UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCES AND ENGINEERING

Exploiting Linked Data in Exploratory Search

by

Pavlos Fafalios

PhD Dissertation

Presented

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

Heraklion, May 2016

UNIVERSITY OF CRETE

DEPARTMENT OF COMPUTER SCIENCE

Exploiting Linked Data in Exploratory Search

PhD Dissertation Presented

by **Pavlos Fafalios**

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

APPROVED BY:

Author: Pavlos Fafalios

Supervisor: Tzitzikas Yannis, Assistant Professor, University of Crete

---

Committee Member: Salampasis Michail, Professor, ATEI of Thessaloniki

---

Committee Member: Rauber Andreas, Associate Professor, Vienna University of Technology

---

Committee Member: Fundulaki Irini, Principal Researcher, FORTH-ICS

Committee Member: Crestani Fabio, Professor, University of Lugano

---

Committee Member: Plexousakis Dimitris, Professor, University of Crete

---

Committee Member: Stephanidis Constantine, Professor, University of Crete

---

Department Chairman: Tsakalidis Panagiotis, Professor, University of Crete

Heraklion, May 2016

This Dissertation is dedicated to my parents, Markos and Anastasia.

For their endless love, support and encouragement.

Acknowledgments

It is hard to express my gratitude to my supervisor, Assistant Professor Yannis Tzitzikas. I have been extremely lucky to have a supervisor who cared so much about my work and progress, and who was always available for me, responding to my questions so promptly. Without his inspirational guidance, enthusiasm, encouragements, and unselfish help, I could never finish my doctoral work. His insights have been really inspiring and crucial.

I would like to deeply thank Professor Michail Salampasis, member of my advisory committee and Chair of MUMIA Cost Action, for the excellent guidance and suggestions, and for offering me the opportunity to discuss my work with experienced researchers during my participation in several MUMIA events. I also thank the third member of my advisory committee, Associate Professor Andreas Rauber, for his valuable comments and suggestions all these years. I am also grateful to the other members of my examination committee, Principal Researcher Irini Fundulaki, Professor Fabio Crestani, Professor Dimitris Plexousakis, and Professor Constantine Stephanidis, for their constructive suggestions and comments. I am also thankful to the anonymous reviewers of my papers for providing me valuable feedback which helped me improve my work.

I would like to acknowledge the support of the Institute of Computer Science of the Foundation of Research and Technology (FORTH-ICS), and especially the Information Systems Laboratory (ISL), for both the financial support and the facilities. In particular I thank the Head of ISL, Professor Dimitris Plexousakis, as well as all the faculty and staff members of ISL, for creating a highly creative and inspiring environment for me. I also thank the "Maria Michael Manasaki" legacy for recognizing my progress and awarding me a fellowship for the academic year 2014-2015. Furthermore, I would like to thankfully acknowledge the support of iMarine (FP7, 2011-2014), MUMIA COST Action (IC1002, 2010-2014), and BlueBridge (H2020, 2015-2018). Part of my dissertation was supported and influenced by these projects.

My deepest thanks to all the faculty, staff, and students at University of Crete. My special thanks goes to Gelly Kosma, secretary of Computer Science Department, for providing me with so much help and secretarial support during my postgraduate studies. I would also like to thank Giannis Sourlantzis for the technical support during my thesis defense.

Last but not least, I was fortunate to meet so many exceptional and inspiring friends and colleagues during my postgraduate studies in Crete. First, I want to deeply thank Konstantina Konsolaki for her endless love, patience and generous support. She always encouraged me and stood by me through thick and thin. I also want to deeply thank the

following friends and colleagues for the so many good times we have shared all these years: Giorgos Bouloukakis, Giorgos Grekas, Giorgos Mathioudakis, Katerina Orfanoudaki, Georgia and Vili Kanli, Vasilis Papakontantinou, Maria Chalkiadaki, little Myrto, Georgia Troullinou, Nina Saveta, Nikos Minadakis, and Vasilis Euthymiou. Special thanks to Yannis Marketakis and Panagiotis Papadakos for the many interesting discussions and for answering at EVERY question I had, as well as to Kostas Stefanidis for the useful advices at the end of my PhD. I would also like to thank Ioannis Kitsos for introducing me in *entity mining*, Manolis Baritakis for implementing several parts of X-Link, Michalis Mountantonakis for the Hack4Med hackathon and the trip in Turin, Maria Moutsaki for the secretarial support and the help in all my trips, and Dimitris Agelakis for the technical support.

This work is a result of the constant and endless love of my parents, Markos and Anastasia, and my sister Nantia. They always believed and supported me in any possible way.

Abstract

In recent years we have witnessed an explosion in publishing data on the Web, mostly in the form of Linked Data. An important question is how typical users, who mainly use keyword search queries, can access and exploit this constantly increasing body of knowledge. Although existing interaction paradigms in Semantic Search hide their complexity behind easy-to-use interfaces, they have not managed to cover common search needs. At the same time, according to several studies, a large number of search tasks are of exploratory nature. However, in such tasks the traditional "ranked list" approach for interacting with the retrieved results is often inadequate.

The objective of this thesis is to enable effective exploratory search services which can bridge the gap between the classic responses of non-semantic search systems (e.g., Professional Search Systems, Web Search Engines) and semantic information expressed in the form of Linked Open Data (LOD). Towards this direction, we introduce an approach in which named entities (like names of persons, locations, chemical substances, etc.) are exploited as the glue for automatically connecting documents (search results) with data and knowledge. We study an approach where this entity-based integration is performed at real-time, without any human intervention and without the need of prebuilt indexes. This allows the provision of "fresh" information, the easy configuration of this functionality according to the needs of the underlying search application, as well as its easy exploitation by existing search systems.

The provision of the aforementioned functionality is challenging. At first, the LOD that are available on the Web are big, are distributed in many knowledge bases, are increased and updated continuously, and also cover many domains. Consequently, there is the need of an interoperability model that will allow the specification of the entities of interest as well as of the related and useful semantic data. In addition, the number of extractable entities from the search results can be very high and the same is true for the amount of semantic information that can be retrieved from the LOD for these entities (i.e., the number of their attributes and of their associations with other entities). Thus, there is also the need of methods that can estimate the important (for the search context) entities, attributes and associations.

To cope with above challenges, this thesis proposes a semantic analysis process in which the search results are connected with data and knowledge at real-time without any human intervention. For describing the entities of interest, as well as the related (and useful for the application context) semantic information, we propose a generic model for

configuring a Named Entity Extraction (NEE) system, while for specifying the semantics of this model, we introduce an RDF/S vocabulary, called "Open NEE Configuration Model", which allows a NEE system to describe (and publish as LOD) its entity-mining capabilities. To enable associating the result of a NEE process with an applied configuration, we propose an extension of the Open Annotation Data Model which also allows publishing the annotation results as LOD. To examine the feasibility of this model, we developed the system X-Link which, contrary to existing NEE systems, allows its easy configuration by exploiting one or more semantic Knowledge Bases. To identify the important semantic information related to the search results, we introduce and study a ranking method that is based on the Random Walk model and which exploits the extracted entities and their connectivity. The exploitation of the selected semantic information is achieved either through the visualization of the related semantic graph and/or in the context of a faceted interaction model that allows the user to gradually restrict the search space. Besides, this thesis studied the exploitation of such graphs for re-ranking the list of retrieved results aiming to promote relevant but low-ranked hits.

The dissertation reports extensive evaluation results of the proposed functionalities and methods. As regards the system X-Link, a task-based evaluation with users showed its ease of configuration, while a case study illustrated the efficiency of the supported operations. The comparative evaluation of the proposed probabilistic scheme for ranking entities and semantic data showed that the proposed approach is more effective compared to other ranking approaches (producing more than 20% better ranking). As regards the presentation of the important entities (and of their associations), the conducted survey in a marine-related search context demonstrated that the majority of participants (more than 70%) prefer to see a graph representation of entities related to the retrieved results regardless the type of the submitted query. The evaluation of the proposed probabilistic algorithm for re-ranking the retrieved search results (using TREC datasets related to the medicine domain) showed that this approach can notably improve the list of results by promoting relevant hits in higher positions. Finally, the implementation and the experimental results of the proposed search process demonstrated its feasibility and efficiency, and also enabled us to reveal its limitations.

Keywords: Exploratory Search, Linked Open Data, Named Entity Extraction, Random Walk, Semantic Search

Supervisor: Tzitzikas Yannis
Assistant Professor
Computer Science Department
University of Crete

Περίληψη

Τα τελευταία χρόνια παρατηρείται μια έκρηξη στη δημοσίευση δεδομένων στον Παγκόσμιο Ιστό, κυρίως με τη μορφή Διασυνδεδεμένων Δεδομένων (Linked Data). Ένα βασικό ερώτημα όμως είναι πως αυτός ο συνεχώς αυξανόμενος πλούτος γνώσεων μπορεί να αξιοποιηθεί από απλούς χρήστες για καλύτερη αναζήτηση πληροφοριών. Αν και τα υπάρχοντα συστήματα σημασιολογικής αναζήτησης αποκρύπτουν την πολυπλοκότητα τους αξιοποιώντας φιλικά στη χρήση μέσα αλληλεπίδρασης, δεν έχουν καταφέρει ακόμα να καλύψουν κοινές – γενικού σκοπού – ανάγκες αναζήτησης και διασύνδεσης πληροφοριών. Παράλληλα, σύμφωνα με διάφορες μελέτες, ένα μεγάλο ποσοστό των αναζητήσεων είναι εξερευνητικού χαρακτήρα. Σε τέτοιου είδους πληροφοριακές ανάγκες οι παραδοσιακές απαντήσεις που έχουν τη μορφή γραμμικής λίστας αποτελεσμάτων συνήθως δεν είναι ικανοποιητικές.

Ο σκοπός αυτής της διατριβής είναι η παροχή προηγμένων υπηρεσιών εξερευνητικής αναζήτησης οι οποίες γεφυρώνουν το χάσμα μεταξύ των κλασσικών απαντήσεων μη σημασιολογικών συστημάτων αναζήτησης (επαγγελματικής ή μη φύσεως) και σημασιολογικής πληροφορίας εκφρασμένης με τη μορφή Ανοιχτών Διασυνδεδεμένων Δεδομένων (ΑΔΔ). Προς αυτή τη κατεύθυνση, εισάγουμε και εξετάζουμε μια προσέγγιση κατά την οποία επώνυμες οντότητες (όπως για παράδειγμα πρόσωπα, περιοχές, χημικές ουσίες, κτλ.) αξιοποιούνται ως ο συνδετικός κρίκος για την αυτόματη διασύνδεση εγγράφων (αποτελεσμάτων αναζήτησης) με δεδομένα και γνώση. Μελετούμε μια προσέγγιση όπου αυτή η – βασισμένη σε οντότητες – ενοποίηση πραγματοποιείται σε πραγματικό χρόνο, κατά τη στιγμή της αναζήτησης, χωρίς εμπλοκή του χρήστη, αλλά και χωρίς την ανάγκη προκατασκευασμένων ευρετηρίων. Αυτό επιτρέπει την παροχή «φρέσκιας» πληροφορίας, την εύκολη παραμετροποίηση αυτής της λειτουργικότητας σύμφωνα με τις ανάγκες του υποκείμενου συστήματος αναζήτησης, αλλά και την εύκολη αξιοποίηση της από τα υπάρχοντα εργαλεία ανάκτησης πληροφοριών.

Η παροχή της παραπάνω λειτουργικότητας έχει διάφορες προκλήσεις. Αρχικά, τα ΑΔΔ που είναι διαθέσιμα στον Παγκόσμιο Ιστό έχουν μεγάλο μέγεθος, είναι κατανεμημένα σε πολλές Βάσεις Γνώσεων, αυξάνονται και ενημερώνονται συνεχώς, και καλύπτουν πολλές θεματικές περιοχές. Εκ τούτου, προκύπτει η ανάγκη για ένα μοντέλο διαλειτουργικότητας που να επιτρέπει τον προσδιορισμό των οντοτήτων ενδιαφέροντος και των σχετικών σημασιολογικών δεδομένων (από διάφορες βάσεις γνώσεων). Συνάμα ο αριθμός των εξορύξιμων οντοτήτων από τα αποτελέσματα αναζήτησης μπορεί να είναι αρκετά μεγάλος και το ίδιο ισχύει για το μέγεθος της

σημασιολογικής πληροφορίας που μπορεί να ανακτηθεί από τα ΑΔΔ για αυτές τις οντότητες (ήτοι το πλήθος των χαρακτηριστικών τους και των συσχετίσεών τους με άλλες οντότητες). Εκ τούτου προκύπτει η ανάγκη για μεθόδους που να μπορούν να εκτιμήσουν τις πιο σημαντικές οντότητες, καθώς και τη σημαντική σχετική σημασιολογική πληροφορία, για τα αποτελέσματα της εκάστοτε αναζήτησης.

Για την επιτυχή αντιμετώπιση των παραπάνω προκλήσεων, η διατριβή προτείνει μία διαδικασία ανάλυσης αποτελεσμάτων αναζήτησης κατά την οποία τα αποτελέσματα συνδέονται με δεδομένα και γνώσεις σε πραγματικό χρόνο, χωρίς τη μεσολάβηση του χρήστη. Για την περιγραφή των οντοτήτων ενδιαφέροντος και της σχετικής σημασιολογικής πληροφορίας προτείνεται ένα γενικό μοντέλο παραμετροποίησης Συστημάτων Εξαγωγής Οντοτήτων (ΣΕΟ), ενώ για τον ακριβή προσδιορισμό της σημασιολογίας αυτού του μοντέλου, εισάγουμε ένα RDF/S λεξιλόγιο με όνομα ''Open Named Entity Extraction (NEE) Configuration Model'', το οποίο επιτρέπει ένα ΣΕΟ να περιγράψει (και να δημοσιεύει ως ΑΔΔ) τις δυνατότητες του. Για να καταστήσουμε δυνατή τη συσχέτιση του αποτελέσματος της διαδικασίας εξαγωγής οντοτήτων με τις παραμέτρους που χρησιμοποιήθηκαν, προτείνουμε μια επέκταση του μοντέλου Open Annotation η οποία επιτρέπει και τη δημοσίευση των αποτελεσμάτων της εξόρυξης ως ΑΔΔ. Για την διερεύνηση της επιτευξιμότητας αυτού του μοντέλου αναπτύχθηκε το σύστημα X-Link το οποίο σε αντίθεση με τα υπάρχοντα ΣΕΟ επιτρέπει με εύκολο τρόπο τον προσδιορισμό των κατηγοριών οντοτήτων και των σημασιολογικών δεδομένων που ενδιαφέρουν την υποκείμενη εφαρμογή αξιοποιώντας μια ή περισσότερες σημασιολογικές Βάσεις Γνώσεων. Για τον εντοπισμό των πιο σημαντικών σημασιολογικών πληροφοριών που σχετίζονται με τα αποτελέσματα μιας αναζήτησης, εισάγουμε και μελετάμε μια μέθοδο κατάταξης βασισμένη στο μοντέλο Τυχαίου Περιπάτου (Random Walk) που αξιοποιεί τις εξηγμένες οντότητες και τη διασύνδεσή τους. Η αξιοποίηση αυτών των σημασιολογικών πληροφοριών γίνεται είτε με την οπτικοποίηση του σχετικού γράφου ή/και στα πλαίσια ενός πολυδιάστατου μοντέλου αλληλεπίδρασης που επιτρέπει στον χρήστη να περιορίσει τον πληροφοριακό του χώρο αυξητικά. Πέραν αυτού, η διατριβή αυτή μελέτησε την αξιοποίηση τέτοιων γράφων και για την ανακατάταξη της λίστας αποτελεσμάτων με σκοπό την βελτίωση της, συγκεκριμένα για την προώθηση εγγράφων που αν και είναι συναφή με την επερώτηση δεν είναι στις πρώτες θέσεις της κατάταξης.

Η διατριβή αναφέρει εκτενή αποτελέσματα αξιολόγησης των προτεινόμενων λειτουργιών και μεθόδων. Αναφορικά με το σύστημα X-Link, η αξιολόγηση με χρήστες έδειξε την ευκολία παραμετροποίησης, ενώ μια μελέτη περίπτωσης έδειξε την απόδοση των υποστηριζόμενων λειτουργιών του. Η συγκριτική αξιολόγηση του προτεινόμενου αλγορίθμου κατάταξης των οντοτήτων και της σχετικής σημασιολογικής πληροφορίας έδειξε ότι η προτεινόμενη προσέγγιση είναι πιο αποτελεσματική σε σχέση με άλλες μεθόδους ανακατάταξης. Αναφορικά με τον τρόπο παρουσίασης των ση-

μαντικών οντοτήτων (και των διασυνδέσεων τους) που σχετίζονται με μία απάντηση, τα αποτελέσματα μιας αξιολόγησης που έγινε με χρήστες στην περιοχή της αναζήτησης θαλάσσιων ειδών, έδειξε ότι η πλειονότητα των συμμετεχόντων (περισσότεροι από το 70%) προτιμούν μια γραφική απεικόνιση των οντοτήτων που σχετίζονται με τα αποτελέσματα αναζήτησης, ανεξαρτήτως του τύπου επερώτησης. Η αξιολόγηση του προτεινόμενου πιθανοτικού αλγορίθμου ανακατάταξης των επιστρεφόμενων αποτελεσμάτων που έγινε με συλλογές αξιολόγησης από το TREC (Text Retrieval Conference) που αφορούν τον τομέα της ιατρικής, κατέδειξε ότι η προσέγγιση αυτή μπορεί να βελτιώσει σημαντικά τη λίστα αποτελεσμάτων προωθώντας συναφή έγγραφα σε υψηλότερες θέσεις. Τέλος η υλοποίηση και τα πειραματικά αποτελέσματα της προτεινόμενης διαδικασίας αναζήτησης κατέδειξαν την επιτευξιμότητα και την απόδοσή της, και μας επέτρεψαν συνάμα να εντοπίσουμε τους περιορισμούς της.

Λέξεις κλειδιά: Εξερευνητική Αναζήτηση, Διασυνδεδεμένα Δεδομένα, Εξόρυξη Οντοτήτων, Τυχαίος Περιηγητής, Σημασιολογική Αναζήτηση

Επόπτης: Τζίτζικας Γιάννης
Επίκουρος Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 General Objective

Web Searching is nowadays an integral part of our daily lives. As an indicative example, Google Search Engine [42] now processes over 40,000 search queries every second on average which translates to over 3.5 billion searches per day and 1.2 trillion searches per year worldwide[1]. Nevertheless, in exploratory or undirected search tasks, where searchers often are unfamiliar with the domain of their goals, unsure about the ways to achieve their goals, and/or need to learn about the topic in order to understand how to achieve their goals (e.g., to find a book, buy a product, perform a patent search, etc.), the classic ranked list of results is not enough [145]. In this context, impressions produced out of *overview* information can play an important role not only in understanding an information space, but also in helping users select promising sub-topics for further exploration.

The main objective of this dissertation is to advance exploratory search services by connecting search results with related available semantic information. Towards this direction, in this thesis we propose an entity-based exploratory search process that provides searchers with different perspectives of the search results. The proposed process allows searchers to locate easily and fast even low ranked hits which nonetheless are relevant to their information need, and to get "fresh" (semantic) information related to their search task. Furthermore, we elaborate on whether and how such related semantic information (entities and entity attributes) can be exploited also for producing a better ranking for the displayed results. The proposed approach is general and configurable, and provides a way of making the Web of Data exploitable by end-users.

[1] `http://www.internetlivestats.com/google-search-statistics/` (February 16, 2016)

1.2 Motivation and Vision

Bridging the Web of Documents with the Web of Data

The Web has evolved from an information space of interconnected Web pages to one where both unstructured documents and structured data in various forms coexist. Linked Open Data (LOD) [36] provides a publishing paradigm for structured data in which not only documents, but also data can be a first class citizen of the Web. As of 2014, the LOD cloud contains more than 1 thousand datasets describing millions of things through billions of statements.[2] An important question is how typical Web users, who mainly use keywords in searching, can access and exploit this increasing body of knowledge. Although existing interaction paradigms in Semantic Search hide their complexity behind easy-to-use interfaces, they have not managed to cover common search needs. At the same time, most search methods are appropriate for *focalized search*, i.e., they make the assumption that users can accurately describe their information need using a small sequence of words and that they are interested only in the top hits. However, a high percentage of search tasks are *exploratory* and this is actually the norm in Professional Search. Exploratory search describes information-seeking search tasks that are open-ended, persistent, and multifaceted, and aims to provide searchers with guidance in exploring unfamiliar information landscapes [209]. In such tasks focalized search very commonly leads to inefficient interactions and poor results [145].

Our objective is to enable effective exploratory search services which can bridge the gap between the responses of non-semantic search systems (e.g., Professional Search systems, Web Search engines) and semantic information, i.e., LOD. An important observation is that entity names (like names of persons, locations, etc.) occur in all kinds of artifacts: documents, database cells, RDF triples, etc. Therefore, a basic hypothesis that we investigate is whether and how we can exploit *named entities*[3] for offering a kind of entity-based integration method; entities are used as the "glue" for automatically connecting search results (or documents in general) with structured data. To reach to a widely applicable approach and to tackle the constant evolution of published LOD, we investigate a scenario where these (connection and exploration) services are provided as meta-services, i.e., they can be applied over existing systems, and this entity-based integration is performed at *query time* (for serving fresh results) with no human effort.

