedings of Conference on Information and Knowledge Management (CIKM)*, 2002.

[SP04] Conrad Sanderson and Kuldip K Paliwal. Identity verification using speech and face information. *Journal of Digital Signal Processing*, 14(5):449–480, 2004.

[SPLI04] Mette Skov, Henriette Pedersen, Birger Larsen, and Peter Ingwersen. Testing the principle of polyrepresentation. In *Proceedings of the SIGIR Workshop on Information Retrieval in Context*, pages 47–49, 2004.

[SRM09] Hinrich Schütze, P Raghavan, and Christopher Manning. An introduction to information retrieval, 2009.

[SS07] S. Srinivasan and M. Slaney. A bipartite graph model for associating images and text. In *Proceedings of the Workshop on Multimodal Information Retrieval*, 2007.

[SSH14] Bahjat Safadi, Mathilde Sahuguet, and Benoit Huet. Linking text and visual concepts semantically for cross modal multimedia search. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2014.

[SWS+00] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[SWS05] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, pages 399–402, 2005.

[TDCM12] Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux. Combining inverted indices and structured search for ad-hoc object retrieval. In *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*, 2012.

[TdM14] Ilaria Tiddi, Mathieu d'Aquin, and Enrico Motta. Walking linked data: a graph traversal approach to explain clusters. In *Proceedings of the Fifth International Workshop on Consuming Linked Data (COLD)*, 2014.

[Tow07] Christopher Town. Multi-sensory and multi-modal fusion for sentient computing. *International Journal of Computer Vision*, 71(2):235–253, 2007.

[TPK11] Theodora Tsikrika, Adrian Popescu, and Jana Kludas. Overview of the wikipedia image retrieval task at imageclef 2011. In *Conference and Labs of the Evaluation Forum (CLEF)*, 2011.

[TYW$^+$08] Xinmei Tian, Linjun Yang, Jingdong Wang, Yichen Yang, Xiuqing Wu, and Xian-Sheng Hua. Bayesian video search reranking. In *Proceedings of the 16th ACM International Conference on Multimedia*, 2008.

[VNH03] Atulya Velivelli, Chong-Wah Ngo, and Thomas S Huang. Detection of documentary scene changes by audio-visual fusion. *Journal of Lecture notes in computer science*, pages 227–237, 2003.

[Wal04] Brian Walsh. Markov chain monte carlo and gibbs sampling, 2004.

[WCCS04a] Yi Wu, Edward Y. Chang, Kevin Chen-Chuan Chang, and John R. Smith. Optimal multimodal fusion for multimedia data analysis. *Proceedings of the 12th annual ACM international conference on Multimedia*, 2004.

[WCCS04b] Yi Wu, Edward Y Chang, Kevin Chen-Chuan Chang, and John R Smith. Optimal multimodal fusion for multimedia data analysis. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 572–579, 2004.

[WHH$^+$09] Meng Wang, Xian Sheng Hua, Richang Hong, Jinhui Tang, Guo Jun Qi, and Yan Song. Unified Video Annotation via Multigraph Learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(5):733–746, May 2009.

[Whi06] Ryen W White. Using searcher simulations to redesign a polyrepresentative implicit feedback interface. *Journal of Information Processing & Management*, 42(5):1185–1202, 2006.

[WJH$^+$07] Shuo Wang, Feng Jing, Jibo He, Qixing Du, and Lei Zhang. Igroup: presenting web image search results in semantic clusters. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 587–596, 2007.

[WKYJ03] Jun Wang, Mohan S Kankanhalli, Weiqi Yan, and Ramesh Jain. Experiential sampling for video surveillance. In *Proceedings of the First ACM SIGMM International Workshop on Video Surveillance*, pages 77–86, 2003.

[WLCS04] Yi Wu, C-Y Lin, Edward Y Chang, and John R Smith. Multimodal information fusion for video concept detection. In *Proceedings of International Conference on Image Processing (ICIP)*, volume 4, pages 2391–2394, 2004.

[WLT^{+}12] Meng Wang, Hao Li, Dacheng Tao, Ke Lu, and Xindong Wu. Multimodal graph-based reranking for web image search. *IEEE Transactions on Image Processing*, 21(11), 2012.

[WYJ16] Wei Wang, Xiaoyan Yang, and Shouxu Jiang. Cross-modal search on social networking systems by exploring wikipedia concepts. In *Digital Libraries: Knowledge, Information, and Data in an Open Access Society*, pages 381–393. Springer, 2016.

[WZZL10] Shikui Wei, Yao Zhao, Zhenfeng Zhu, and Nan Liu. Multimodal fusion for video search reranking. *IEEE Transactions on Knowledge and Data Engineering*, 22(8):1191–1199, 2010.

[XC06] Huaxin Xu and Tat-Seng Chua. Fusion of AV features and external information sources for event detection in team sports video. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):44–67, 2006.

[Yan06] Rong Yan. *Probabilistic models for combining diverse knowledge sources in multimedia retrieval.* PhD thesis, IBM, 2006.

[YH10] Linjun Yang and Alan Hanjalic. Supervised reranking for web image search. In *Proceedings of the 18th ACM International Conference on Multimedia*, pages 183–192, 2010.

[YHJ03a] Rong Yan, Alexander Hauptmann, and Rong Jin. Multimedia search with pseudo-relevance feedback. In *Proceedings of International Conference on Image and Video Retrieval*, 2003.

[YHJ03b] Rong Yan, Alexander G Hauptmann, and Rong Jin. Negative pseudo-relevance feedback in content-based video retrieval. In *Proceedings of the eleventh ACM International Conference on Multimedia*, pages 343–346, 2003.

[YHWW06] Xun Yuan, Xian-Sheng Hua, Meng Wang, and Xiu-Qing Wu. Manifold-ranking based video concept detection on large database and feature pool. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, pages 623–626, 2006.

[YMN10] Ting Yao, Tao Mei, and Chong-Wah Ngo. Co-reranking by mutual reinforcement for image search. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 34–41, 2010.

[ZB11] David Zellhöfer and Thomas Böttcher. BTU DBIS' multimodal Wikipedia retrieval runs at ImageCLEF 2011. In *Proceedings of Conference and Labs of the Evaluation Forum (CLEF)*, 2011.

[ZBL$^+$04] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Journal of Advances in Neural Information Processing Systems*, 16(16):321–328, 2004.

[ZGL$^+$03] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 912–919, 2003.

[ZSOD08] Hilal Zitouni, Sare Sevil, Derya Ozkan, and Pinar Duygulu. Re-ranking of web image search results using a graph algorithm. In *Proceedings of 19th International Conference on Pattern Recognition (ICPR)*, pages 1–4, 2008.

[ZSS08] Ming Zhong, Kai Shen, and Joel Seiferas. The convergence-guaranteed random walk and its applications in peer-to-peer networks. *IEEE Transactions on Computers*, 57(5):619–633, 2008.

[ZYYZ14] Sicheng Zhao, Hongxun Yao, You Yang, and Yanhao Zhang. Affective image retrieval via multi-graph learning. In *Proceedings of the ACM International Conference on Multimedia*, pages 1025–1028, 2014.