Advancing Information Retrieval (IR)

The work in this thesis is interesting also from a "pure IR" perspective. The main objective of Search Engines is to estimate the relevance of documents to the information needs ex-

[2] `http://lod-cloud.net/`

[3] Named entities are elements in text that belong to pre-defined categories (class labels) such as names of persons, organizations, locations, expressions of times, quantities, etc.

pressed in the user's query. However, as stressed by several researchers [70, 195], only the user can determine the true relevance of a document, i.e., the usefulness, pertinence, appropriateness, or utility of that document with respect to the user search intents expressed in her/his query. Relevance is time, situation, and user specific. In assessing the utility of a document users are influenced by many factors that go beyond topical relevance, i.e., whether or not a given document is about the topics covered in the users' queries. Relevance is in fact a multi-dimensional notion that depends on many complex factors related to the search task, to the user, to her/his context, and to the document context as well. As a consequence, in a classic "ranked list" environment, the user either has to inspect a long list of hits to identify the desired one, or to submit several queries and inspect the returned answers. It is also worth mentioning that in Professional Search, it is often unacceptable to miss relevant documents. For instance, this is evidently true in Patent or Bibliography Search. It follows that we need methods that exploit several aspects of the user and search context as well as methods that move beyond the classic "ranked list" approach, considering richer models of interaction and approaches to answering [19]. Thus, an advanced search system should:

- allow easy and fast access even to low-ranked hits by enabling browsing and inspecting the found hits in groups according to various criteria
- offer overviews of the search results
- compute and show descriptions and count information for the various groups, or other aggregated values
- allow gradual restriction of the search results
- offer approaches that exploit one or more aspects of the user and/or search context for improving the search results (e.g., through add-on services like re-ranking, query-expansion, etc.)

Faceted Exploration [185] is a widely used interaction scheme that satisfies the first four of the above requirements. In this thesis, we enrich this interaction paradigm with LOD for offering advanced exploratory search services. In addition, we exploit the search context, specifically LOD entities existing in the search results, for improving the list of results through a probabilistic re-ranking method. However, LOD is big, distributed, interconnected and ever-evolving, meaning that selecting the semantic information that better characterizes the retrieved results, as well as being efficient, is challenging.

1.3 Research Questions

The research questions that this thesis aims at tackling are the following:

- How can one *specify* the "entities of interest" for a given domain (i.e., the entities that are meaningful in the application context and that could be useful on a vertical search scenario)? How to exploit the various knowledge bases that may contain in-

formation about these entities? How to specify the additional semantic information about these entities that is useful to provide in a search context? Can we dynamically configure all the above and then fetch and exploit such semantic information at query-time, i.e., without any prior indexing?

- Since the related semantic information can be huge, how to *select* the categories and entities (as well as entity associations and properties[4]) that better characterize and enhance the search results in the current context? How the selected semantic information should be presented? Can this "semantic analysis" be done at query-time?
- Since common IR techniques often do not work well for more complex search tasks (that go beyond lookup/fact-finding), can we automatically (without any user intervention) *exploit* the related semantic information for improving the retrieved list of results, e.g., for promoting relevant but low-ranked hits containing important (for the search context) semantic information? Under what circumstances such a re-ranking approach is effective and enhances IR or ineffective and thus must be avoided?

1.4 The Approach

To tackle the above challenges, in this thesis we first elaborate on exploiting the LOD at real-time for configuring a Named Entity Extraction (NEE)[5] service and we introduce a generic (abstract) configuration model that, amongst others, can serve provenance information to the output of the entire process. The proposed model allows specifying the entities of interest as well as how to link (retrieve related resources) and enrich (retrieve properties and related entities) the named entities. This enhanced configurability allows the dynamic configuration of a NEE system even while a corresponding service is running.

To explicitly define the semantics of the proposed model, we introduce an RDF/S vocabulary, called "Open NEE Configuration Model". This vocabulary allows a NEE service to describe and publish as Linked Data its entity mining capabilities, but also to be dynamically configured and to exchange such configurations. To allow relating the output of a NEE process with an applied configuration, we propose an extension of the Open Annotation Data Model [189] which also enables a NEE service to expressively describe the annotation data. As a proof of concept, we present `X-Link`, a configurable LOD-based NEE framework that implements the proposed configuration model. Through a task-based user study we examine the usability (ease of configuration) of `X-Link`, while the results of a case study over a publicly available semantic knowledge base demonstrate the feasibility and the limitations of the proposed approach.

To identify the semantic information (entities and properties) that better characterizes

[4]We call *association* a relationship between two entities and *property* an attribute of an entity.

[5]Named Entity Extraction (NEE) is the process of identifying entities in texts and, very commonly, linking them to relevant semantic resources (more in 3.2).

a list of search results, we introduce a probabilistic (Random Walk-based) ranking model. The proposed model exploits both the ranking of the search hits, the extracted named entities and their connectivity, and is also exploited for producing and showing to the user *top-K semantic graphs*. A top-K semantic graph can complement the query answer with useful information regarding the *connectivity* of the identified entities. The introduced ranking model promotes the entities identified in the top-ranked results as well as the semantic information that is linked with many important (highly-scored) entities. We report the results of a survey that demonstrate the usefulness of semantic graphs in a marine-related search context, while the results of a comparative evaluation with other ranking methods demonstrate the effectiveness of the proposed ranking scheme. We also report extensive experimental results that support the feasibility of the entire "semantic analysis" process.

For example and regarding *Marine Search*, by analyzing the snippets of the top-100 results that a Web Search engine returns for the query *yellowfin tuna* (with fish species as the entities of interest and exploiting DBpedia [140] at real-time), in the top-K semantic graphs we get information about the taxonomy of *yellowfin tuna* (family, order, etc.), other tuna species identified in the search results that belong to the same family or conservation status system (e.g., *bigeye tuna*), how all these entities are associated, etc. We get all this information in about 3 seconds without performing any additional query. Figure 1.1 depicts an example of a top-5 semantic graph.

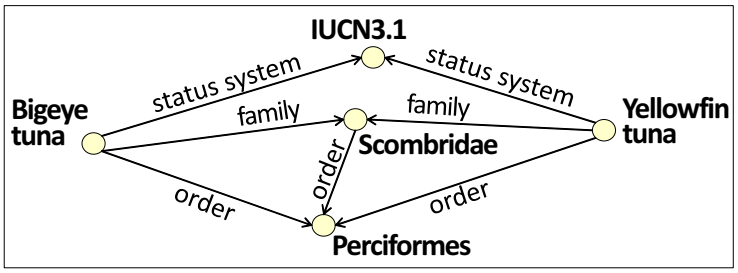


Figure 1.1: An example of a top-5 semantic graph.

Finally, we elaborate on whether and how we can exploit the identified entities and the related semantic information for *re-ranking* the retrieved list of results. The objective is to improve the search results by promoting low-ranked but relevant hits containing important (for the search context) semantic information. Towards this direction, we model the search process as a random walker of the graph defined by both the returned documents and the extracted entities. For analyzing the graph and scoring the nodes, we again exploit the ranking of the documents (or their scores if these are given), the "importance" of the extracted entities within the search context, and their connectivity. The aim is to compute the probabilities the random walker to be in each document-node. These probabilities

actually define a new ranking for the search results. A low-ranked document containing some highly-scored entities will receive a higher score and thus it will be high in the new ranked list of results. Experimental results over datasets of the TREC Clinical Decision Support Track [191] demonstrate that the proposed re-ranking approach can notably improve the list of results returned by a classic IR system by moving relevant but low-ranked hits in higher positions. However, additional semantic information about the entities (properties and related entities) can mislead the random walker and affect negatively the stochastic analysis process.

Figure 1.2 depicts an example from the *medicine* domain. In this example, two entities of type disease were identified in the first three returned hits and thus can be considered important for the search context. However, these entities were also detected in the 20th hit. After applying the proposed re-ranking method, the 20th hit will be ranked higher in the new list of results because it contains two important (highly-scored) entities.

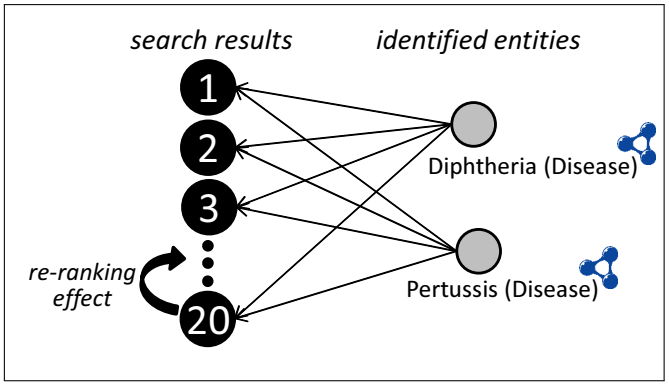


Figure 1.2: Effect of results re-ranking.

1.5 Contributions of this Dissertation

The key contributions of this thesis are the following:

- We introduce a semantic analysis process in which the search results are connected with data and knowledge at query time with no human effort. We present the results of a survey that demonstrate the usefulness of this approach in a marine-related search scenario as well as extensive experimental results that reveal the applicability and efficiency of the proposed search process.
- To tackle configuration issues that arise within the context of the proposed search process, we propose a *generic (abstract) model* for configuring a NEE system that exploits the LOD. To explicitly define the semantics of the proposed model we introduce an RDF/S vocabulary, called Open NEE Configuration Model, while to allow relating the output of a NEE process with an applied configuration, we propose an

extension of the Open Annotation Data Model. We also present `X-Link`, a configurable LOD-based NEE framework that implements the proposed models. Finally, we evaluate `X-Link` in terms of usability (ease of configuring it) and efficiency.

- We introduce a biased probabilistic (Random Walk-based) algorithm for ranking entities and properties that can automatically identify the important (to the search context) semantic information. We present the results of a comparative evaluation with other ranking methods that illustrate the effectiveness (and also quantify the difference) of the proposed ranking scheme.
- We introduce and detail an entity-based Random Walk model for re-ranking the results returned by a search system. The objective is to improve the list of results by moving low-ranked but relevant hits in higher positions. We present the results of an evaluation (using datasets of TREC Clinical Decision Support track) that demonstrate the effectiveness of this model in a medicine-related search context.
- We discuss approaches on how the user can interact with the result of the analysis process for supporting exploratory searching, and we present `X-Search` and `Theophrastus`, two systems we have designed and implemented which support the proposed semantic analysis process, configurability, and interaction model.

Our contribution can be described using keywords as follows:
Exploratory Search, Entity-based Search, Professional Search, Faceted Search, Semantic Analysis of Search Results, Linked (Open) Data, Semantic-driven IR, Exploiting Linked Data in IR, Re-Ranking of Search Results, Stochastic Analysis, Random Walk, Entity Ranking, Configurability, Named Entity Extraction, Entity Linking, Entity Enrichment, Semantic Data Ranking, Semantic Graphs, Semantic Annotation.

In Appendix C, we report the publications resulted from this thesis and we give links to the several systems, prototypes and models we have designed and implemented.

1.6 Outline of Dissertation

The rest of this dissertation is organized in the following way:

- Chapter 2 presents the main motivating scenarios, discusses implementation approaches for offering the proposed functionality, defines the basic concepts, and finally describes the steps of the considered search and configuration processes.
- Chapter 3 provides the background and reviews the related literature regarding Semantic Search, Named Entity Extraction (NEE), Categorizing/Grouping Search Results, Automatically Improving Search Results, and Link Analysis.
- Chapter 4 elaborates on the configurability of a NEE system, introduces the Open NEE Configuration Model and the extension of the Open Annotation Data Model, and presents the `X-Link` framework and its evaluation.

- Chapter 5 describes in detail the probabilistic (Random Walk) models used for ranking the extracted entities, the related semantic data and the search results, and presents evaluation results regarding several aspects of the proposed approach.
- Chapter 6 focuses on how the user can interact with the result of the analysis process and presents the systems `X-Search` and `Theophrastus`.
- Chapter 7 concludes the thesis with a summary of the results and a discussion on possible future directions.
- Appendix A shows screenshots of the configuration forms used in the evaluation prototype of `X-Link`.
- Appendix B gives the formulas of the metrics used in the evaluation of the proposed results re-ranking method.
- Appendix C reports the publications resulted from this dissertation and gives links to the several systems, prototypes and models we have developed.
- Appendix D reports the acronyms used in this dissertation.

Chapter 2
Motivation and Context

In this chapter, we first describe the main motivating scenarios (2.1), we discuss several approaches for offering the proposed functionality (2.2), we define the basic concepts (2.3), and finally we give the steps of the considered search (2.4) and configuration (2.5) processes.

2.1 Motivating Scenarios

Marine Search

Consider the following scenario from the *marine* domain[1]:

A marine biologist seeks for publications about several marine species and submits a query to a marine-related professional search system. Figure 2.1 depicts a screenshot of such a system. We notice that the user has typed the query "tuna species" and has selected to search in a collection of journals. Figure 2.2 shows a screenshot from the results page. We can see that the user gets only a linear list of search results. However, a search system could exploit a NEE service for identifying entities (e.g., names of fish species) that may exist in the answer set. Systems that provide this kind of semantic enrichment of search results present to the user the detected entities (e.g., several species identified in the search results) allowing the user to narrow the search space to a set of results that contain (refer) one or more of the identified entities. Figure 2.3 shows an indicative screenshot of a system that offers this functionality. We notice that the user can see in a left bar several entities (grouped in categories) that have been identified in the search results, like species and countries. Thereby, the user can instantly restrict the search space by clicking on one or more of the shown entity names. For instance, in the example of Figure 2.3 the user has focused on eight results referring the fish species "bigeye tuna".

Furthermore, the system allows the user to semantically explore one of the identified entities. For example, as it is shown in Figure 2.4, by clicking on the small icon next to an entity name, the user can inspect semantic resources related to the selected entity name,

[1]Which is a real scenario related to the iMarine project (EU FP7, 2011-2014, `http://www.i-marine.eu/`).

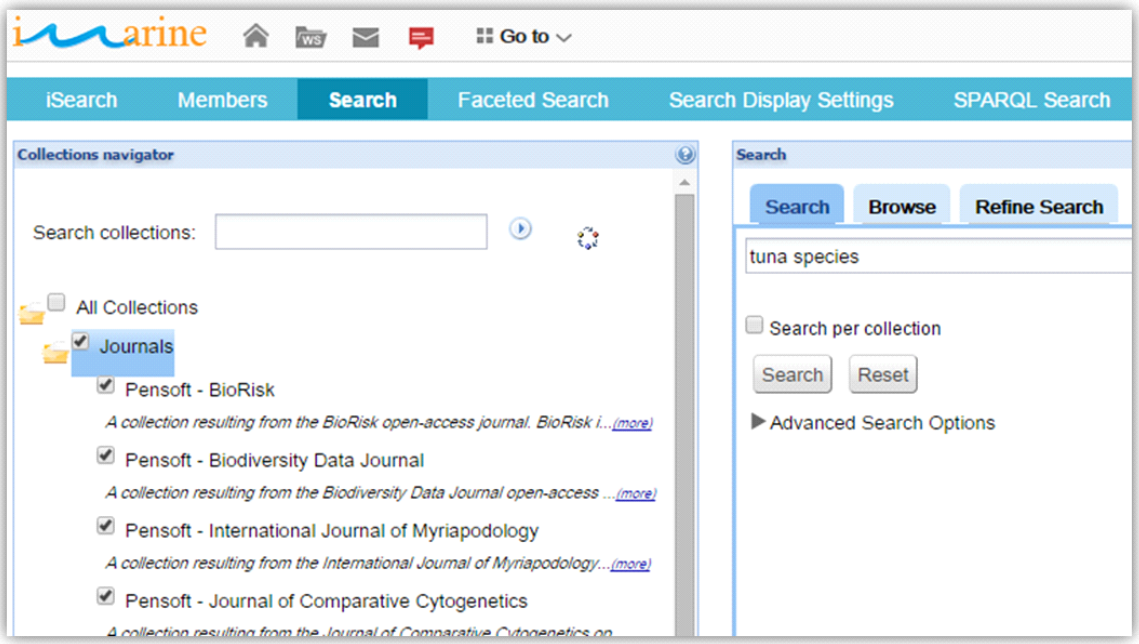


Figure 2.1: Search form of a marine-related search application.

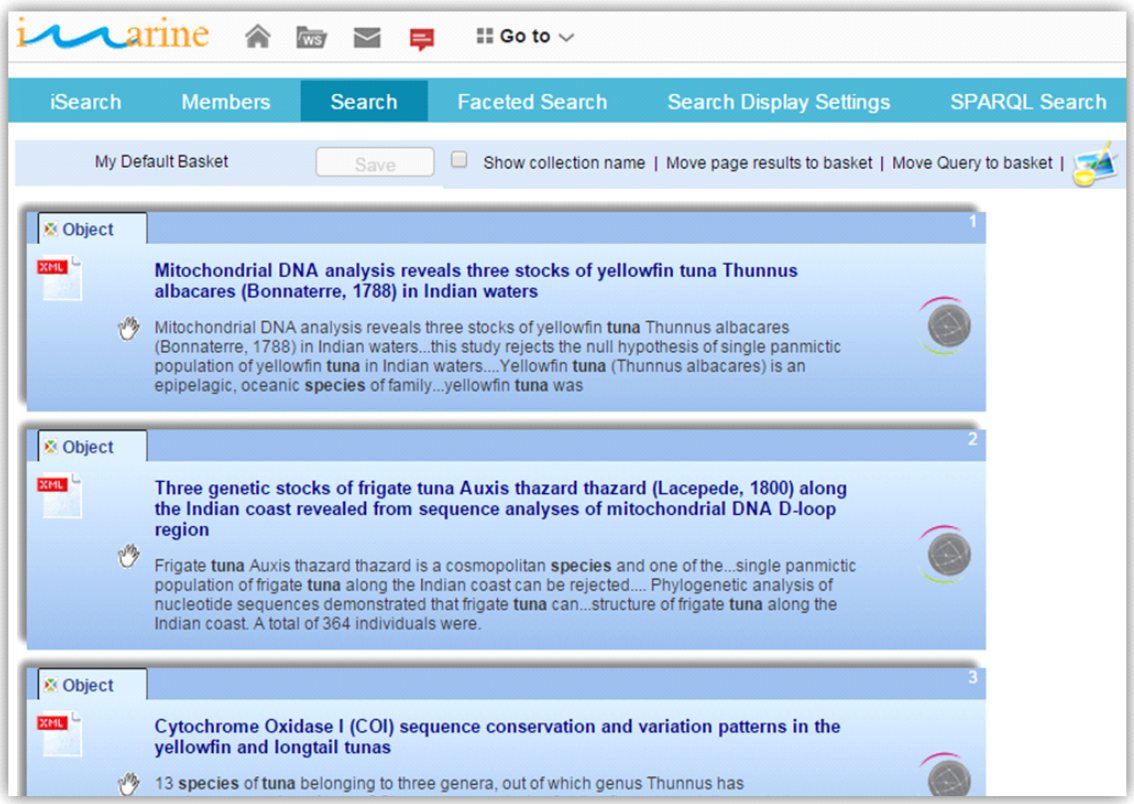


Figure 2.2: Search results of a marine-related search application.

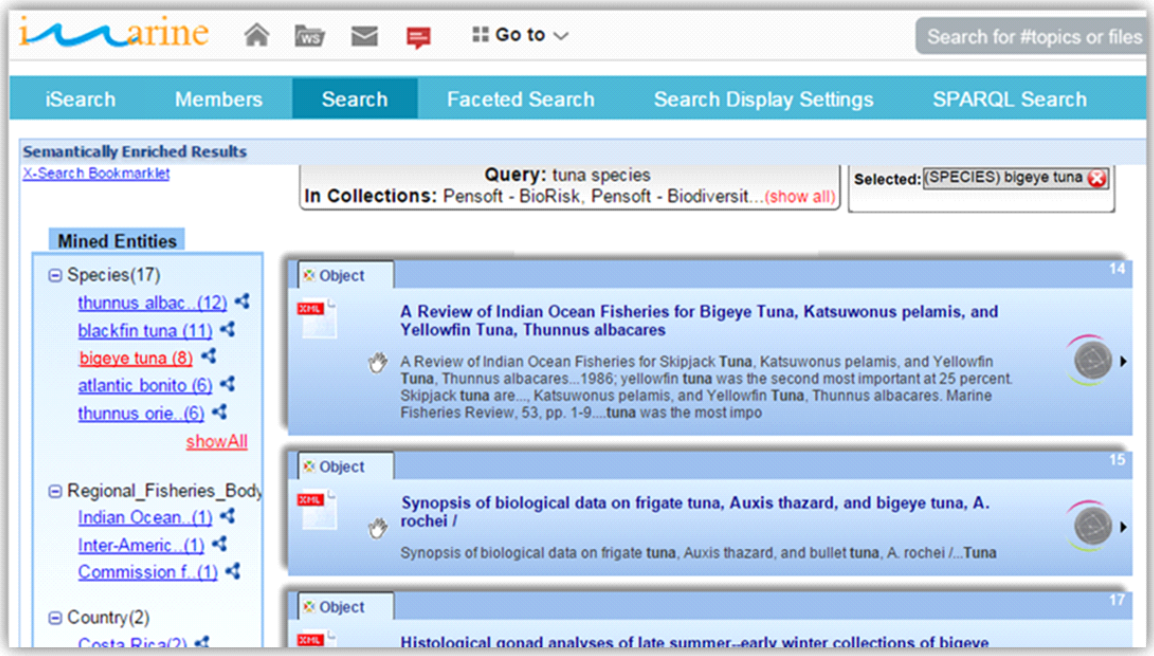


Figure 2.3: Semantically-enriched search results of a marine-related search application.

while by clicking on one of these resources, the user can further explore its properties (as depicted in Figure 2.5). For example, regarding a fish species, the user can see an image, its taxonomy, other related entities, etc. Such functionality allows the user to instantly inspect semantic information that may exist in different places and that may be laborious and time consuming to locate.

However, the number of identifiable entities can be very high and the same is true for the amount of structured information that is available for these entities in one or more knowledge bases (e.g., as LOD). Thus, there is a need of a method that allows *specifying* this "data of interest", i.e., the *entities of interest*, the related *semantic information* about these entities that is useful to provide, and the *knowledge bases* that contain such information.

In addition, the number of entities identified in the search results, as well as the amount of semantic information that has been derived for these entities, can be very high. Thus, there is also the need of *ranking* all this semantic information for identifying and presenting to the user the most important entities, associations and properties.

Configurability

Notice that each community of users (e.g., an organization or an institution) has different and ever-evolving needs, which means that the search system should support different configurations. For instance, scientists in an organization may also want to inspect other

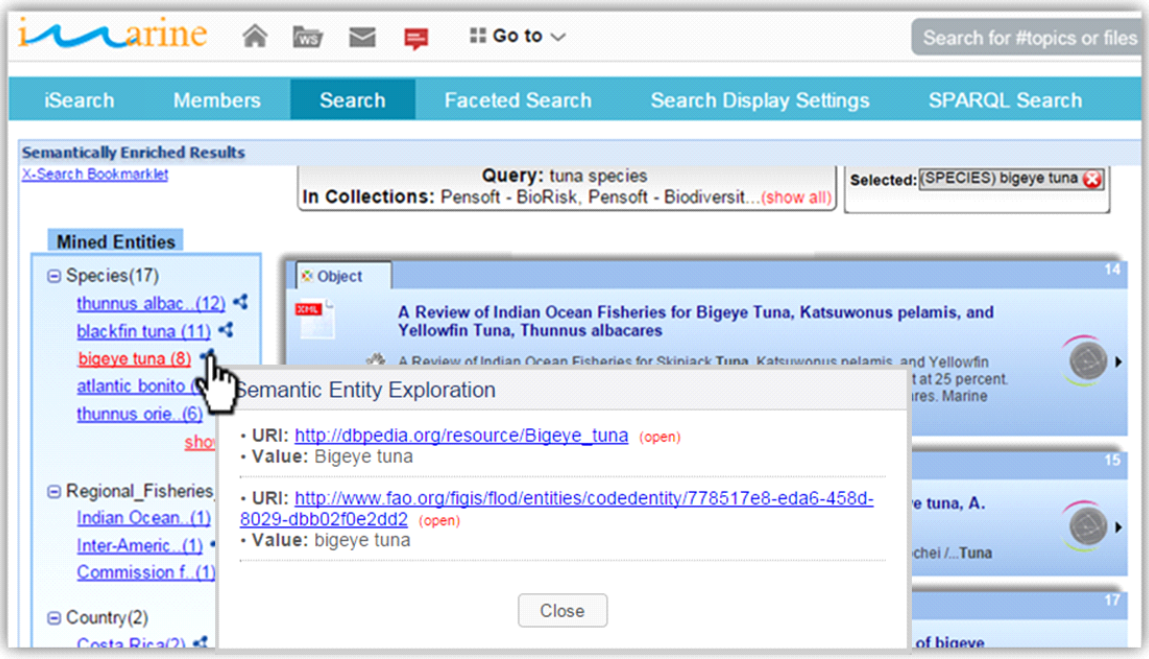


Figure 2.4: Inspecting semantic resources related to an identified entity name.

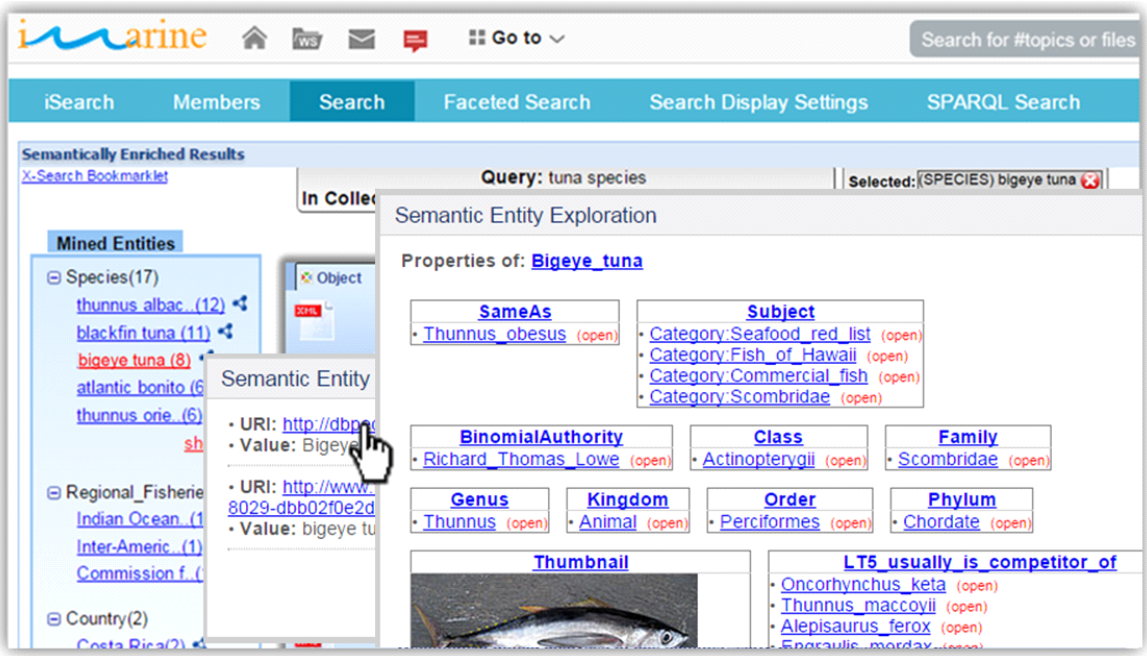


Figure 2.5: Exploring the properties of a semantic resource.

categories of entities in the search results (apart from fish species), e.g., water areas. In addition, different communities/users may want to explore different kind of information for the identified entities; one may want images from DBpedia, others with papers that describe the genome of the species.

For coping with the above requirements, we would like to can easily configure the system (actually the underlying NEE tool) for satisfying the needs of each community of users. In addition, and since the needs of a community constantly change, we would like to be able to *dynamically* change the configuration, e.g., to enable the identification of fish species also in another natural language, to change the information to retrieve for the detected entities, etc. It would be also useful if the search system could dynamically (ideally at query-time) discover the NEE services to use according, for example, to the submitted query. For instance, if a user submits a query requesting documents about water areas, the system could select to use a service that supports identification of water areas.

Since a lot of information about named entities is already available as LOD, its exploitation by a NEE system could bring wide coverage and fresh information. However, existing LOD-based NEE systems (e.g., DBpedia Spotlight [147]) are mainly dedicated to one specific knowledge base which is indexed beforehand, not exploiting thereby the dynamic and distributed nature of LOD. For instance, consider a NEE system that supports a category of entities X. Consider now that a new knowledge base appears which contains plenty of information for entities belonging to X. It would be useful if one could somehow "plug" it in the NEE system (with the less possible effort), enabling thereby the linkage of the identified entities with resources in the new knowledge base. Moreover, the information that existing NEE systems return for the identified entities is not rich enough and cannot be controlled. For example, one cannot configure the properties that are useful for a particular application, e.g., to restrict the properties to only related entities of a specific type, or properties in a specific natural language, etc.

Patent Search

Patent Search is a type of Professional Search and most patent searches (e.g., *patentability* and *validity*) are crucially important for businesses' patent management success. In Patent Search, in many situations, one must look beyond keywords to find and analyze patents based on a more sophisticated understanding of the patent's content and meaning [122]. Thus, services such as entity identification and analysis could become a significant aid to such searches [39]. In that context, the provision of facets that correspond to various kinds of extracted entities can help patent examiners to get an overview, to quickly restrict the search space and to make sense of the query results. The usefulness of NEE in Patent Search is also revealed by the emergence of systems like `quantalyze`[2] in which

[2] `https://www.quantalyze.com`

quantities such as temperatures are spotted in the patent documents, their respective semantic context is identified and the quantity itself is normalized to a standard unit.

However, more kinds of entities should be supported and this must be configurable. For example, consider that a patent searcher must examine the patentability of a submitted patent related to *medicine*. In this case, the interesting types of entities (that it would be useful to identify and provide to the user) may include drugs and diseases. Likewise, for examining the patentability of a patent related to Food and Nutrition, the useful entities may include ingredients, chemical substances, etc. Thereby, it would be useful if the patent searchers could somehow specify the interesting categories of entities related to their search task, and then exploit them for exploring the (often very large) search space.

2.2 Possible Approaches

There are several approaches that could be used for offering the functionality described in the aforementioned motivating scenarios. Some of them are described below.

- **Real-Time NEE over the *snippets* of the top-L hits of the answer.** NEE is performed only over the *snippets*[3] of the top-L (e.g., $L = 1,000$) returned results. This approach is applicable also at a meta (uncooperative) search level where the corpus is not available.
- **Real-Time NEE over the *contents* of the top-L hits of the answer.** Here the full contents of the top-L returned results are downloaded (at real-time) and then NEE is performed. Clearly, this process can take much more time. This approach is applicable also at a meta search level.
- **Offline NEE over the entire corpus.** Here the entire corpus is analyzed *offline* (assuming that the corpus is available) and an appropriate index (or database) is built for using it at run time. Then, for each incoming query, the entities of the top-L hits of the answer are fetched from the index and are given to the user. An important observation is that the size of the entity index in the worst case could be in the scale of the corpus. Also note that this approach cannot be applied at meta search level.
- **Offline NEE over the top hits of the answers of the frequent queries.** Another approach is to process the top hits of the answer for only the *frequent queries* using the approach described in [84, 87, 90], called IOS (from Instant Overview Search). In that case, for each frequent query of the log file (e.g., for those which are used for query suggestions), we compute its answer, fetch the top-L hits, apply NEE, and finally save the results in an index as they should be shown. The benefit of this approach (apart from the *instant response*) is that we do not have to process the entire collection but only the top-L hits of the most frequent queries. This significantly reduces the required computational effort and storage space. The downside of this approach

[3] Snippet is the (usually query-dependant) small text used for representing a search result.

is that if a user submits a query that does not belong to the index then the system cannot offer results. In that case the system could offer to the user the "real-time" approach as it was described earlier. Finally, this approach is applicable also at a meta search level but the index must be periodically refreshed.

In this thesis, we focus and study the real-time approach over the snippets or the full contents of the top hits of the answer.

2.3 Basic Concepts

We define (quite informally) the notion of *"Entities of Interest"*, what a *"Semantic Knowledge Base"* is within our context, and the task *"Semantic analysis of search results"*.

Entities of Interest (EoI) are names of entities belonging to pre-defined categories/classes (such Person, Location, Organization, Chemical Substance, etc.) that are meaningful in the application context.

As an example, the EoI of a marine-related search system may be names of fish species, water areas, countries, etc., while for a medical search system the EoI may include names of drugs, diseases and proteins. These entities may be indicated by the community that is intended to use the search application. The current trend is to publish information about entities in Semantic Knowledge Bases as LOD.

A Semantic Knowledge Base (SKB) is a semantic repository of RDF data accessible as LOD or through a SPARQL Protocol service (called SPARQL endpoint) [12, 13].

Examples of such publicly available SKBs are DBpedia [140], DrugBank [2], and YAGO2 [115]. In general, such SKBs contain plenty of information for several types of named-entities. Notice also that the LOD cloud[4] contains numerous datasets covering many domains.

"Semantic analysis of search results" refers to the task of (a) identifying EoI in the textual results of a search system, (b) retrieving semantic information about the identified entities by exploiting one or more SKBs, and (c) producing a semantic graph related to the search results where nodes correspond to entities and properties of entities, while edges correspond to associations among these entities and properties.

In a nutshell, the input of this process is a set of ranked documents (or ranked document snippets) and the output is an RDF graph characterizing the documents according to a configuration, i.e., according to the specified EoI and SKBs.

[4] `http://lod-cloud.net/`

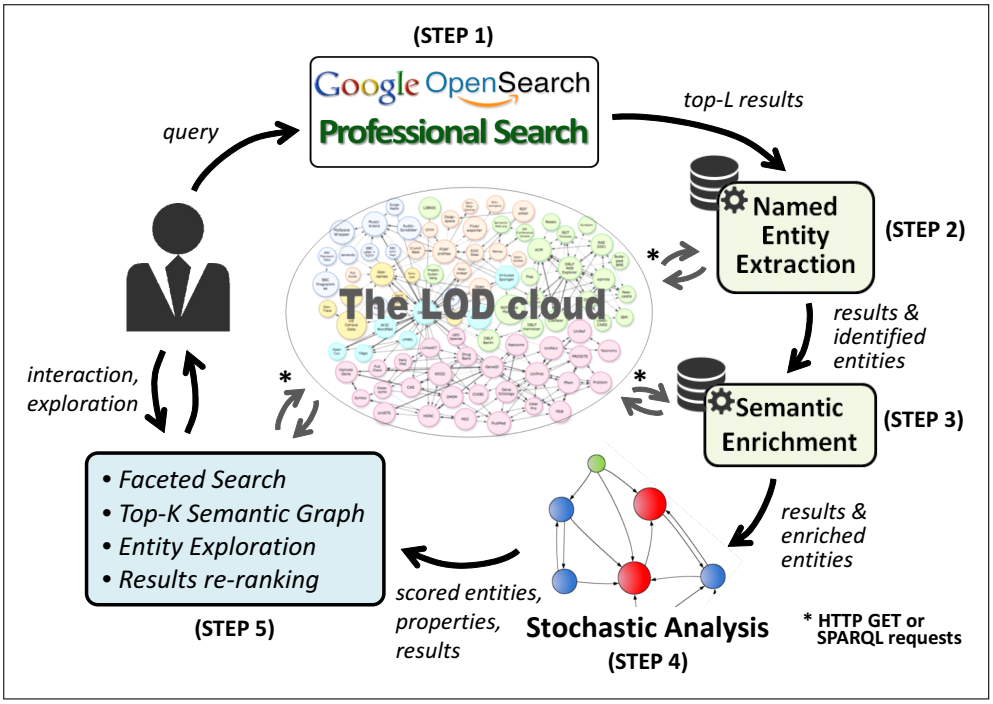


Figure 2.6: Semantic analysis of search results.

2.4 The Considered Search Process

For supporting the motivating scenarios discussed in 2.1, we consider the following process (also depicted at Figure 2.6) which actually analyzes the steps of the *semantic analysis* task:

- `(STEP 1) RESULTS RETRIEVAL:` A user submits a query describing an information need to a search system operating over a collection of documents, e.g., to a Professional Search system or to a Web Search engine. The top-L (e.g., L = 1,000) hits are retrieved.
- `(STEP 2) ENTITY MINING:` The search system uses a component, we call it Semantic Analyzer, which exploits a NEE system for identifying EoI in the textual contents of the retrieved results. A list of identified entities is derived. For configuring the EoI (in a preprocessing step), we exploit the LOD, i.e., we can define that the EoI are the names of entities returned by a given SPARQL query if submitted to a given SKB. Thereby, each entity is accompanied by one or more URIs.
- `(STEP 3) SEMANTIC ENRICHMENT:` Semantic Analyzer retrieves more information about the identified entities by accessing one or more SKBs. The semantic information (properties and related entities) that is interesting in the application context is configured in a preprocessing step and can be expressed through one or more SPARQL queries over the specified SKBs. For instance, by running SPARQL queries we can

retrieve the incoming and outgoing properties of each entity URI.

- `(STEP 4) STOCHASTIC ANALYSIS:` For the identification of the most *important* entities, properties and search results, a probabilistic (Random Walk-based) ranking scheme is used.
- `(STEP 5) EXPLOITATION:` The result of the probabilistic analysis (scored *entities*, *properties*, and *results*) can be exploited in several ways. At first, the results can be visualized and exploited in a faceted and session-based interaction scheme [185] that allows users to restrict their focus or information need gradually. Apart from this, a top-K semantic graph can be provided, allowing the user to gradually increase or reduce the value of K. This graph is very important for showing how the entities are associated. This functionality can be also offered on-demand as a complementary representation of the identified entities. Finally, the search results can be re-ranked (automatically or on-demand) according to their final scores aiming to promote relevant but low-ranked hits containing important (for the search context) semantic information. ⋄

Figure 2.7 depicts a UML sequence diagram of the above process, while the upper part of Figure 2.8 shows possible use cases for the interaction between system and end-user.

Notice that the proposed approach does not analyze the submitted query; it only analyzes the top-L retrieved results. If the user submits a query that semantically is not related to the EoI, e.g., in a marine-related search system the query *everest mountain* is submitted (which is irrelevant to the application context), then the NEE system may not identify entities (Step 2 of the process). Of course, in this case our approach does not produce a semantic graph. Nevertheless, this is an important information which the user can take into account, e.g., for changing her/his query.

2.5 The Considered Configuration Process

The aforementioned search process is fully *configurable*. The user/administrator can configure the EoI, the SKBs that are used for retrieving more information about the EoI, and the type of information to retrieve. Thereby, one can configure it for different domains and contexts. The configuration process is illustrated in Figure 2.9. After deciding the target community/domain, the user/administrator must specify the EoI, the SKBs that contain data about the selected EoI, as well as the semantic information about the EoI that is useful to retrieve and provide. The lower part of Figure 2.8 shows use cases for the interaction between system and administrator.

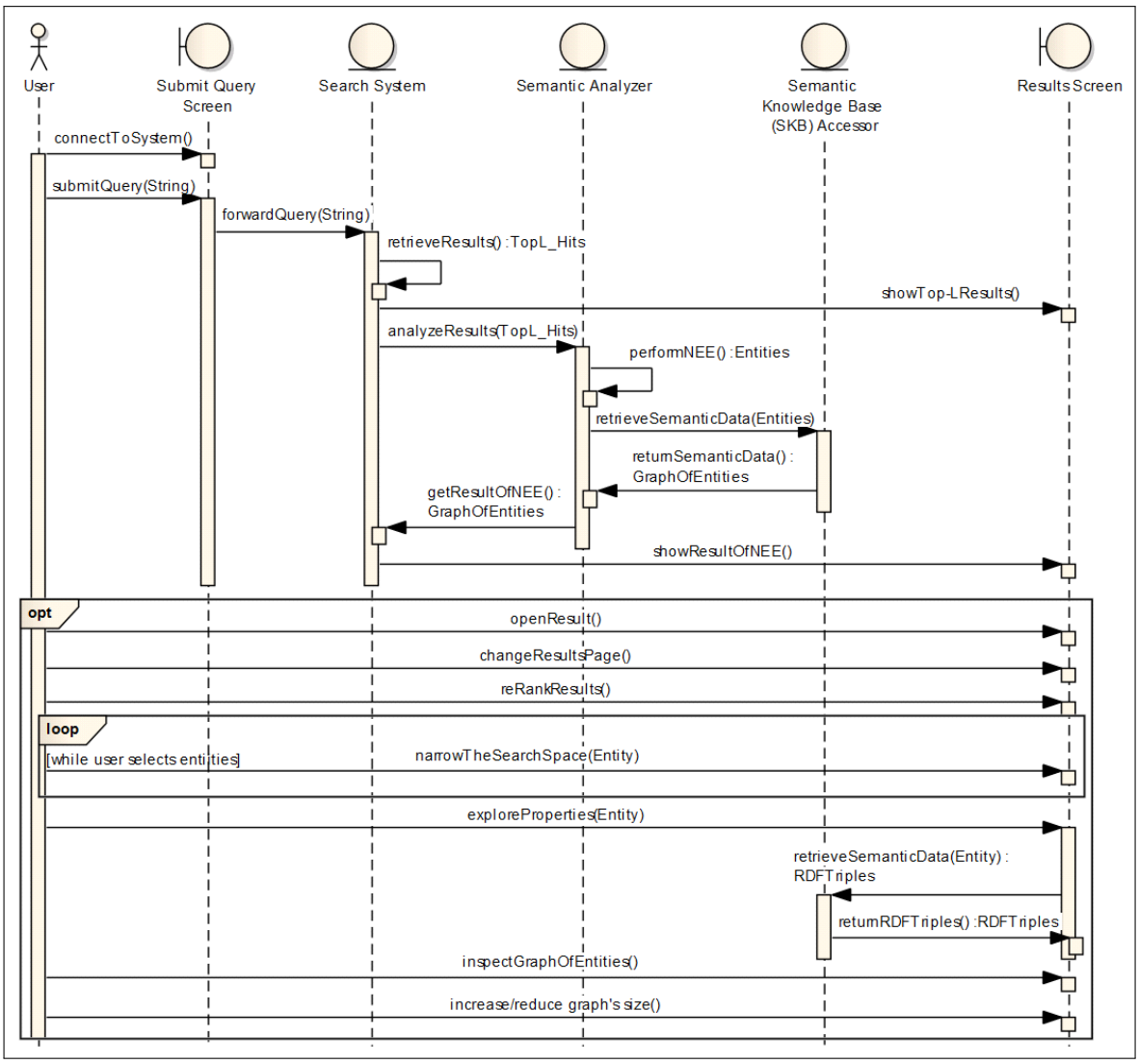


Figure 2.7: Sequence diagram of the *semantic analysis* process.

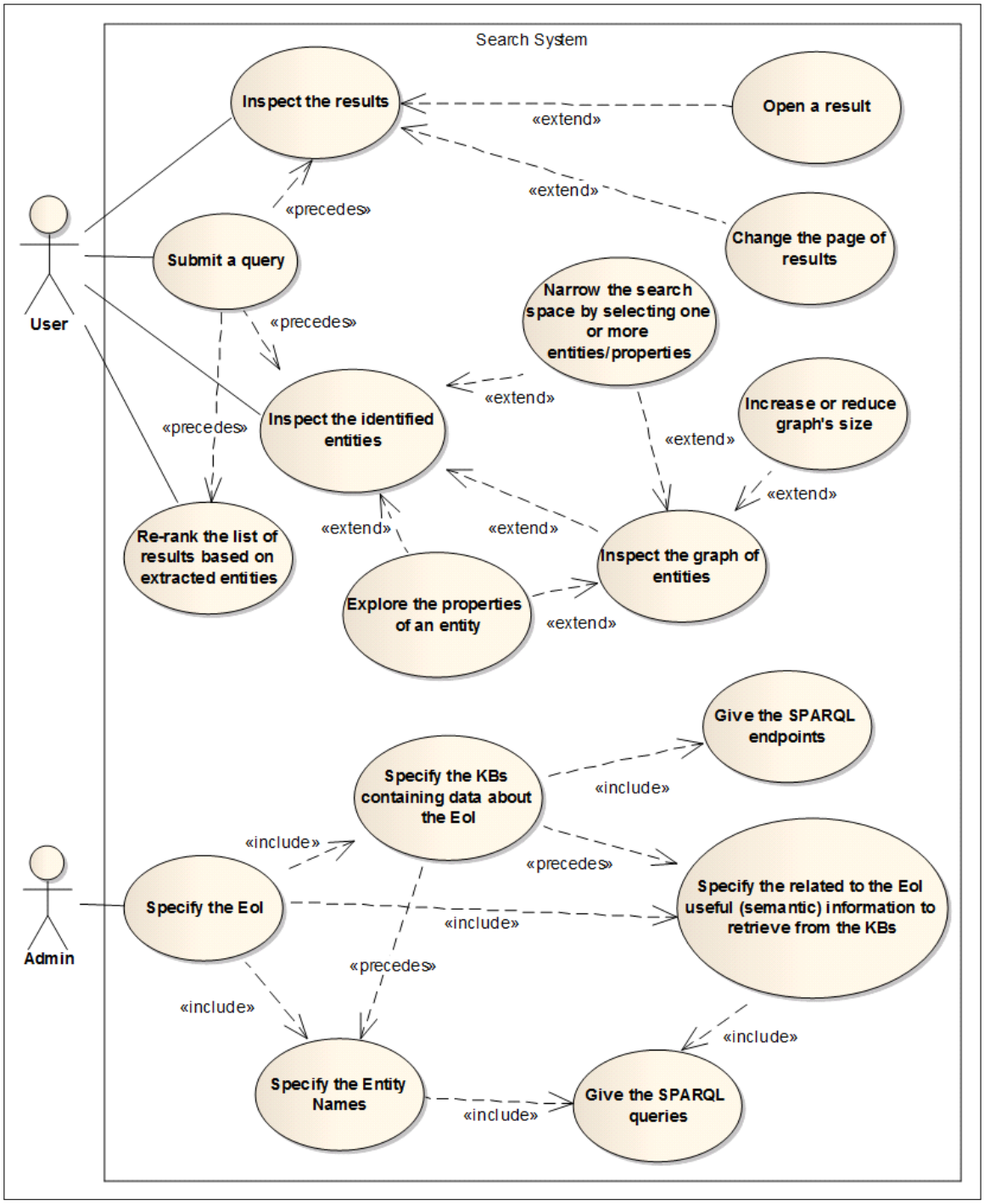


Figure 2.8: Use Case diagram for the interaction between system and end-user (upper part) and between system and administrator (lower part).

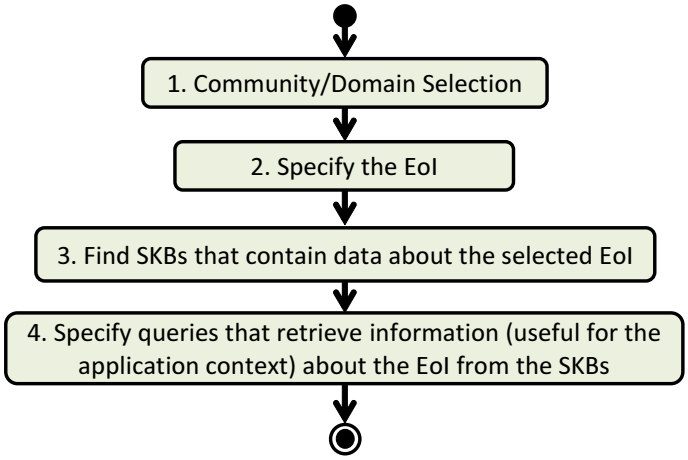


Figure 2.9: The considered configuration process.

Chapter 3
Background and Related Work

In this chapter, we first (in 3.1) provide an introduction to the Web of Data and Semantic Search, and we review the related literature of the two main directions of Semantic Search: Semantic Data Search and Semantic-driven IR. Then (in 3.2), we review the Named Entity Extraction (NEE) process and report several LOD-based NEE tools. Since what we propose can be considered an approach for enriching and grouping the search results, in 3.3 we report works which have shown that such overview information improves the search experience. In 3.4 we discuss works on improving (automatically) the results returned by a search system. Finally (in 3.5), since the Random Walk-based ranking model that we propose is founded on Link Analysis, we present several Link Analysis techniques, we discuss how such techniques can be compared and evaluated, and we review approaches (and how they have been evaluated) that have applied Link Analysis in the Web of Data.

Figure 3.1 depicts how the several concepts, areas and topics involved in this thesis are related to each other. The figure consists of three layers. The lower layer contains the information sources (in our case the Web), the middle layer contains the core services operating over the information sources (in our case IR, NEE and Link Analysis), and the upper layer contains add-on services that aim to enhance the core services (in our case services that enhance IR). Our work exploits Semantic Web and falls into Semantic-driven IR. Furthermore, the proposed approach exploits NEE and Link Analysis for semantically analyzing search results. The outcome of this analysis is exploited for offering add-on services that enhance IR, specifically for grouping the search results (and offering faceted browsing), and for improving them through a re-ranking method.

3.1 Semantic Web and Semantic Search

3.1.1 The Semantic Web (or the Web of Data)

The World Wide Web has enabled the creation of a global information space comprising linked documents. As the Web becomes ever more enmeshed, there is a growing desire for direct access to raw data not currently available on the Web or bound up in hypertext

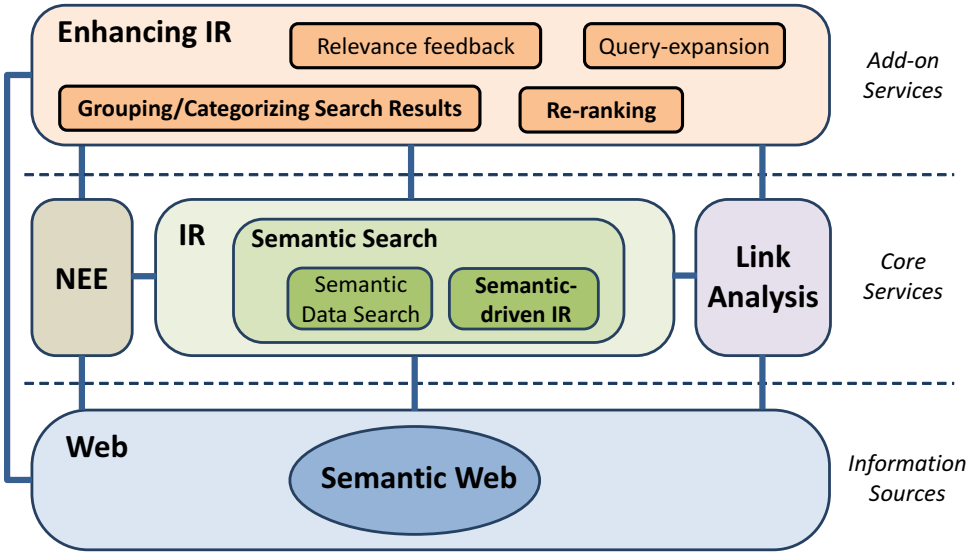


Figure 3.1: Associations among the several concepts, areas and topics involved in this dissertation.

documents. Linked Data provides a publishing paradigm in which not only documents, but also data, can be a first class citizen of the Web, thereby enabling the extension of the Web with a global data space based on open standards, the Web of Data [113].

In detail, Linked Data is about employing the Resource Description Framework (RDF) and the Hypertext Transfer Protocol (HTTP) to publish structured data on the Web and to connect data between different data sources, effectively allowing data in one data source to be linked to data in another data source. The principles of Linked Data were first outlined by Berners-Lee in 2006[1], and provide broad guidance upon which data publishers have begun to realize the Web of Data. This guidance has been extended by technical documents (such as the one by Heath and Bizer [113]) that capture best practices emerging from the Linked Data community and provide recipes on which publishing systems can be based. Figure 3.2 depicts the technology stack that constitutes the Semantic Web[2]. Notice that Linked Data uses a small selection of these technologies.

The Web of Data can be accessed using Linked Data browsers, just as the traditional Web of documents is accessed using HTML browsers [37]. However, instead of following links between HTML pages, Linked Data browsers enable users to navigate between different data sources by following RDF links. This allows the user to start with one data source and then move through a potentially endless Web of data sources connected by RDF links.

[1] `http://www.w3.org/DesignIssues/LinkedData.html`

[2] This figure is a modified version of the Semantic Web technology stack visualization created by Benjamin Nowack (`http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake`).

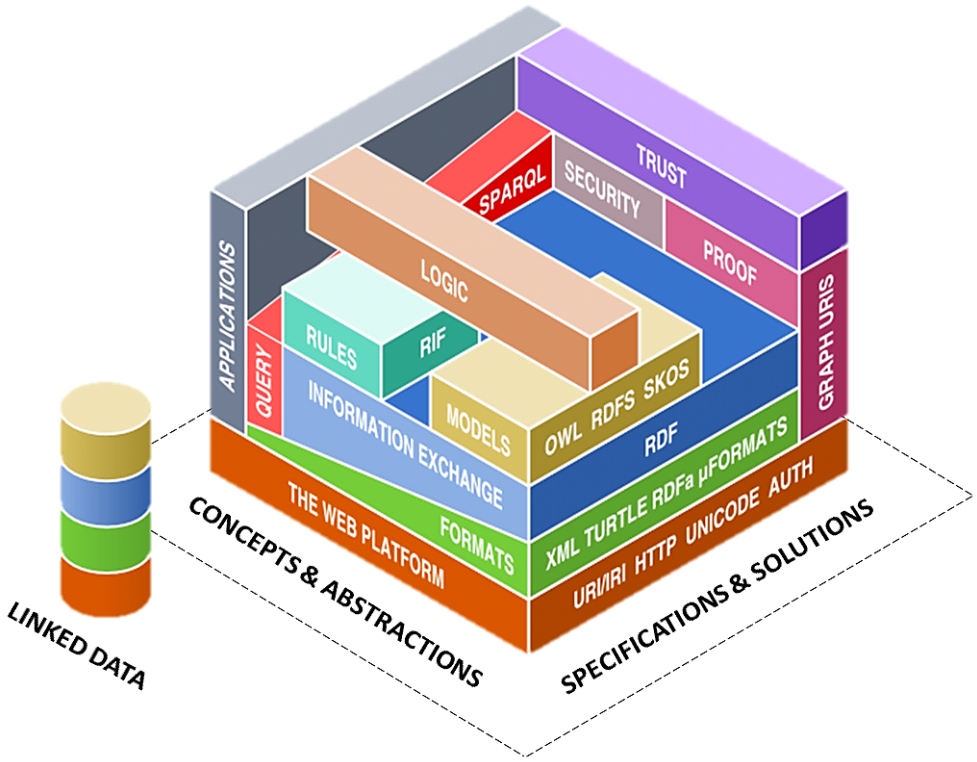


Figure 3.2: The Semantic Web technology stack.[2]

Just as the traditional document Web can be crawled by following hypertext links, the Web of Data can be crawled by following RDF links. Working on the crawled data, search engines can provide sophisticated query capabilities, similar to those provided by conventional relational databases. Because the query results themselves are structured data, not just links to HTML pages, they can be immediately processed, thus enabling a new class of applications based on the Web of Data.

3.1.2 Semantic Search

Due to the increasing size of the Semantic Web we have seen the adoption of ideas from IR to the problem of search in semantic data (*Semantic Search*). In these scenarios the search is focused not on a document collection, but on semantic data, e.g., entities, properties, relationships, etc. The other way around, the emergence of semantic technologies have resulted in standards and tools that allow the representation of domain knowledge at a high level of expressivity. Semantic repositories and reasoning engines have now advanced to a state where querying and processing of this knowledge can scale to large-scale scenarios, thus semantic technologies are posed to provide significant contributions to IR problems.

Forms of Semantic Search

The term "Semantic Search" is highly contested, primarily because of the perpetual and endemic ambiguity around the term "Semantics" [108]. While "Search" is understood to be some form of IR, "Semantics" typically refers to the interpretation of some syntactic structure to another structure, the semantic structure, that defines in more detail the meaning that is implicit in the surface syntax (or even the real world that the syntax describes). Semantics can be given to various parts of the IR model, including the representation of the queries and the documents. This semantics can then be used to process queries against documents, as well as to support users during query construction and the presentation of results.

The classification of Semantic Search approaches is complex, not just because of their diversity, in the sense of how differently this problem has been approached in the literature, but also because of the large number of dimensions involved in the information search task [94]. According to the `SemSearch` International Workshop [197], Semantic Search can be defined through two main directions. First is *Semantic Data Search* which mainly deals with the retrieval of semantic data. The second is *Semantic-driven IR*, the application of semantic technologies to the IR problem. Semantic Data Search includes crawling, storage and indexing of semantic data, semantic data search and ranking, search in multi-data-source, multi-repository scenarios, dealing with vague or incomplete semantic data, infrastructure for searching semantic data on the Web, etc. On the other hand, Semantic-driven IR includes expressive document models, knowledge extraction for building expressive document representation, matching and ranking based on expressive document representation, infrastructure for Semantic-driven IR, and so on. In addition, the research on *ranking* approaches for Semantic Search can be broadly classified into three main categories: *entity ranking*, *relationship ranking*, and *semantic document ranking*. Jindal et al. [121] and Roa-Valverde and Sicilia [180] review and compare (by means of several classification criteria) the main approaches for ranking on the Web of Data.

The context of this thesis as described in Chapter 2 can be considered *Semantic-driven IR*, while the Random Walk-based ranking scheme that we propose falls into the *entity ranking* problem for the search context. Below we review the more important works regarding both Semantic Data Search and Semantic-driven IR.

Semantic Data Search

`Swoogle` [79] has been one of the first and most popular Semantic Web search engines. It runs an automated hybrid crawl to harvest Semantic Web data from the Web, and then provide search services for finding relevant ontologies, documents and terms using keywords and additional semantic constraints. In addition to search, Swoogle also provides

aggregated statistical metadata about the indexed Semantic Web documents and Semantic Web terms, and calculates metrics that allow ontology designers to check the popularity of certain properties and classes.

Rocha et al. [183] present a search architecture that combines classical search techniques with spreading activation [65] techniques applied to a semantic model of a given domain. Specifically, users express their information need in terms of keywords and all concept instances that are related to a given word are obtained. The instances of the underlying SKB are indexed and an instances graph is created. The result given by the traditional search engine is a set of node instances ordered by their similarity with the query, and this set of nodes is supplied to the spreading activation algorithm as the initial set of nodes for the propagation.

`OntoSearch2` [162] is a Semantic Web search engine that allows for keyword search, formal queries and fuzzy queries on a collection of manually submitted OWL ontologies. It relies on scalable reasoning capabilities based on a reduction of OWL ontologies into DL-Lite ontologies.

`Sqore` [203] is a prototype search engine that allows for structured queries in the form of OWL descriptions. Desired properties of entities to be found in ontologies are described as OWL entities and the engine searches for similar descriptions in its collection.

Cheng et al. [56–58] formulate the problem of entity search and emphasize its application on information integration. Specifically, their work applies offline entity extraction and indexing, and online query processing and scoring of the retrieved entities. The authors propose a dual-inversion framework with two indexing and partition schemes for efficient and scalable entity search [57]. Users formulate queries that directly describe what types of entities they are looking for using the prefix `#` (e.g., `#phone`, `#professor`, `#email`, etc). As output users get directly the entity instances that match the query in a ranked order by their matching scores.

`Sindice` [160] is a Semantic Web index or entity look-up service that focuses on scaling to very large quantities of data. It provides keyword and URI-based search, structured-queries, while it relies on some simple reasoning mechanisms for inverse-functional properties.

The `NAGA` semantic search engine [129] builds on a SKB which is organized as a graph with typed edges, and consists of millions of entities and relationships extracted from web-based corpora. A graph-based query language was introduced, which enables the formulation of queries with additional semantic information. The authors also propose a novel scoring model for ranking the query results which formalizes several notions such as confidence, informativeness and compactness.

The `Falcons` search engine [55] offers searching for entities and concepts over RDF data. It maps certain keyword phrases to query relations between entities, and uses (reasoning involving) class hierarchies to quickly restrict initial results.

Li et al. [142, 143] elaborate on *entity-relationship* queries which contain an arbitrary number of predicates on desired entities. The system consists of various offline and online components, including a position-based ranking model for accurate ranking of query answers and an entity-centric index for efficient query evaluation.

`SWSE` [117] is a keyword-based entity search engine which focuses on providing semantic information about the resulting entities rather than only links to the corresponding data sources. It operates over RDF Web data which is automatically gathered by crawlers. SWSE also provides a SPARQL endpoint enabling structured query on the entire collection.

`Watson` [73] provides keyword-based search facilities for finding ontologies and semantic data online, and allows exploring the content of the retrieved semantic documents. It also provides a set of APIs containing high level functions for finding, exploring and querying semantic data and ontologies that have been published online.

Finally, Ciglan et al. [63] propose the `SemSets` retrieval model for answering semantic type keyword queries from a SKB. The model exploits and combines traditional document-based IR, link structure of the semantic data and entity membership in *semantic sets* (groups of semantically related entities), in order to compute ranks of distinct resources given a user keyword query.

Semantic-driven IR

The integration of semantics in IR has attracted the interest of both the industry and the research community.

Understanding user's query

`Google Knowledge Graph (GKG)` [5] has evidenced the increasing interest of exploiting semantic data in Web searching. `GKG` tries to understand the submitted query and presents a semantic description (in a right panel) of the entity that the user is maybe looking for. However, for a bit more complex queries the user does not get any semantic information. For instance (and for the time being), for the query *"Barack Obama and Honolulu"*, `GKG` does not return any semantic information, although *Honolulu* is the birth place of *Barack Obama*, i.e., the two entities are highly related. Instead, our approach analyzes the search results and therefore can identify many entities, while it can show how these entities are associated.

Yahoo's `Spark` system [38, 149] is a recommendation engine that links a user's initial query to an entity within a knowledge base and provides a ranking of the related entities. `Spark` extracts several signals from a variety of data sources, including Yahoo! Web Search, Twitter and Flickr. These signals are combined with a machine-learned ranking model to produce a final recommendation of entities to user queries. The main difference of our work compared to `Spark` is that we analyze the returned results, not the submitted query. This can reveal hidden relationships among a searchable entity and some other entities

identified in the list of results (i.e., relationships that do not exist in the underlying knowledge base). Furthermore, if the query does not correspond to an entity in the knowledge base, `Spark` does not return related entities. On the contrary, since we do not analyze the submitted query, our approach can return entities (and associations among them) for any type of query.

Creating summaries

`EntityCube` [3] is a research prototype by Microsoft for exploring real-world entities such as persons, locations and organizations. `EntityCube` generates *summaries of entities* from billions of public Web pages and allows the exploration of their relationships. For example, users can use it to find an automatically generated biography page and social-network graph for a person, or to discover a relationship path between two persons.

Enriching/Constructing snippets

Haas et al. [106] establish the notion of *enhanced search results* that extend Web search snippets to include multimedia objects such as images and video, intent-specific key value pairs, and elements that allow the user to interact with the contents of a Web page directly from the results page. The generation of the enhanced snippets is enabled by exploiting the Semantic Web and advances in IR.

Alsarem et al. [21] present a semantic-snippet generation system called `ENsEN`. `ENsEN` is based on an algorithm for ranking LOD entities identified in a set of Web Search results. This ranking algorithm exploits both textual data associated to the identified entities (abstract from DBpedia, text window from Web page), Linked Data (for associating the entities), and the ranking of the retrieved results.

Exploiting ontologies in IR

The work by Varelas et al. [205] investigates approaches for computing the semantic similarity between concepts in an ontology using their relationships. Building upon this idea, the authors propose an IR method which can detect similarities between documents containing semantically similar but not necessarily lexicographically similar terms.

`ESTER` [30] is a modular search system that combines full-text and ontology search. It supports entity recognition by assigning words or phrases in the corpus to the entities from the ontology they refer to. It supports two basic operations: prefix search and join. Both of these are supported efficiently with a compact index.

Antonio Rinaldi [176] proposes an ontology-driven approach for semantic IR. The approach is based on ontologies which are used for dynamically building a semantic network. This network is based on linguistic properties and it is combined with a metric of semantic relatedness. The proposed methods, metrics, and techniques are implemented in a system called DySE (Dynamic Semantic Engine). DySE implements a context-driven approach in which the keywords are processed in the context of the information in which they are retrieved in order to solve semantic ambiguity and to give a more accurate re-

trieval based on the user interests.

Fernandez et al. [94] explore the use of semantic information in IR to support more expressive queries and more accurate results. The proposed system takes as input a user's natural language query. The query is processed by an ontology-based question answering system and pieces of relevant ontological knowledge are returned. Then, the system retrieves and ranks the documents containing this information. The final output of the system consists of a set of ontology elements that answer the user's question and a complementary list of semantically ranked relevant documents.

Semantic Analysis of Search Results

A very recent work by Schuhmacher et al. [190] presents a quite similar (to our approach) entity-based process to analyze search results, and tackles the same problem of retrieving a ranking of entities in response to a query submitted to a search system. Specifically, given a user-provided textual query and a background corpus with query-relevant documents that however include entity annotations, the goal is to rank the entities by relevance to the query. The authors propose a learning-to-rank approach and study different features that use documents, entity mentions, and information from SKBs. To evaluate their approach, the authors created their own dataset (using queries from well-known document retrieval benchmarks and human relevance judgments), and tested the use of different features for ranking the entities. Although this paper follows our work (as introduced in [83], [89] and [85]), its difference compared to our approach is not discussed/evaluated. The main differences of our approach are the following:

- our ranking model adopts a quite different (probabilistic) model that exploits the ranking of the returned documents
- our ranking model does not require a training phase (given a configuration, it can be directly applied over a search system)
- the process that we propose applies *real-time* and *configurable* LOD-based NEE over the textual contents of the results returned by any search system (our approach does not require the underlying corpus to have been pre-annotated)

The semantic-snippet generation system introduced by Alsarem et al. [21] (discussed earlier) also elaborates on the same problem of ranking entities identified dynamically in a set of search results. In this work, the authors introduce an entity ranking algorithm, called `LDRANK`, which combines the biased probabilistic method that we propose in 5.3 and a variation of the Singular Value Decomposition (SVD) algorithm that exploits textual data associated with the identified entities. The approach as well as evaluation results are analyzed in 5.5.5.

3.1.3 Our Approach

As regards *Semantic Data Search*, existing works focus on retrieving and ranking resources from a SKB, and users get as output directly resources that match the query, not documents. Therefore, such works are quite distant from the way users search for information. On the contrary, the approach that we propose does not change the (user-friendly) way users search for information, but acts as a *mediator* between any search system and semantic information; users still get documents as search results, but also get and interact with semantic information (Linked Data) that is highly related to the returned results.

As regards *Semantic-driven IR*, our work is the first that proposes an entity-based and real-time process to semantically analyze search results and which tackles the problem of ranking entities (and related semantic data) extracted dynamically from a list of retrieved results in response to a keyword-based query submitted to an IR system. Our objective is to bridge the gap between the responses of non-semantic search systems and semantic data. Towards this direction, we exploit named entities identified dynamically in a set of retrieved results for offering a kind of *real-time* and *entity-based* integration of documents with structured data (Linked Data). We show how semantic information related to a list of search results can be exploited both for offering advance exploratory search services and in pure IR for improving the list of results. Furthermore, we pay particular attention on the configurability of the proposed analysis process with the aim to offer an approach that is applicable to many different contexts and domains.

3.2 Named Entity Extraction

Named Entity Extraction (NEE), also known as Named Entity Recognition (NER) and Semantic Annotation, is the process of identifying entities in text belonging to a set of predefined categories (class labels) such as Person, Location, Organization, etc. This task usually includes the Entity Linking process which tries to link the named entity with a resource (reference) in a SKB[3]. Entity Linking is also considered a way of Named Entity Disambiguation (NED), since a resource (e.g., a URI or a Wikipedia page) can determine the identity of an entity.

NEE is useful in several tasks, e.g., in question answering [153], for annotating (Web) documents [91, 137], as well as in the context of our approach, i.e., in semantic analysis of search results [83, 88]. We should also stress that the importance of NEE, especially for the Semantic Web, is justified by the fact that the Semantic Web realization highly depends on the availability of metadata (structured content in general) describing Web content, defined through a formal semantic structure. Thus, a major challenge for the Semantic Web is the extraction of structured data through the development of automatic NEE tools.

[3]From now on, we consider as NEE the process that includes both NER and Entity Linking.

There is a plethora of non LOD-based NEE tools like `Wikipedia Miner` [150], Yahoo! `Content Analysis API` [16] and `TagMe` [95]. Since the approach that we propose is based on LOD, below we first discuss the most relevant LOD-based NEE tools of general purpose (3.2.1), we then report some semantic annotation systems tailored for the *life sciences* domain (3.2.2), and finally we discuss the main differences of our approach (3.2.3).

3.2.1 LOD-based NEE Tools of General Purpose

DBpedia Spotlight [147] is a REST API tool for annotating mentions of DBpedia resources in text, providing a solution for linking unstructured information sources to the LOD. It finds and returns entities in a text, ranks them depending on how relevant they are with the text content, and links them with URIs from DBpedia. The result of entity extraction can be stored into various forms (HTML, XML, JSON or XHTML+RDFa). As regards configurability, users can provide whitelists (allowed) and blacklists (forbidden) of resource types for annotation. The available types are derived from the class hierarchy provided by the DBpedia Ontology. In addition, the interesting resources can be constrained using a SPARQL query. However, this configurability allows only the specification of the interesting resources from the existing ones; the user/administrator cannot add a new category of entities (e.g., describing resources coming from another SKB), update a category or specify how to link and enrich the identified entities.

AlchemyAPI [1] is a Natural Language Processing (NLP) service which provides a scalable platform for analyzing Web pages, documents and tweets along with APIs for integration. The retrieved entities are ranked based on their importance in the given text and the results can be stored as JSON, Microformats, XML and RDF (using a dedicated schema[4]). In addition, the named entity extractor is able to disambiguate the detected entities, link them to various datasets on the LOD and resolve co-references.

OpenCalais: Calais [7] is a toolkit that allows incorporating semantic functionality within a blog, content management system, website or application. The OpenCalais Web Service automatically creates semantic metadata for the submitted content. Using NLP, Machine Learning (ML) and other methods, Calais analyzes a document, finds the entities within it and gives them a score based on their text relevance. The results can be saved as JSON, RDF (using a dedicated schema[5]), Microformats, N3 or simple text. In addition, it supports automatic connection to the LOD.

AIDA [217] is a framework and online tool for entity detection and disambiguation. Given a natural-language text, AIDA maps mentions of ambiguous names to entities registered in the YAGO2 SKB [115]. It accepts plain text, HMTL as well as semi-structured inputs

[4] `http://rdf.alchemyapi.com/rdf/v1/s/aapi-schema`
[5] `http://www.opencalais.com/files/owl.opencalais-4.3a.xml`

like tables, lists, or short XML files. AIDA is centered around collective disambiguation exploiting the prominence of entities, similarity between the context of the mention and its candidates, and the coherence among candidate entities for all mentions. The results can be stored in JSON.

Wikimeta [15] is a NLP semantic tagging and annotation system that allows incorporating semantic knowledge within a document, website or content management system. It tries to link each detected named entity with an entity in DBpedia based on a disambiguation process described by Charton et al. [53]. Wikimeta API is compliant with REST and the responses are formatted in XML and JSON. The datasets used to train the NLP tools of Wikimeta are derived from Wikipedia.

Lupedia [6] uses a gazetteer which is a list of surface forms associated to a subset of entities in DBpedia and LinkedMDB (a dataset that contains movies descriptions). The default configuration takes the longest sequence of consecutive words that corresponds to an entry in the gazetteer and annotates it with the corresponding entity in the SKB. The results can be stored in HTML, JSON, RDFa or XML.

3.2.2 Life Sciences-tailored Annotation Tools

Domeo Annotation Toolkit [62] is a collection of software components that enables users to create, share and curate *ontology-based* annotations for online documents. It supports fully automated, semi-automated, and manual *biomedical* annotation with full representation of the provenance of annotations, as well as personal or community annotations with authorization and access control. Annotations are represented using the Annotation Ontology (AO) RDF model [61]. Its user interface is an extensible Web component which enables direct biomedical annotation of HTML and XML documents. Domeo performs entity mining and accesses ontologies as well as other automated markup facilities via Web Service calls.

Utopia Documents [23] is a desktop application for reading and exploring PDF files like scientific papers. By exploiting domain-specific ontologies and plugins, it links both explicit and implicit information (of biological or chemical interest) embedded in the articles to online resources. Utopia Documents allows editors and authors to annotate terms with definitions from online resources and allows readers to easily find these definitions. It also transforms static tables and figures into dynamic, interactive objects and simplifies the process of finding related articles by automatically linking references to their digital online versions. Via its plugins it has access to a wealth of bioinformatics data: each plugin uses appropriate client libraries to access web-service endpoints and other remotely accessible resources, such as relational databases and RDF stores.

The **NCBO Annotator** [123] is an *ontology-based* Web Service for annotating textual biomed-

ical data with biomedical ontology concepts. The NCBO Annotator provides access to almost 200 ontologies from BioPortal and UMLS, and is an alternative to manual annotation through the use of a concept recognition tool. The annotator also leverages the structure of the ontologies to expand annotations. Such annotations allow unstructured free-text data to become structured and standardized, and also contribute to create a biomedical Semantic Web that facilitates data integration.

Whatizit [173] is a text processing system that allows a user to perform text-mining tasks. Whatizit identifies *molecular biology terms* and links them to related (publicly available) databases. The identified terms are wrapped with XML tags that carry additional information, such as the keys to the databases where relevant information is kept. Any vocabulary can be integrated into Whatizit as a pipeline and also several vocabularies can be integrated in a single pipeline. Examples of already integrated vocabularies are *Swissprot*, *Gene Ontology*, *NCBI's taxonomy* and *Medline Plus*.

3.2.3 Our Approach

The main difference of our approach is that we focus on *configurability*. Specifically, we propose a method which exploits the dynamic and open nature of LOD for specifying the EoI as well as how to link and enrich the identified entities. This enhanced configurability allows the dynamic configuration of a NEE system even while a corresponding service is running. On the contrary, the configuration of existing systems is a laborious task even for persons with computer science background and requires many technical skills. Other differences include:

- The proposed approach does not index semantic information (e.g., RDF triples or URIs); it just indexes plain lists of entities (gazetteers) regarding only the supported categories of entities. This makes the NEE system *lightweight* and *portable*.
- By adopting the proposed approach, a NEE system can retrieve at real-time more information about the identified entities (properties and related entities) and this is configurable. On the contrary, the majority of the existing systems return only the corresponding URIs and maybe some related Web pages.
- Existing systems do not describe their entity mining capabilities in a standard format.

Note also that the Open NEE Configuration Model that we propose, as well as the extension of the Open Annotation Data Model, can be applied by existing systems. Furthermore, having open and exchangeable configurations offers many benefits including:

- *Exchangeability and Portability.* Configurations can be exchanged by users/communities, e.g., for annotating different corpora of documents using the same configuration, i.e., the same categories, lists of entities, SKBs, etc. In addition, the availability of a model like the one that we propose enables a NEE service to offer an API

that accepts and uses such configurations, while the result of the annotation process can be in a standard format, allowing its further exploitation in several contexts.

- *Aggregation and Integration of multiple configurations.* A common model allows someone to collect such configurations (provided by different NEE services) and then, by querying them, to select those services that satisfy the needs of the intended application.
- *Benchmarking.* Common configurations would allow comparative evaluation of different NEE systems, e.g., with respect to efficiency, effectiveness of entity disambiguation, etc.
- *Extensibility.* The expression of the model as an RDF Schema allows someone to extend it by exploiting also other vocabularies.

3.3 Categorizing/Grouping Search Results

The idea of enriching the classical query-and-response process of current search engines, with static and dynamic metadata for supporting *exploratory search* was proposed in [163] and it is described in more detail (enriched with the results of a user-based evaluation) in [164]. In that work the notion of *dynamic metadata* refers to the outcome of results clustering algorithms which take as input the snippets of the top returned hits, where snippets are query-word dependent and thus they cannot be extracted, stored and indexed a-priori. Note that the result of NEE if applied at real-time over the textual snippets or the full contents of the top returned results also falls into the case of *dynamic metadata*.

The *categorization* of search results is a useful feature as it has been shown by several user studies. For instance, Kaki and Aulathe [127] have shown that the search speed is improved and users get more relevant results. Another user study [126] have illustrated that categories are successfully used as part of users' search habits. Specifically, users are able to access results that are located far in the ranked list and formulate simpler queries in order to find the needed results. In addition, the categories are beneficial when more than one result is needed like in an *exploratory* or *undirected search* task. According to Kules et al. [136] and Wilson et al. [211], such overview information can play an important role not only in understanding an information space, but also in helping users select promising sub-topics for further exploration.

Grouping the search results is not only useful to public Web Search, but is also particularly useful in Professional Search (search in the workplace), e.g., in industrial research and development [134]. A user study by Pratt and Fagan [170] has indicated that categorizing dynamically the results of a search process in a *medical* search system provides an organization of the results that is clearer, easier to use, more precise, and in general more helpful than simple relevance ranking. As we also discussed in our motivating scenarios (2.1), in Professional Patent Search, in many cases one has to look beyond keywords to find

and analyze patents based on a more sophisticated understanding of the patent's content and meaning [122]. Notice also that Professional Search often requires a long time. For instance, in the domain of Patent Search, persons working in patent offices may spend days for a particular patent search request. The same happens in Bibliographic and Medical Search. Thus, entity identification, and in general text analysis technologies, could become a significant aid to professional searchers and can be seen as becoming the cutting edge of IR science [39].

Analogous results have been reported for searching over collections of *structured* artifacts, e.g., ontologies. For instance, the work by Allocca et al. [20] have showed that making explicit the relationships between ontologies and using them to structure (or categorize) the results of a Semantic Web search engine leads to a more efficient ontology search process.

3.4 Automatically Improving Search Results

There is a large body of literature on improving automatically the results returned by a search system. The works can be classified in three main categories: *automatic query expansion*, *pseudo-relevance feedback*, and *re-ranking*.

3.4.1 Automatic Query Expansion

Automatic query expansion is the process of reformulating a query to improve retrieval performance without any user interaction. Specifically, the original query submitted by a user is automatically expanded with other words that best capture the actual user intent, or that simply produce a more useful query, i.e., a query that is more likely to retrieve relevant documents. The computational steps involved mainly include data acquisition and preprocessing, candidate feature generation and ranking, feature selection, and query reformulation.

The work by Carpineto and Romano [49] surveys approaches of automatic query expansion. The authors classify the approaches into five main groups according to the conceptual paradigm used for finding the expansion features: linguistic methods (e.g., using word stems and morphological variants [34], or syntactic analysis [193]), corpus-specific statistical approaches (e.g., making use of latent semantic indexing [165] or interlinked Wikipedia articles [151]), query-specific statistical approaches (e.g., using document summaries of top retrieved documents [52]), methods that exploit search log analysis (e.g., using top results from past queries [97], or extracting terms from clicked results [175]), and methods using Web data (e.g., using anchor texts [135] or categories of Wikipedia articles [214]). More recent works focus on multilinguality [104, 125] or exploit ontologies [25, 54], while the work by Dalton et al. [72] enriches the query with features from

query entities and their links to knowledge bases, including structured attributes and text.

3.4.2 Pseudo-Relevance Feedback

Relevance feedback analyzes the results that are initially returned from a given query and uses information (provided by the user) about whether or not those results are relevant to perform a new query. Pseudo-relevance feedback automates the manual part of relevance feedback so that the user gets improved retrieval performance without any interaction.

Pseudo-relevance feedback has been widely used in IR and has been implemented in different retrieval models: e.g., vector space model [182], probabilistic model [181], language model [138]. Yu et al. [218] introduce a Web page segmentation algorithm to assist pseudo-relevance feedback in Web IR. Lee et al. [139] propose a cluster-based resampling method to select relevant documents based on the relevance model. The main idea is to use document clusters to find dominant documents from the initial retrieval set, and to repeatedly feed the documents to emphasize the core topics of a query. Tao and Zhai [194] present a method based on statistical language models. The idea is to integrate the original query with feedback documents in a single probabilistic mixture model and regularize the estimation of the language model parameters in the model so that the information in the feedback documents can be gradually added to the original query. Cao et al. [46] propose the integration of a term classification process to predict the usefulness of expansion terms, while Xu et al. [214] exploit Wikipedia entity pages for query-dependent pseudo-relevance feedback. Finally, more recent works focus on temporal pseudo-relevance feedback for microblog search [152, 210].

3.4.3 Re-ranking

Re-ranking aims to improve the original list of results by reordering the returned hits. Chidlovskii et al. [59] present a collaborative re-ranking system architecture for integrating user and community profiling to the information search process. Kanhabua and Nrvg in [128] propose a number of methods to determine the time of queries using temporal language models and show how to increase the retrieval effectiveness by using the determined time of queries to re-rank the search results. Zhuang and Cucerzan [222] introduce a novel method, called Q-Rank, to effectively refine the ranking of search results for any given query by constructing the query context from search query logs. Cui et al. [68] and Zitouni et al. [223] focus on re-ranking image search results using visual similarity. In the same context, Jain and Varma [118] propose a re-ranking approach based on aggregated click data, while a more recent work by Wang et al. [208] uses query-specific semantic signatures.

3.4.4 Our approach

Our approach is classified as *automatic re-ranking*. However, we propose a quite difference approach in which we elaborate on whether and how we can exploit LOD entities (extracted from a list of results returned by a search system) for re-ranking the results, aiming to promote relevant but low-ranked hits containing important (for the search context) semantic information. The proposed approach is detailed in 5.4.

3.5 Link Analysis

In network theory, Link Analysis is a data-analysis technique used to evaluate relationships (connections) between nodes. Relationships may be identified among various types of nodes (objects), including organizations, people and transactions. Link Analysis has been used for investigation of criminal activity (fraud detection, counterterrorism, and intelligence), computer security analysis, search engine optimization, market research, as well as medical research.

As regards the Web, a Link Analysis ranking algorithm starts with a collection of Web pages to be ranked. The algorithm then proceeds by extracting the hyperlinks between the pages and constructing the underlying hyperlink graph. The hyperlink graph is constructed by creating a node for every Web page, and a directed edge for every hyperlink between two pages. The graph is given as input to the Link Analysis ranking algorithm. The algorithm operates on the graph, and produces a weight for each Web page. This weight captures the authoritativeness of the page and is used to rank the pages. Depending on how the initial set of pages is obtained, Link Analysis algorithms are distinguished between *query independent* algorithms and *query dependent* algorithms. In the former case, the algorithm ranks the whole Web, while in the latter case, the algorithm ranks a subset of Web pages that is associated with the query at hand.

Link Analysis ranking can be traced back to two seminal papers by Brin and Page [161] and Kleinberg [133]. These two papers changed the way people think about the Web and spawned the research area of Link Analysis ranking. `PageRank` [161] was proposed as a *query independent* algorithm that produces a `PageRank` value for all Web pages and later became a commercial success story as an integral component of Google [42], the dominant search engine on the Web at this time. Kleinberg [133] describes how to rank a *query dependent* subset of Web pages. Using a text-based Web Search engine, a *root set* is retrieved consisting of a short list of Web pages relevant to a given query. Then, the root set is augmented by pages pointing to pages in the root set and also pages pointed by pages in the root set, to obtain a larger *base set* of Web pages. This is the query dependent subset of Web pages on which the algorithm operates.

Given the set of Web pages, the next step of a Link Analysis algorithm is to construct

the underlying hyperlink graph. A node is created for every Web page and a directed edge is placed between two nodes if there is a hyperlink between the corresponding Web pages. Even if there are multiple links between two pages, only a single edge is placed. No self-loops are allowed. The edges could be weighted using, for example, content analysis of the Web pages, similar to the spirit of the work of Bharat and Henzinger [33]. Usually, links within the same Web site are removed since they do not convey an endorsement; they serve the purpose of navigation. In addition, isolated nodes are removed from the graph.

In the next section, we describe several Link Analysis techniques.

3.5.1 Link Analysis Techniques

Citation-based measures

Estimating the "importance" of a scientific paper can be viewed as the predecessor of Link Analysis ranking. Research in bibliometrics has long been concerned with the use of citations to produce quantitative estimates of the importance and impact of individual scientific papers and journals. The most well-known measure in this field is Garfield's impact factor [100]. Under the standard definition, the impact factor of a journal j in a given year is the average number of citations received by papers published in the previous two years of journal j. Disregarding the question of whether two years is the appropriate period of measurement, we observe that the impact factor is a ranking measure based fundamentally on a pure counting of the in-degrees of nodes in the network.

Pinski and Narin [166] proposed a more subtle citation-based measure of standing, stemming from the observation that not all citations are equally important. They argued that a journal is *influential* if, recursively, it is heavily cited by other influential journals.

The PageRank algorithm

The intuition underlying the citation-based measures is that a good authority is a page that is pointed by many nodes in the graph. Brin and Page [161] extended this idea further by observing that not all links carry the same weight. Links from pages of high quality should confer more authority. It is not only important how many pages point to a page, but also what is the quality of these pages. Therefore, they have proposed an one-level weight propagation scheme, where a good authority is one that is pointed by many good authorities. They employ this idea in the `PageRank` algorithm.

The `PageRank` algorithm performs a *Random Walk* on the graph that simulates the behavior of a "random surfer". The surfer starts from some node chosen according to some distribution D (usually assumed to be the uniform distribution). At each step, the surfer proceeds as follows: with probability $1-p$ an outgoing link is picked uniformly at random and the surfer moves to a new page, and with probability p the surfer jumps to a random

page chosen according to distribution D. The jump probability p is passed as a parameter to the algorithm. The authority weight a_i of node i (called the `PageRank` of node i) is the fraction of time that the surfer spends at node i, that is, it is proportional to the number of visits to node i during the random walk.

Formally, the `PageRank` of node i is given by the following formula:

$$a_i = p \cdot D(i) + (1 - p) \sum_{j \in in(i)} \frac{a_j}{|out(j)|} \tag{3.1}$$

where $in(i)$ is the set of all pages that point to the page i, while $out(j)$ is the set of all pages that are pointed by the page j.

Note that the `PageRank` value of a page i is the sum of two components: one part of the value is equal for all pages if we assume uniform distribution (and expresses the probability of a random jump to i), and the other part comes from pages that point to i. The values can be computed iteratively and iterations should be run to convergence. According to [161], the number of iterations required for convergence is empirically $O(\log n)$, where n is the number of graph edges.

Following the `PageRank` algorithm, a large number of modifications and extensions were proposed. John A. Tomlin [196] proposes a generalization that computes flow values for the edges of the Web graph and a `TrafficRank` value for each page. In `Topic-Sensitive PageRank` proposed by Taher Haveliwala [112], documents are accessed non-uniformly according to their topics. In `Weighted PageRank` extensions [159, 212], links are followed non-uniformly according to their popularity. A link-semantics aware extension by Baeza-Yates and Davis [26] recognizes links with different meanings and compute a `PageRank` weighted by the link semantics. Finally, there exists a large body of work that deals with personalization of the `PageRank` algorithm [112, 120, 174].

Kleinberg's HITS algorithm

Kleinberg [58] proposed a more refined (query-dependant) notion for the importance of Web pages. He proposed a two-level weight propagation scheme where endorsement is conferred on authorities through hubs, rather than directly between authorities. In his framework, every page can be thought of as having two roles. The hub role captures the quality of the page as a pointer to useful resources, while the authority role captures the quality of the page as a resource itself. A good authority is a source of useful information, while a good hub is a page that contains a useful collection of links. If we make two copies of each page, we can visualize the graph as a bipartite graph, where hubs point to authorities. There is a mutual reinforcing relationship between the two. A good hub is a page that points to good authorities, while a good authority is a page pointed by good hubs.

In order to quantify the quality of a page as a hub and an authority, Kleinberg asso-

ciated every page with a hub and an authority weight. Following the mutual reinforcing relationship between hubs and authorities, Kleinberg defined the hub weight to be the sum of the authority weights of the nodes that are pointed by the hub, and the authority weight to be the sum of the hub weights that point to this authority. Let **h** denote the n-dimensional vector of the hub weights, where h_i (the i-th coordinate of vector **h**) is the hub weight of node i, and correspondingly let **a** denote the n-dimensional vector of the authority weights (where a_i is the authority weight of node i). We have that:

$$a_i = \sum_{j \in in(i)} h_j \qquad and \qquad h_i = \sum_{j \in out(i)} a_j \tag{3.2}$$

Building upon the mutual reinforcing relationship between hubs and authorities, Kleinberg proposed the following iterative algorithm for computing the hub and authority weights. Initially all authority and hub weights are set to 1. At each iteration the operations O ('out') and I ('in') are performed. The O operation updates the authority weights, and the I operation updates the hub weights, both using the Equations 3.2. A normalization step is then applied, so that the vectors **a** and **h** become unit vectors in some norm. The algorithm iterates until the vectors converge. This idea was later implemented as the `HITS` (Hyperlink Induced Topic Distillation) algorithm [102].

The `HITS` algorithm has two implicit properties. The first is symmetry. Both hub and authority weights are computed in the same way; authority weights are the sum of the hub weights, while hub weights are the sum of authority weights. The second is equality. When computing the hub weights (resp. authority weights), the sum operator treats all authority (resp. hub) weights equally.

There is a plethora of works that are based on the `HITS` algorithm. For instance, Bharat and Henzinger [33] and Chakrabarti et al. [51] consider improvements on the `HITS` algorithm by using textual information to weight the importance of nodes and links. Mendelzon and Rafiei [146] and Rafiei and Mendelzon [171] consider a variation of the `HITS` algorithm that uses random jumps. The same algorithm (so-called `Randomized HITS`) is also considered in the works by Ng et al. [157, 158]. Finally, extensions of the `HITS` algorithm that use multiple eigenvectors were proposed by Azar et al. [24].

The SALSA algorithm

An alternative query-dependant algorithm, called `SALSA`, was proposed by Lempel and Moran [141]. `SALSA` combines ideas from both `HITS` and `PageRank`. As in the case of `HITS`, the graph is visualized as a bipartite graph, where hubs point to authorities. The `SALSA` algorithm performs a random walk on the bipartite hubs and authorities graph, alternating between the hub and authority sides. The random walk starts from some authority node selected uniformly at random. The random walk then proceeds by alternating between

backward and forward steps. When being at a node on the authority side of the bipartite graph, the algorithm selects one of the incoming links uniformly at random and moves to a hub node on the hub side. When being at node on the hub side the algorithm selects one of the outgoing links uniformly at random and moves to an authority. The authority weights are defined to be the stationary distribution of this random walk.

The SALSA algorithm can be thought of as a variation of the HITS algorithm. In the I operation of the HITS algorithm the hubs broadcast their weights to the authorities, and the authorities sum up the weight of the hubs that point to them. The SALSA algorithm modifies the I operation as follows. Instead of broadcasting, each hub divides its weight equally among the authorities to which it points. Therefore:

$$a_i = \sum_{j \in in(i)} \frac{1}{|out(j)|} h_j \tag{3.3}$$

Similarly, it modifies the O operation so that each authority divides its weight equally among the hubs that point to it. Therefore:

$$h_i = \sum_{j \in out(i)} \frac{1}{|in(j)|} a_j \tag{3.4}$$

Spreading Activation

Spreading activation is a method for searching associative networks, neural networks, or semantic networks. The search process is initiated by labeling a set of source nodes (e.g., concepts in a semantic network) with weights or "activation" and then iteratively propagating or "spreading" that activation out to other nodes linked to the source nodes. Most often these "weights" are real values that decay as activation propagates through the network. When the weights are discrete this process is often referred to as marker passing. Activation may originate from alternate paths, identified by distinct markers, and terminate when two alternate paths reach the same node.

Each node i in a directed graph G has an associated activation value $a(i)$ which is a real number in the range $[0, 1]$. A link $l_{i,j}$ connects the source node i with the target node j. Each link has an associated weight $w(l_{i,j})$ which is usually a real number in the range $[0, 1]$. The parameters of the algorithm are (a) a firing threshold F (a real number in the range $[0, 1]$), and (b) a decay factor D (a real number in the range $[0, 1]$).

The graph is initialized by setting all activation values $a(i)$ to zero. We then set one or more origin nodes to an initial activation value greater than the firing threshold F (a typical initial value is 1.0). For each unfired node i in the graph having an activation value $a(i)$ greater than the node firing the threshold F, and for each link $l_{i,j}$ connecting the source node i with a target node j, we set:

$$a(j) = a(j) + (a(i) \cdot w(l_{i,j}) \cdot D) \tag{3.5}$$

If a target node receives an adjustment to its activation value so that it would exceed 1.0, then we set its new activation value to 1.0. Likewise, we maintain 0.0 as a lower bound on the target node's activation value (should it receive an adjustment below 0.0). Once a node has fired it cannot fire again, although variations of the basic algorithm permit repeated firings and loops through the graph. Nodes receiving a new activation value that exceeds the firing threshold F are marked for firing on the next spreading activation cycle. In addition, if activation originates from more than one node, a variation of the algorithm permits marker passing to distinguish the paths by which activation is spread over the graph.

The procedure terminates when either there are no more nodes to fire or, in the case of marker passing from multiple origins, when a node is reached from more than one path. Variations of the algorithm (that permit repeated node firings and activation loops in the graph) terminate after a steady activation state is reached (with respect to some delta) or when a maximum number of iterations is exceeded.

Spreading activation can be applied in IR by means of a network of nodes representing documents and terms contained in those documents [64, 65]. It is also very powerful to perform proximity searches, where given an initial set of concepts, the algorithm returns other concepts which are strongly connected to them. An overview of spreading activation techniques applied in IR is given by Fabio Crestani [65]. Spreading activation techniques have also been widely applied in Web [66] and Semantic [17, 63, 183] searching.

3.5.2 Applying Link Analysis on the Web of Data

There is a plethora of works that exploit Link Analysis-based methods for scoring semantic data.

For instance, the work by Bamba and Mukherjea [28] adopts a modification of HITS algorithm for estimating the importance of RDF resources. `Swoogle` [78] applies Link Analysis methods (based on a *rational surfer model*) for ranking the importance of RDF documents. Hogan et al. [116] propose a PageRank-like method for ranking RDF resources that takes into account the provenance of data. Zhang et al. [221] study the problem of *ontology summarization* and use three weighted variations of PageRank and HITS to define the eigenvector centrality of RDF statements. The work by Harth et al. [110] uses the original PageRank method to assign authority values to data sources and then propagates the authority values to identifiers referenced in the sources. Delbru et al. [75] introduce `DING`, a methodology for ranking information resources based on both dataset and entity ranks (which are computed by performing PageRank-like algorithms). Kleb and Abecker [132] present a method (based on spreading activation) for entity disambiguation

applied to RDF ontologies, which determines the most likely references for a given natural language identifier. Finally, Dali et al. [71] present a learning to rank approach for ranking RDF resources that equally match the query. Specifically, PageRank and HITS are used as centrality-based features (among other features) for ranking query-independent entities returned by a structured query.

3.5.3 Our approach

We propose a *biased Random Walk* model for ranking entities and properties, as well as for re-ranking a list of retrieved results. The proposed PageRank-like algorithm exploits both the ranking of the retrieved results, the importance (within the search context) of the extracted entities, and their connectivity. The model is thoroughly analyzed in Chapter 5.

3.5.4 Comparing Link Analysis Techniques

Distance Measures

We are interested in comparing different Link Analysis ranking algorithms, as well as studying the ranking behavior of a specific algorithm as we modify the underlying graph or some of its parameters. To all intents, a Link Analysis ranking algorithm A is a function that maps a graph G to an n-dimensional vector $A(G)$. Let A_1 and A_2 be two Link Analysis ranking algorithms, then we can define the distance between the algorithms A_1 and A_2 on graph G as $d(A_1(G), A_2(G))$, where $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ is some function that maps two real n-dimensional weight vectors $\mathbf{a_1}$, $\mathbf{a_2}$ to a real number $d(\mathbf{a_1}, \mathbf{a_2})$.

Geometric Distance Measures

We want to capture the closeness of the actual weights assigned to every node. The authority weight vectors can be viewed as points in an n-dimensional space, thus we can use common geometric measures of distance (e.g., the Manhattan distance). Let A_1 and A_2 be two Link Analysis ranking algorithms and let $\mathbf{a_1}$, $\mathbf{a_2}$ be the weight vectors of the algorithms on some graph G. We can define the distance measure d_1 between A_1 and A_2 on G as follows:

$$d_1(\mathbf{a_1}, \mathbf{a_2}) = min_{\gamma_1,\gamma_2 \geq 1} \sum_{i=1}^{n} |\gamma_1 a_1(i) - \gamma_2 a_2(i)| \tag{3.6}$$

The constants γ_1 and γ_2 are meant to allow for an arbitrary scaling of the two vectors, thus eliminating large distances that are caused solely due to normalization factors.

Rank Distance Measures

Now we are interested in measuring the similarity between the *ordinal rankings* induced

by two different algorithms. The motivation is that the ordinal ranking is the usual end-product seen by the user.

Let **a** be the k-dimensional authority weigh (score) of an algorithm A over some graph $G = (P, E)$. The vector **a** induces a ranking of the nodes in P, such that a node i is ranked above node j if $a_i > a_j$. If all weights are distinct, the authority weights induce a *total ranking* of the elements in P. If the weights are not all distinct, then we have a *partial* ranking of the elements in P (total rankings also refer as permutations).

Distance measures between permutations

The problem of comparing permutations has been studied extensively [77, 80]. In this setting, we have to compare two complete rankings of a set of elements.

Let P be a set of elements that we want to order (in our case the nodes in the graph). A permutation s is defined as a bijection from the set P to the set $[n] = 1, 2, \ldots, n$ where n is the size of P. The value $s(i)$ is interpreted as the position (rank) of the element $i \in P$ in the ranking. We say that element i is ranked ahead of element j if $s(i) < s(j)$. We also use N to denote the set of all distinct unordered pairs of nodes in P, and S_P to denote the set of all possible permutations of the elements of P.

The Kendall's tau distance measure [130] between permutations is defined as follows. Given two permutations $s1$ and $s2$, we define the indicator function $I_{s1s2}(i, j)$ such that $I_{s1s2}(i, j) = 0$, if i and j are ranked in the same order in both $s1$ and $s2$, and $I_{s1s2}(i, j) = 1$ otherwise. Kendall's tau is defined as: $K(s1, s2) = \sum_{\{i,j\}\in N} I_{s1,s2}(i, j)$. Kendall's tau is equal to the number of bubble sort swaps that are necessary to convert one permutation to the other. The maximum value of Kendall's tau is $n(n-1)/2$, and it occurs when the permutation $s1$ is the reverse of the permutation $s2$. Thus, we can normalize the Kendall's tau distance with $n(n-1)/2$, so that it takes values in $[0, 1]$.

Another metric for comparing permutations is Spearman's Footrule metric [77], which takes into account also the distance between the two permutations. Formally, for two permutations $s1, s2 \in S_P$, $F(s1, s2) = \sum_{i=1}^{n} |s1(i) - s2(i)|$.

Measures for comparing partial rankings

In an ideal world a ranking algorithm would produce a distinct authority weight for each node in the set P. Then we would be able to compare rankings by applying directly the distance measures on permutations. However, there are cases where the ranking algorithms may assign equal weights to two or more different nodes. There are several ways of generalizing the Kendall's tau distance for the case of partial rankings, e.g., minimizing Kendall rank distance, Hausdorff Kendall rank distance, and Kendall rank distance with penalty [41, 92].

3.5.5 Evaluating Link Analysis Techniques

In Information Retrieval

Evaluating the effectiveness of a Link Analysis technique in IR involves human subjectivity (for result relevance) and comprises two main methods: *user-based evaluations* where a number of users make relevance assessments regarding a set of queries and an ordered list of results for each query, and *test collections* (e.g., TREC [207]) where several systems apply their ranking algorithms to a collection of documents for which the relevant results are known for each query in a query set. Below, we detail a representative example of a user study.

Borodin et al. [41] introduced a theoretical framework for the study of Link Analysis ranking algorithms in the area of IR, which allows comparing different algorithms, and also proposed new families of algorithms which are based on the hubs and authorities framework (HITS). By performing an extensive (user-based) experimental evaluation (of the proposed algorithms as well as the algorithms HITS, PageRank and SALSA), they studied the quality of the algorithms and examined how the existence of different structures in the graphs affects their performance.

In total, 34 different queries were used in the user study. For each query, they started with a root set of pages related to the query. This root set was obtained by retrieving the first 200 pages returned by querying Google. This set was then expanded to the base set by including nodes that point to, or are pointed by, pages in the root set (for every page in the root set, they included only the first 50 pages that point to this page, in the same order they were returned by Google). They then extracted the links between the pages of the base set and constructed the hyperlink graph (the navigational links were removed).

In order to assess the relevance of the documents they performed an *on-line user study*. The introductory page contained the queries with links to the results, together with some instructions. By clicking on a query, the union of the top-10 results of all algorithms was presented to the user. The users were then asked to rate each document as “Highly Relevant”, “Relevant”, or “Non-Relevant”. An option “Don’t Know” (chosen as default) was also given, in the case that the user could not assess the relevance of the result. When clicking on the submit button, their feedback was stored into a file. In addition, no queries were assigned to any users, they were free to inspect whichever ones and however many they wanted.

They evaluated the quality of the different ranking algorithms using *precision over the top-10 results*. This is the fraction of results in the top-10 positions of the ranking that are relevant to the query. They were also interested in studying how the algorithms relate to each other. For the comparison of two rankings they used geometric and rank distance measures. They also measured the number of documents that two rankings have in common in the top-10 results, as well as the average intersection over the top-10 results (where

the average is taken over the intersection over the top-1, top-2, up to top-10).

In the Web of Data

Here we report how the works that apply Link Analysis-based methods in RDF data (described in 3.5.2) have been evaluated.

Bamba and Mukherjea [28] did not perform a formal evaluation and they only report *empirical results* from the implementation of the proposed ranking technique in an IR system for biomedical patents. Hogan et al. [116] present only *performance* evaluation results (ranking computation time) based on a large RDF dataset obtained from the Web.

Ding et al. [78] evaluated the proposed ranking algorithm (OntoRank) on a real dataset collected by `Swoogle`. They compared the *performance* between PageRank and OntoRank by ranking the results of ten popular local-names (according to Swoogle's statistics). Then, they compared the number of *strict Semantic Web ontologies*[6] in the first 20 results and showed that OntoRank outperforms PageRank. In addition, they inspected the best ranked Semantic Web documents using both PageRank and Ontorank, and they showed that OntoRank is intended to expose more ontologies.

Zhang et al. [221] evaluated their approach to ontology summarization using *ground truth* summaries produced by human experts. Specifically, three small ontologies were selected as the test case (since they can be reviewed by human to produce ground truths), and five judges, who are all experts on Semantic Web, were invited to take a peer review on each test case. Vocabulary overlap (a widely used content-based metric in text summarization) was used to measure the quality of the ontology summaries. The results showed that their approach to ontology summarization is feasible and promising.

Harth et al. [110] conducted *performance* and *quality* evaluations of the proposed method on a large Web dataset. They compared four variants of the proposed algorithm and a naive version of PageRank. Regarding the quality evaluation, the authors conducted a *user study* in which they asked participants to manually rate results of queries for each algorithm. For each query, they presented five different top-10 ranked lists, each corresponding to one of the ranking methods. The evaluators were asked to order these lists from 1 to 5, according to which lists they deemed to represent the best results for each query. Five scenarios were covered: query for all persons in the dataset, keyword search of evaluator's own name, keyword search of "Tim Berners-Lee", keyword search of "Dan Brickley", and keyword search of "John Breslin". For each scenario, the mean rank assigned by the evaluators for each method was plotted. The results showed that the proposed algorithm gives significantly better results than simply implementing PageRank on the object graph.

[6]According to [78], a Semantic Web ontology is called *strict* if its *ontology ratio* exceeds 0.8. The *ontology ratio* of a Semantic Web document is the fraction of its class-instances being recognized as classes and properties, and is used to identify ontologies among documents.

Delbru et al. [75] evaluated the quality of the proposed DING methodology by performing a *user study* where users provided relevance judgements for candidate algorithms. The user study was divided into two experiments (regarding different datasets). Each experiment included 10 queries, varying from simple keyword search to more complex structured queries (SPARQL). Each participant received a questionnaire containing a description of the query in human language, and three lists of top-10 results. Each result was described by the human-readable label and the URI of the entity. The task was to rank each of the proposed rankings in relation to the baseline ranking, using categorical variable. The participants had to choose between 5 categories: Better, Slightly Better, Similar, Slightly Worse, Worse. The Pearson's chi-square was used to perform the test of "goodness of fit". The results showed an improved ranking quality.

Kleb and Abecker [132] evaluated the quality of their proposed reference resolution algorithm by comparing the best (based on *human choice*) ontology reference with the reference retrieved by their algorithm. Specifically, in order to reflect the ambiguity of ontology elements, the authors used a highly ambiguous geography ontology. As input data, they used news articles crawled from the European Media Monitor which collects news of European newspapers and clusters them topic-wise. They selected the topic *natural disasters* in order to guarantee for documents including geographic entities. The included entities were manually annotated with their exact ontology surrogate. The results showed that the proposed reference resolution algorithm achieved 97.73% recall and 48.34% precision.

Finally, Dali et al. [71] evaluated their proposed ranking algorithm using real-world datasets (general and domain specific) and *human evaluation* based on *crowd sourcing*. In particular, given a question and the answers computed by the system, the evaluators had to vote for the answer which should be ranked first and also to indicate the confidence of their choice. The number of votes for an answer was used as the ranking criterion. The evaluation compared the proposed approach with two relevant baselines and the results showed that the proposed approach outperforms the baselines, and the improvement is large for the domain specific dataset.

Synopsis. We notice that the majority of existing works have been evaluated using user studies and human judgments ([71, 75, 110, 132, 221]), while one work provides only empirical results [28], one presents only time results [116], and one uses a very specialized (to their problem) evaluation approach [78]. The reason is the fact that there is not a standard evaluation procedure and collection that can be used by the majority of ranking approaches, since each approach focuses on a specific ranking problem that is affected by several parameters. These parameters are not the same across (most of) the approaches and thus no common evaluation procedure can be applied (e.g., some works focus on ranking whole RDF documents, others on ranking RDF resources or RDF statements, etc.). Nevertheless, there are initiatives (like INEX [76] and TREC [27]) that focus on the related

problem of retrieving and ranking *entities* from a structured knowledge repository based on free text queries.

Chapter 4
Configuring Named Entity Extraction

In this chapter, we focus on how to tackle configuration issues that arise within the context of the proposed search process. This corresponds to the configuration steps described in 2.5 (Figure 2.9). Recall that the number of identifiable entities as well as the amount of structured information that is available for these entities can be very high. Therefore, there is a need of a method that allows *specifying* this "data of interest". Instead of proposing a model of specific purpose that is applicable only to one particular domain, we have elaborated on the general problem of configuring a Named Entity Extraction (NEE) system because this is important for supporting professional exploratory search in various domains.

Below, we first provide a few fundamental notions and notations (4.1). We then introduce the proposed configuration model (4.2) and we describe the "Open NEE Configuration Model", an RDF/S vocabulary that allows describing and exchanging the configuration supported by a NEE system (4.3). In 4.4 we present an extension of the Open Annotation Data Model which allows relating the output of a NEE process with an applied configuration. In 4.5 we present `X-Link`, a configurable NEE framework we have designed and implemented which realizes the proposed configuration model. In 4.6 we evaluate `X-Link` in terms of usability and efficiency, and finally (in 4.7) we conclude the chapter.

4.1 Notions and Notations

We first formalize the structured knowledge available as LOD or queryable through a SPARQL endpoint. Consider an infinite set U of URI references, an infinite set B of blank nodes and an infinite set L of literals [32]. A triple $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an *RDF triple* (s is called the *subject*, p the *predicate* and o the *object*). A Semantic Knowledge Base (SKB) K, or equivalently an *RDF graph* G, is a set of RDF triples. For an RDF Graph G_i, we shall use U_i, B_i, L_i to denote the URIs, blank nodes and literals, respectively, that appear in the triples of G_i. The *nodes* of G_i are the values that appear as subjects or objects in the

triples of G_i.

Let now C be a set of *entity categories*, e.g., C = –Fish Species, Country, Water Area" are possible categories for the marine domain. For a category $c \in C$, let $E(c)$ denote the set of *entity names* in c, e.g., E(Country) = –Afghanistan, Albania, Algeria, ...". Inversely, let $ctg(e) \in C$ denote the category of an entity name (e.g., ctg(Algeria) = Country). For an entity name e, let $U(e)$ denote the URIs that are related to e and exist in one or more RDF graphs, e.g., U(Chum Salmon) = –`http://dbpedia.org/resource/Chum_salmon`, `https://www.googleapis.com/freebase/v1/rdf/m/03ysh6`". For an entity URI u, let $Descr(u)$ be a set of RDF triples that express information about u in an RDF graph.

Now, for an input document, say doc, we define as $Ent(doc, c)$ the set of entities identified in doc (by applying NER) that belong to the category c. Obviously $Ent(doc, c) \subseteq E(c)$. In general, in a given document we can identify entities of various categories, each of these entities is associated with URIs and each of these URIs with triples that describe these URIs. Specifically, for a document doc:

- $Ent(doc) = \cup_{c \in C} Ent(doc, c)$ is the set of entities identified in doc
- $U(doc) = \cup_{e \in Ent(doc)} U(e)$ is the set of URIs of these entities
- $Graph(doc) = \cup_{u \in U(doc)} Descr(u)$ is a set of triples about these URIs that actually define an RDF graph

In many cases we have a name that corresponds to entities of different categories. For example, "argentina" may refer to the country Argentina or the fish genus Argentina. In general, a name may correspond to n categories. In such cases we consider that we have n different entities, one for each category. Therefore, each of these entities will have one category (i.e., $|ctg(e)| = 1$). This choice enables to apply afterwards NED methods.

4.2 The Proposed Configuration Model

Figure 4.1 depicts the configuration model that we propose in the form of a UML class diagram. Each `Category` has a name and can be associated with one or more Knowledge Base Mirrors (KBMs). A KBM holds the URL of a SPARQL endpoint and it is associated with three kinds of elements:

(a) SPARQL Queries
(b) SPARQL Template Queries for Entity Linking
(c) SPARQL Template Queries for Entity Enrichment

The elements of type (a) are used for specifying the *entity names* of interest by providing one or more KBM-answerable SPARQL queries.

The elements of type (b) allow specifying how entity names correspond to *entity URIs*, by providing one or more KBM-answerable SPARQL queries.

The elements of type (c) allow specifying what *extra information* (in the form of RDF triples) should be fetched for each entity URI, by providing one or more KBM-answerable

SPARQL queries. This template query is also associated with a *name* that can be used for labeling this extra information.

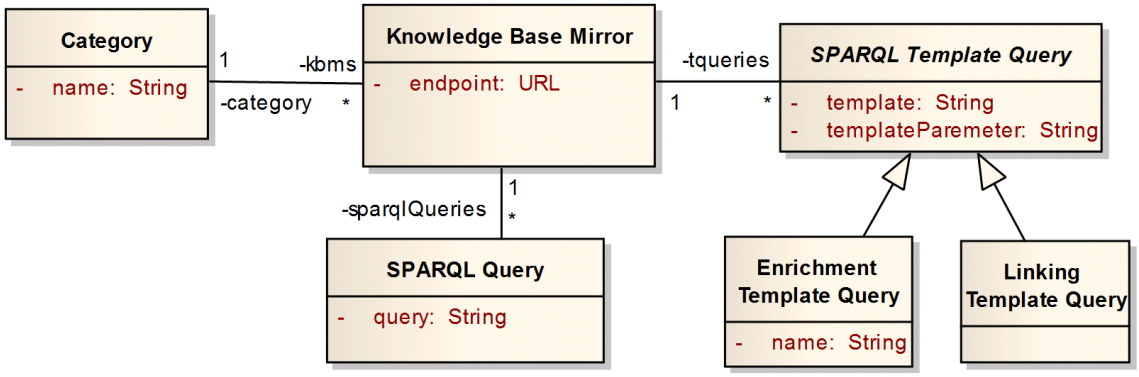


Figure 4.1: A generic model for configuring a NEE system.

A configuration essentially defines an information structure as defined in 4.1. Specifically, it defines the set of categories C. For each category $c \in C$, the corresponding set of entity names $E(c)$ is obtained by running the corresponding SPARQL queries to the related KBMs. For each entity name $e \in E(c)$, its linked URIs $U(e)$ are obtained by running the corresponding Linking Template Queries (where e is passed as parameter), and for each URI $u \in U(e)$ the triples $Descr(u)$ are obtained by running the corresponding Enrichment Template Queries (where u is passed as parameter). Approaches for ranking the URIs related to a named entity, i.e., the set $U(e)$, are described and evaluated in [29, 82].

Example of the Configuration Model

Let us now describe an indicative instantiation of the above model. Consider a set of two categories C = –Fish Species, Country". The category Fish Species is associated with two KBMs:

- KBM_1 = `http://dbpedia.org/sparql` (SPARQL endpoint of DBpedia)
- KBM_2 = `http://www.fao.org/figis/flod/endpoint` (SPARQL endpoint of FAO Fisheries Linked Open Data (FLOD) [4])

The category Country is associated with one KBM:

- KBM_3 = `http://factforge.net/sparql` (SPARQL endpoint of FactForge [35])

For KBM_1, we can set the SPARQL query of Figure 4.2 for specifying the fish species of interest, or the one shown in Figure 4.3 in case we are interested only in English fish names.

For *Entity Linking*, KBM_1 can be associated with the template query shown in Figure 4.4 which aims at retrieving URIs of type Fish whose label *contains* the name of an entity (ignoring case). Notice that the query contains the character sequence `[ENTITY]` (including the [and]) which is replaced (at query-time) by the entity name. For example, by pro-

```
SELECT DISTINCT str(?label) WHERE {
  ?uri rdf:type <http://dbpedia.org/ontology/Fish> ; rdfs:label ?label }
```

Figure 4.2: SPARQL query for retrieving a list of fish names from DBpedia.

```
SELECT DISTINCT str(?label) WHERE {
  ?uri rdf:type <http://dbpedia.org/ontology/Fish> .
  ?uri rdfs:label ?label FILTER(lang(?label)="en") }
```

Figure 4.3: SPARQL query for retrieving a list of fish names in English from DBpedia.

viding the entity name "chum salmon", DBpedia returns the URI `http://dbpedia.org/resource/Chum_salmon`. Of course, one could provide a "stricter" SPARQL template query, focusing on bigger precision. For example, the query in Figure 4.5 retrieves URIs whose label *equals* the entity name (ignoring case). The Linking Template Queries can be also considered a way of "trivial" disambiguation that has the objective to find the resources that better characterize the entity name. However, a characteristic of this trivial disambiguation is that we already know the category of the corresponding entity and thereby we can form accordingly the template query (e.g., we can compare the entity name with the names of entities belonging to a specific RDF class, as in the template queries of Figures 4.4 and 4.5).

```
SELECT DISTINCT ?uri WHERE {
  ?uri rdf:type <http://dbpedia.org/ontology/Fish> .
  ?uri rdfs:label ?label FILTER(regex(str(?label), "[ENTITY]", "i")) }
```

Figure 4.4: SPARQL template query for linking an identified Fish name with resources in DBpedia.

```
SELECT DISTINCT ?uri WHERE {
  ?uri rdf:type <http://dbpedia.org/ontology/Fish> .
  ?uri rdfs:label ?label FILTER (lcase(str(?label)) = lcase("[ENTITY]")) }
```

Figure 4.5: "Stricter" SPARQL template query for linking an identified Fish name with resources in DBpedia.

For *Entity Enrichment*, KBM_1 can be associated with the template query shown in Figure 4.6 which retrieves the *outgoing* properties of an entity URI. Notice that the query contains the character sequence `[URI]` (including the [and]) which is replaced (at query-time) by the entity URI. For example, by providing the URI `http://dbpedia.org/resource/Chum_salmon`, one of the properties that DBpedia returns is the following:

```
http://dbpedia.org/ontology/genus -> http://dbpedia.org/resource/Oncorhynchus
```

By collecting the RDF triples that correspond to a set of entity URIs, we can form an RDF graph from which we can infer whether and how these entity URIs are connected.

```
SELECT DISTINCT ?propertyName ?propertyValue WHERE {
  <[URI]> ?propertyName ?propertyValue }
```

Figure 4.6: SPARQL template query for retrieving the outgoing properties of resource.

Analogously, one can specify SPARQL queries and template queries for all KBMs related to the defined categories. Any of the above queries can also use the *federated* features of SPARQL 1.1 [11], meaning that information from more than one endpoints can be used.

4.3 The Open NEE Configuration Model

Here we introduce an RDF/S vocabulary for describing a configuration based on the proposed model. This vocabulary, apart from defining explicitly the semantics of a configuration, allows a NEE system to describe (and publish as Linked Data) its "entity mining" capabilities. It also allows to better handle the provenance of the outcome of a NEE process by explicitly describing the configuration used during the NEE process.

Figure 4.7 depicts the proposed model, which we call "Open NEE Configuration Model". The vocabulary is accessible through: `http://www.ics.forth.gr/isl/oncm`. We have defined 8 classes and 13 properties (they are briefly described in Table 4.1). The model allows describing the supported categories and the related KBMs. It also allows specifying the methods (if any) that are used for ranking the entities and the matched resources.

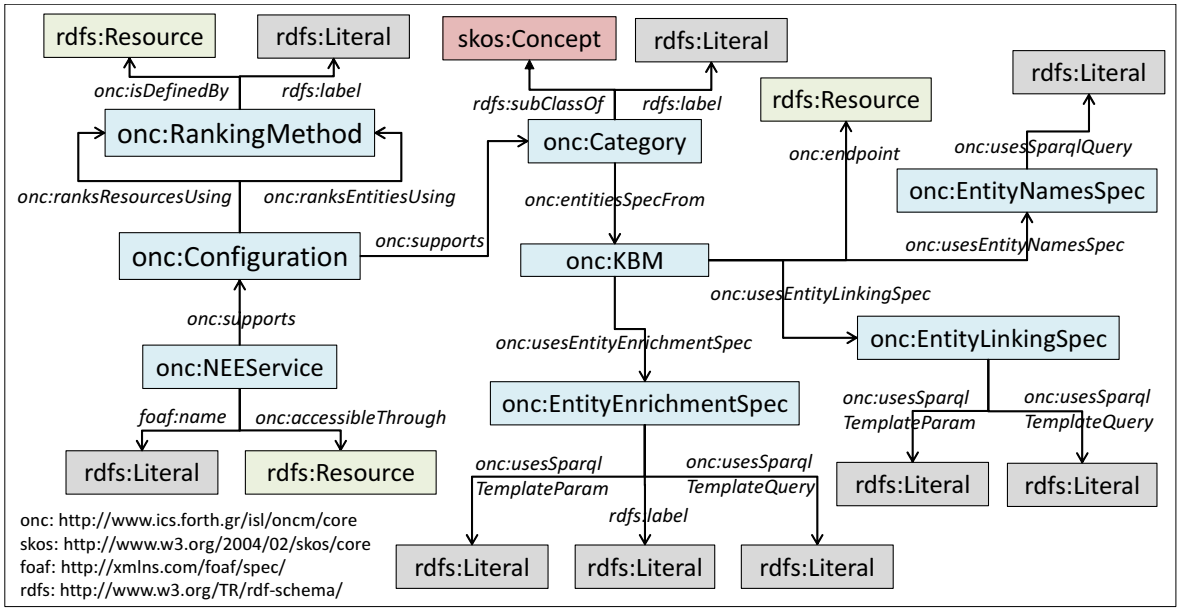


Figure 4.7: The Open NEE Configuration Model.

Table 4.1: Classes and properties of the Open NEE Configuration Model.

Class	Class description
`NEEService`	A Named Entity Extraction (NEE) service.
`Configuration`	The configuration supported by a NEE service.
`Category`	A category/class of entities supported by a configuration.
`RankingMethod`	A method used for ranking the entities or the entity URIs.
`KBM`	A Knowledge Base Mirror (KBM): the gateway for accessing a Semantic Knowledge Base (SKB).
`EntityNamesSpec`	Specification of the entity names of a category.
`EntityLinkingSpec`	Specification of how an entity name corresponds to entity URIs.
`EntityEnrichmentSpec`	Specification of the extra information that should be fetched for an entity URI.
Property	**Property description**
`supports`	Relates a NEE service to a configuration, or a configuration to a supported category.
`accessibleThrough`	Relates a NEE service to a resource, e.g., to a URL describing the API of a service.
`ranksEntitiesUsing`	Relates a configuration to a method for ranking entities.
`ranksResourcesUsing`	Relates a configuration to a method for ranking resources.
`isDefinedBy`	Relates a ranking method to a resource, e.g., to a URL describing the ranking approach.
`entitiesSpecFrom`	Relates a category to a KBM.
`endpoint`	Relates a KBM to the URL of a SPARQL endpoint.
`usesEntityNamesSpec`	Relates a KBM to specification of entity names.
`usesEntityLinkingSpec`	Relates a KBM to an entity-linking specification.
`usesEntityEnrichmentSpec`	Relates a KBM to an entity-enrichment specification.
`usesSparqlQuery`	Relates a specification of entity names to a SPARQL query.
`usesSparqlTemplateQuery`	Relates an entity-linking or entity-enrichment specification to a SPARQL template query.
`usesSparqlTemplateParam`	Relates an entity-linking or entity-enrichment specification to a SPARQL template parameter.

Although the model allows relating a configuration with a NEE service, we can describe a configuration without providing information about the service (i.e., without connecting an instance of `onc:Configuration` with an instance of `onc:NEEService`) because, for example, we want to create a general configuration which will be used for configuring one or more NEE services. Moreover, the model exploits the SKOS [10] vocabulary for indicating that the class `onc:Category` is subclass of the class `skos:Concept`. Thereby, we can interrelate the supported categories exploiting the SKOS properties. For example, we can define that the category Species is a broader concept of the categories Fish Species and Bird Species. This information can be then exploited by the underlying application for offering, for example, a more advanced visualization (e.g., by using nesting if a category is narrower that another). We can also provide domain information by relating a category with concepts/classes from a widely used taxonomy or thesaurus. For instance, if there

is a well-known thesaurus related to the marine domain, we can define that the category Fish Species is related to a concept of that thesaurus. Moreover, we can exploit the provenance data model (PROV [14]) and include provenance information, e.g., who created a configuration, when, etc.

Figure 4.8 depicts an instantiation example of this model, while Figure 4.9 shows the corresponding RDF triples. In this example, the NEE system "X-Link" supports a configuration which can identify entities of type Fish Species. We can also see the KBM that is used for linking, enriching and updating the entities of this category, as well as the method used for ranking the matched resources.

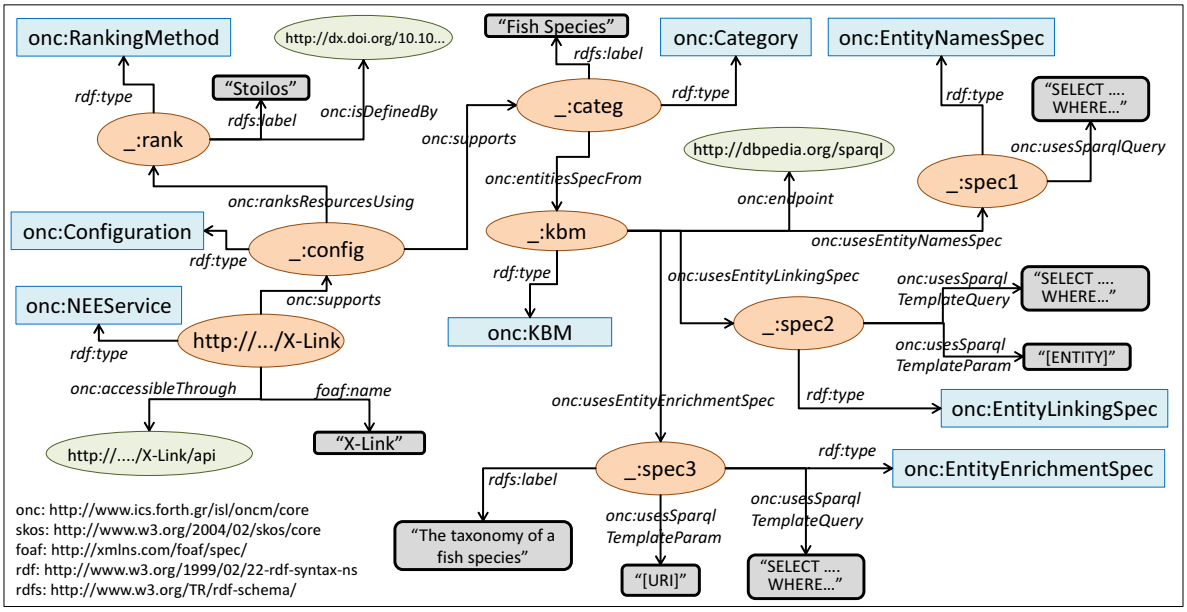


Figure 4.8: Instantiation Example of the Open NEE Configuration Model.

By publishing the configurations supported by one or more NEE services, and considering that there is a registry/catalogue with such configurations, an application can dynamically detect and use the services that satisfy its annotation needs, while we are able to run (SPARQL) queries of the form:

- Give me the NEE services supporting a category with name "Fish" (Figure 4.10)
- Give me all categories supported by the NEE service "X-Link" (Figure 4.11)
- Give me the SPARQL template queries (together with the endpoints) that are used for *linking* entities of the category "Fish" (Figure 4.12)
- Give me NEE services supporting categories of entities related to the marine domain (Figure 4.13)

```
PREFIX onc:  <http://www.ics.forth.gr/isl/oncm/core>
PREFIX skos: <http://www.w3.org/2004/02/skos/core>
PREFIX foaf: <http://xmlns.com/foaf/spec/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX rdfs: <http://www.w3.org/TR/rdf-schema/>

<http://.../X-Link>  rdf:type                      onc:NEEService ;
                     foaf:name                     "X-Link" ;
                     onc:accessibleThrough         <http://.../X-Link/api> ;
                     onc:supports                  _:config .

_:config             rdf:type                      onc:Configuration ;
                     onc:supports                  _:categ ;
                     onc:ranksResourcesUsing       _:rank .

_:rank               rdf:type                      onc:RankingMethod ;
                     rdfs:label                    "Similarity using Stoilos function." ;
                     onc:isDefinedBy               <http://dx.doi.org/10.1007/11574620_45> .

_:categ              rdf:type                      onc:Category ;
                     rdfs:label                    "Fish Species" ;
                     onc:entitiesSpecFrom          _:kbm .

_:kbm                rdf:type                      onc:KBM ;
                     onc:endpoint                  <http://dbpedia.org/sparql> ;
                     onc:usesEntityNamesSpec       _:spec1 ;
                     onc:usesEntityLinkingSpec     _:spec2 ;
                     onc:usesEntityEnrichmentSpec  _:spec3 .

_:spec1              rdf:type                      onc:EntityNamesSpec ;
                     onc:usesSparqlQuery           "SELECT ... FROM ..." .

_:spec2              rdf:type                      onc:EntityLinkingSpec ;
                     onc:usesSparqlTemplateQuery   "SELECT ... FROM ..." ;
                     onc:usesSparqlTemplateParam   "[ENTITY]" .

_:spec3              rdf:type                      onc:EntityEnrichmentSpec ;
                     rdfs:label                    "The taxonomy of a fish species." ;
                     onc:usesSparqlTemplateQuery   "SELECT ... FROM ..." ;
                     onc:usesSparqlTemplateParam   "[URI]" .
```

Figure 4.9: Example of RDF triples describing a configuration.

4.4 Exporting/Exchanging the Annotation Results

It is often useful to know the *provenance* of a NEE process. Provenance also concerns the configuration under which an annotation process was applied. Towards this direction, but also to allow exporting the results in a standard format so that other applications can exploit them, we propose an extension of the Open Annotation Data Model [189]. The Open Annotation Data Model specifies an RDF-based framework for creating associations (an-

```
SELECT ?tool ?name WHERE {
  ?tool a onc:NEEService ; foaf:name ?name ; onc:supports ?config .
  ?config onc:supports ?categ . ?categ rdfs:label "Fish" }
```

Figure 4.10: SPARQL query for retrieving the name and the URL of all services that support the category "Fish".

```
SELECT ?name WHERE {
  <http://.../X-Link> onc:supports ?config .
  ?config onc:supports ?categ . ?categ rdfs:label ?name }
```

Figure 4.11: SPARQL query for retrieving the categories supported by the X-Link NEE system.

```
SELECT ?endpoint ?template WHERE {
  ?categ a onc:Category ; rdfs:label "Fish" ; onc:entitiesSpecFrom ?kbm .
  ?kbm onc:endpoint ?endpoint ; onc:usesEntityLinkingSpec ?linkspec .
  ?linkspec onc:usesSparqlTemplateQuery ?template }
```

Figure 4.12: SPARQL query for retrieving the endpoints and the template queries that are used for linking entities of the category "Fish".

```
SELECT ?tool ?name WHERE {
  ?tool a onc:NEEService ; foaf:name ?name ; onc:supports ?config .
  ?config onc:supports ?categ . ?categ skos:related example:marine }
```

Figure 4.13: SPARQL query for retrieving the name and the URL of all services that support categories related to the marine domain.

notations) between related resources, allowing annotations to be easily shared between platforms.

The extension model (which is an RDF/S vocabulary) is depicted in Figure 4.14 and comprises 1 new class, 8 new properties and 1 new instance (they are briefly described in Table 4.2). The new class `oae:Entity` is subclass of `oa:SemanticTag` and is used for representing a detected entity. The instance `oae:NEE` is a `oa:Motivation` representing the result of a NEE process and can be considered narrower than the `oa:tagging` motivation. The property `oae:usingConfiguration` is used for associating the annotation process with a configuration. The property `oae:detectedAs` is used for representing the string in the document that was detected and considered an entity, while the property `oae:regardsEntityName` represents the actual entity name that exists in a gazetteer (text file containing a list of sorted entity names) of the NEE system. The property `oae:position` is used for representing the positions in the document in which the entity name was detected, `oae:score` represents the score of an entity or of an entity URI, while `oae:conf-`

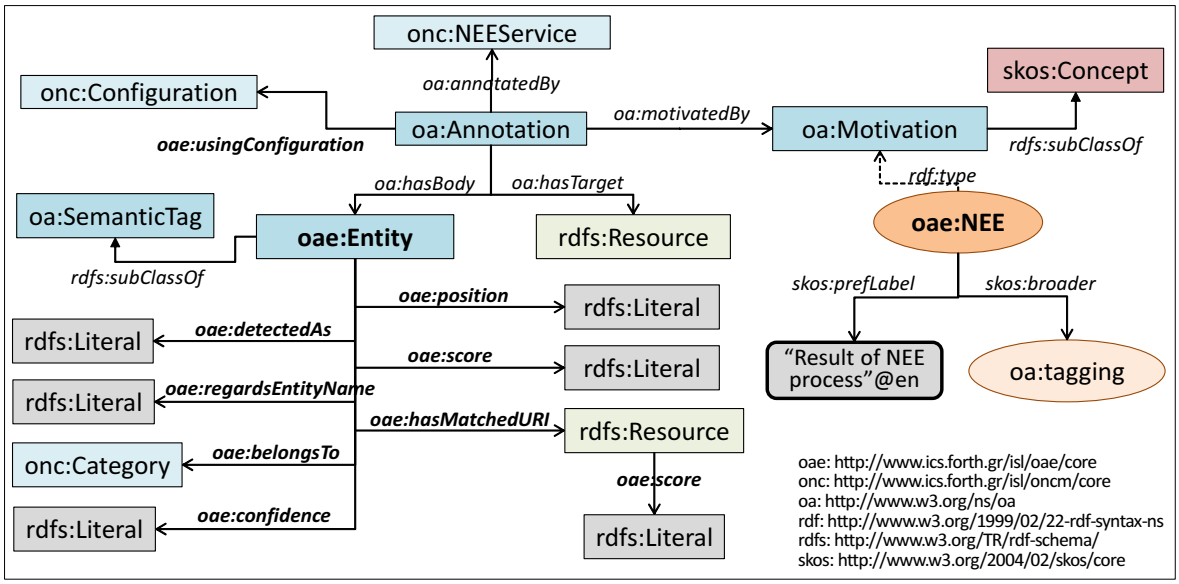


Figure 4.14: The extension of the Open Annotation Data Model.

Table 4.2: The class, the instance and the properties of the Open Annotation Extension.

Class	**Class description**
`Entity`	Represents an identified entity.
Instance	**Instance description**
`NEE`	Represents the result of the Named Entity Extraction process.
Property	**Property description**
`usingConfiguration`	Relates the annotation process to a configuration.
`detectedAs`	Relates an entity to a literal representing the string in the document that was detected and considered an entity.
`regardsEntityName`	Relates an entity to a literal representing the actual entity name that exists in a gazetteer of the NEE system.
`position`	Relates an entity to one or more literals representing the positions in the document in which the entity name was detected.
`score`	Relates an entity to a literal (or a URI to a literal) representing the score of an entity (or of URI).
`confidence`	Relates an entity to a literal representing the confidence of an ambiguous entity.
`belongsTo`	Relates an entity to a category.
`hasMatchedURI`	Relates an entity to a URI.

`idence` can represent the confidence of an ambiguous entity. The property `oae:belongsTo` is used for representing the category of the detected entity. Finally, the property `oae:hasMatchedURI` is used for representing the URIs that match an entity name.

The extension is available at `http://www.ics.forth.gr/isl/oae`. Figure 4.15 depicts an instantiation example in which the fish name "catfish" was detected in a Wikipedia

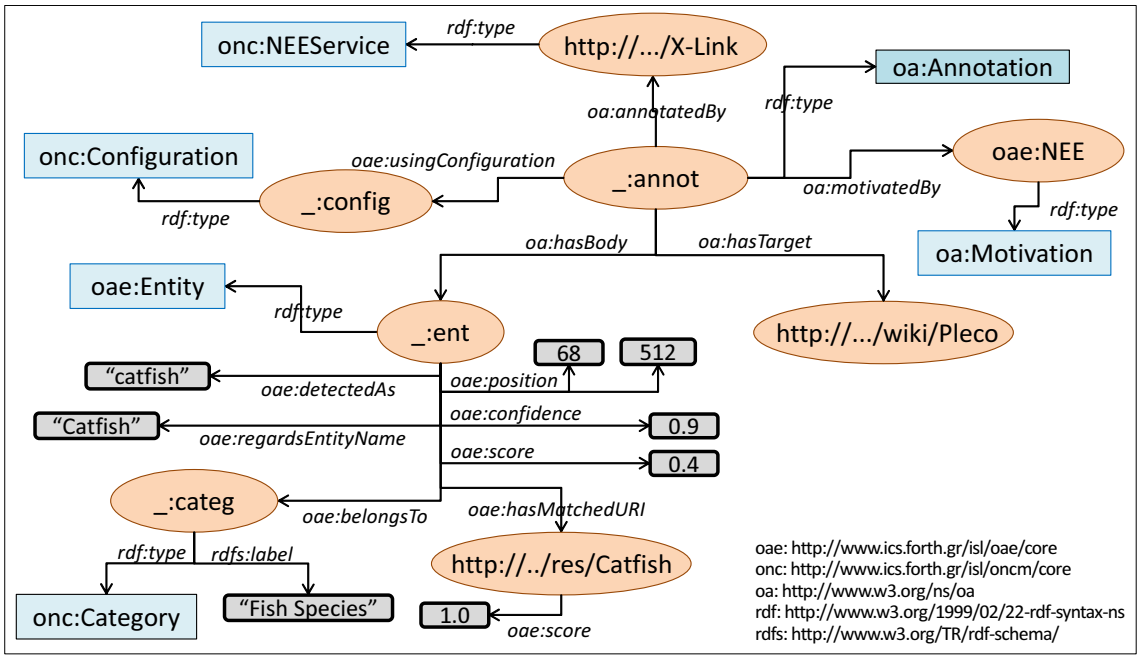


Figure 4.15: Instantiation Example of the Open Annotation Extension.

page by the X-Link NEE system. Figure 4.16 depicts the RDF triples of a similar example. In this example, the NEE system detected two entities of type fish species. Each entity is accompanied by a matched URI which means that the entity linking process was applied. We notice also that the entity enrichment process was applied and two properties were retrieved from DBpedia (the properties `dbp-owl:order` and `dbp-owl:phylum`).

By performing NEE in a set of documents and exporting the results using the proposed extension, we can run (SPARQL) queries of the form:

- Give me documents referring the fish "Catfish" (Figure 4.17)
- Give me documents referring entities of type "Fish" (Figure 4.18)
- Give me documents referring fishes of phylum "Chordate" (Figure 4.19)
- Give me fishes of order "Ostariophysi" detected in the Web page "`http://example.html`" (Figure 4.20)

An application can now offer advanced exploratory search services over the annotated set of documents, e.g., according to the faceted interaction paradigm over RDF data [96, 103, 124].

```
PREFIX oae:     <http://www.ics.forth.gr/isl/oae/core>
PREFIX oa:      <http://www.w3.org/ns/oa>
PREFIX onc:     <http://www.ics.forth.gr/isl/oncm/core>
PREFIX rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX rdfs:    <http://www.w3.org/TR/rdf-schema/>
PREFIX dbp-owl: <http://dbpedia.org/ontology/>
PREFIX dbp:     <http://dbpedia.org/resource/>

_:annot                     rdf:type                    oa:Annotation ;
                            oa:annotatedBy              <http://.../X-Link> ;
                            oa:motivatedBy              oae:NEE ;
                            oae:usingConfiguration      _:config ;
                            oa:hasBody                  _:ent1 ;
                            oa:hasBody                  _:ent2 ;
                            oa:hasTarget                <http://.../wiki/Pleco> .

_:ent1                      rdf:type                    oae:Entity ;
                            oae:detectedAs              "loricariidae" ;
                            oae:regardsEntityName       "Loricariidae" ;
                            oae:confidence              0.95 ;
                            oae:position                42, 115, 312 ;
                            oae:score                   0.6 ;
                            oae:belongsTo               _:categ ;
                            oae:hasMatchedURI           <http://.../Loricariidae> .

<http://../Loricariidae>    dbp-owl:order               dbp:Catfish ;
                            dbp-owl:phylum              dbp:Chordate ;
                            oae:score                   1.0 .

_:ent2                      rdf:type                    oae:Entity ;
                            oae:detectedAs              "catfish" ;
                            oae:regardsEntityName       "Catfish" ;
                            oae:confidence              0.9 ;
                            oae:position                68, 512 ;
                            oae:score                   0.4 ;
                            oae:belongsTo               _:categ ;
                            oae:hasMatchedURI           <http://../Catfish> .

<http://../Catfish>         dbp-owl:order               dbp:Ostariophysi ;
                            dbp-owl:phylum              dbp:Chordate ;
                            oae:score                   1.0 .

_:categ                     rdf:type                    onc:Category ;
                            rdfs:label                  "Fish Species" .
```

Figure 4.16: Example of RDF triples describing the result of a NEE process using the Open Annotation Extension.

```
SELECT ?doc WHERE {
  ?annot a oa:Annotation ; oa:hasTarget ?doc ; oa:hasBody ?ent .
  ?ent a oae:Entity ; oae:regardsEntityName "Catfish" ; oae:belongsTo ?cat .
  ?cat a onc:Category ; rdfs:label "Fish" }
```

Figure 4.17: SPARQL query for retrieving documents referring the fish “Catfish”.

```
SELECT ?doc WHERE {
  ?annot a oa:Annotation ; oa:hasTarget ?doc ; oa:hasBody ?ent .
  ?ent a oae:Entity ; oae:belongsTo ?cat .
  ?cat a onc:category ; rdfs:label "Fish" }
```

Figure 4.18: SPARQL query for retrieving documents referring entities of type “Fish”.

```
SELECT ?doc WHERE {
  ?annot a oa:Annotation ; oa:hasTarget ?doc ; oa:hasBody ?ent .
  ?ent a oae:Entity ; oae:belongsTo ?cat ; oae:hasMatchedURI ?uri .
  ?cat a onc:category ; rdfs:label "Fish" .
  ?uri dbp-owl:phylum dbp:Chordate }
```

Figure 4.19: SPARQL query for retrieving documents referring fishes of phylum “Chordate”.

```
SELECT ?entName WHERE {
  ?annot a oa:Annotation ; oa:hasTarget <http://example.html> ; oa:hasBody ?ent .
  ?ent a oae:Entity ; oae:belongsTo ?cat .
  ?cat a onc:category ; rdfs:label "Fish" .
  ?ent oae:regardsEntityName ?entName ; oae:hasMatchedURI ?uri .
  ?uri dbp-owl:order dbp:Ostariophysi }
```

Figure 4.20: SPARQL query for retrieving fishes of order “Ostariophysi” detected in the Web page “http://example.html”.

4.5 The X-Link Framework

X-Link is a LOD-based NEE framework we have designed and implemented which supports the proposed configuration model. Below, we describe its its functionality (4.5.1) and the supported configurability (4.5.2).

4.5.1 Functionality

X-Link is based on the Gate ANNIE system [40, 69] and supports both gazetteers and NLP functions. Gate ANNIE is a ready-made information extraction system which contains several components (e.g., Tokeniser, Gazetteer, Sentence Splitter, Orthographic Coreference, etc.). X-Link extends Gate ANNIE to allow creating a new supported category and updating an existing one (using gazetteers). This gives us the opportunity to adapt its functionality according to our needs, making X-Link configurable and extendible. The architecture of X-Link, as well as implementation details, can be found in the master's thesis of Manolis Baritakis [29].

Input / Output

Currently X-Link supports the analysis of plain text files, HTML pages, Microsoft Word and Powerpoint files (.doc, .docx, .ppt and .pptx), PDF files, and XML-based files, while it exports the results in XML, CSV and RDF. As regards RDF, X-Link exploits the extension of the Open Annotation Data Model described in 4.4, and supports the formats RDF/XML, N-Triples, and Notation3 (N3).

Entity Mining

X-Link at first reads the contents of the requested document. Then it applies entity mining using Gate ANNIE according to the specified categories of interest. In our setting, Gate ANNIE takes as input the contents of a document and the categories of interest, and the output is a set of detected entities. Each detected entity is accompanied by its category and its position(s) in the document.

Named Entity Disambiguation (NED)

In 4.1 we argued that we may have a name that corresponds to entities of different categories (recall the "argentina" example which may refer to the Country Argentina or the Fish Genus Argentina). Even if only one category is supported, we cannot be sure if a detected entity (that matches an entity name in the gazetteer of the supported category) actually belongs to this category. This is the well-known Named Entity Disambiguation (NED), or Word-Sense Disambiguation, problem whose solution stills an open challenge

[48]. Several approaches have been proposed in the literature, e.g., exploiting Wikipedia data [67, 109], using statistical methods [1], exploiting ontologies [111], or graph-based approaches [154, 204].

In our "exploratory search" context, `X-Link` does not apply any disambiguation method, i.e., if an entity name exists in the gazetteers of two supported categories, then this entity is returned twice, one for each supported category. Notice that in some application scenarios, especially in professional systems, even if we are not sure about the relevance of a piece of information, it is preferable to retrieve and return it. For instance, in Professional Search (e.g., medical search, patent search, bibliography search) it is often unacceptable to miss relevant documents.

NED methods that are appropriate in our case (i.e., NER is based on gazetteers of LOD entities) are being investigated in the master's thesis of Manolis Baritakis [29].

Entity Linking

As regards the *entity linking* process, `X-Link` returns a list of URIs that match a detected entity name and lets the underlying application to decide how to cope with them. For example, an application could present to the user all the URIs that match an identified entity, while another system could rank them and return only the top-ranked URI.

Entity Enrichment

As regards *entity enrichment*, i.e., the retrieval of RDF triples that describe the entity URIs, `X-Link` offers two different functions: a) retrieve triples that are interesting to the application at hand, and b) inspect the connectivity of the entity URIs.

As regards the former, for an entity URI u, $Descr(u)$ is obtained either by running the corresponding template queries or by selecting to retrieve one of the following (common) types of properties: a) outgoing (the entity URI is the `subject` in the RDF triple), b) incoming (the entity URI is the `object` in the RDF triple), c) both outgoing and incoming, d) outgoing in a specific language, e) both outgoing in a specific language and incoming.

As regards the connectivity of the entity URIs, `X-Link` computes a graph for making more evident how the entity URIs are associated. Specifically, this graph contains only the triples which are involved in paths whose both start and end vertex are URIs of detected entities. For example, the graph can show entities that share common properties or which are directly connected (so properties that are not reachable by at least two entities are omitted).

Ways to Use

`X-Link` is a framework that can be used (and extended) by other applications according to

their needs, allowing its exploitation in a plethora of contexts and application scenarios. Specifically, X-Link can be used as a:

- **Java Library** which can be integrated in the code of the intended application.
- **Web Service** which can be used through a REST API.

In the last case it is assumed that a running instance exists, therefore X-Link offers operations that allow *changing* its configuration. This allows changing or refreshing the "knowledge" of X-Link without having to redeploy the underlying application. More information is available at: http://www.ics.forth.gr/isl/X-Link.

4.5.2 Configurability

X-Link supports the configuration model described in 4.2 in two ways: (a) it can read such a configuration from a *properties file*, and (b) it offers a configuration API. It can also read a configuration expressed in RDF using the proposed Open NEE Configuration Model. For publishing the configuration supported by an X-Link service, X-Link offers a function that creates an RDF file describing its current configuration using the Open NEE Configuration Model. For instance, the configuration that is currently supported by an X-Link service configured for the *marine* domain is publicly available at http://www.ics.forth.gr/isl/X-Link/marine/config.n3.

File-based Configuration

An indicative part of the properties file (configured for the *marine* domain) is shown in Figure 4.21. In that example, X-Link supports 7 categories of entities (line 1), i.e., the entity names of these categories have been retrieved and stored in Gate ANNIE. However, the *active categories* are only Fish, Country and Water Area (line 2), i.e., the remaining categories are inactive. The set of *active categories* allows us to define which of the supported categories are currently interesting for an application, thus X-Link can identify entities that belong to these categories only. The category Fish uses one KBM (line 3), which is actually the SPARQL endpoint of DBpedia (line 4), and for updating this category X-Link can use the SPARQL query given in a file (line 5). In addition, we can see the file paths and the parameters of the template queries that are used for linking and enriching the identified fishes (lines 6-9).

Configuration while Running

X-Link can be configured through its API even while a corresponding service is running. The main functions that are supported are the following:

- Add a new category (using lists of entities and/or SPARQL queries)
- Update an existing category (using lists of entities and/or SPARQL queries)

```
1   xlink.categories.supported = Fish;Country;Water_Area;Disease;Drug;Protein;Chemical_Substance
2   xlink.categories.active = Fish;Country;Water_Area
3   xlink.categories.Fish.kbms = dbpedia_fish
4   xlink.categories.Fish.kbms.dbpedia_fish.endpoint = http://dbpedia.org/sparql
5   xlink.categories.Fish.kbms.dbpedia_fish.entitynames = C:/xlink/queries/dbp_fishes.sparql;
6   xlink.categories.Fish.kbms.dbpedia_fish.templatequeries.linking =
                                          C:/xlink/templates/dbp_fish_linking.tquery
7   xlink.categories.Fish.kbms.dbp_fish.templatequeries.linking.parameter = [ENTITY]
8   xlink.categories.Fish.kbms.dbp_fish.templatequeries.enriching =
                                          C:/xlink/templates/dbpedia_fish_enriching.tquery
9   xlink.categories.Fish.kbms.dbpedia_fish.templatequeries.enriching.parameter = [URI]
```

Figure 4.21: A part of X-Link's properties file configured for the marine domain.

- Remove a category
- Change the displayed name of a category (rename)
- Set/change the KBMs of a category
- Set/change the SPARQL queries and the template queries of a KBM
- Set/change the *active* categories

Regarding the update of an existing category, the user/developer is able to either totally *replace* a category (i.e., remove its old entity names and add the new ones) or just add the new entity names. Notice also that each of the above functions changes accordingly the properties file and updates several files in Gate ANNIE. For example, when a new category is created, the corresponding gazetteer file is created and loaded in Gate ANNIE, the name of the category is added in the set of supported categories in the properties file, etc.

Portability of Configurations

The configurations can be easily exchanged. For instance, consider that a person A configures the system and then sends the configuration files to a person B. The person B sets the system to use the configurations files received by the person A (by simply providing some paths). Now the person B is able to enjoy exactly the same configuration as person A.

The size of the configuration files is relatively small and mainly depends on the number of supported categories and on the number of named entities in each category. Indicatively, the configuration files for supporting 4 categories related to the marine domain have size less than 5MB. These files include the gazetteers of the supported categories and several files required by Gate ANNIE. Note also that X-Link does not store any semantic information (e.g., URIs or RDF triples), since the entity linking and the entity enrichment processes are performed at real-time.

We should also stress that adopting the Open NEE Configuration Model simplifies even more the exchange of configurations since an RDF file can be just provided.

4.6 Evaluation

We first examine (through a *user study*) the usability of `X-Link`, specifically the convenience of configuring it (4.6.1). Then (in 4.6.2), we report the results of a *case study* over an online (publicly available) SKB regarding the efficiency of the proposed LOD-based approach. Other aspects (regarding the NEE effectiveness and the formulation of SPARQL queries) are discussed in 4.6.3.

4.6.1 Task-based User Study

The purpose of the user study is a) to test the usability of the proposed approach, i.e., how *fast* and *conveniently* a user can configure `X-Link`, and b) to identify usability problems that will allow us to improve the system. Note that the target user is an *administrator* or a *developer* who wants to use `X-Link` for building and dynamically configuring an application.

Tasks and Scenario

We deployed `X-Link` as a Web application configured for the *marine* domain which can identify Fish Species in a text or Web document. The administrator of the system can change the configuration through an administration page. Specifically, she/he can add, remove and update categories (i.e., specify the EoI), and specify how to link and enrich the identified entities. These tasks actually correspond to the configuration steps 2 and 4 described in 2.5 (Figure 2.9). Figure 4.22 depicts an indicative screenshot of the evaluation prototype. By clicking on the configuration button (at the upper right corner), the user can see the configuration options depicted in Figure 4.23. Screenshots of the configurations forms can be found in Appendix A.

11 subjects participated in the user study, 23 to 34 years old, members of the Information Systems Laboratory at FORTH-ICS, with computer science background and a basic knowledge of Linked Data and the SPARQL query language. Note that 11 participants are enough for revealing severe usability problems. Specifically, according to Robert Virzi [206], in a usability evaluation 80% of the usability problems are detected with four or five subjects, additional subjects are less and less likely to reveal new information, while the most severe usability problems are likely to have been detected by the first few subjects. Furthermore, according to Laura Faulkner [93], at least 10 subjects are needed to reduce the risk of not revealing usability problems.

We shortly (in about 5 minutes) described and demonstrated the application and its functionality to the participants, and then we asked them to perform the following tasks:

T1 *Add* a new category of entities

T2 *Update* a category

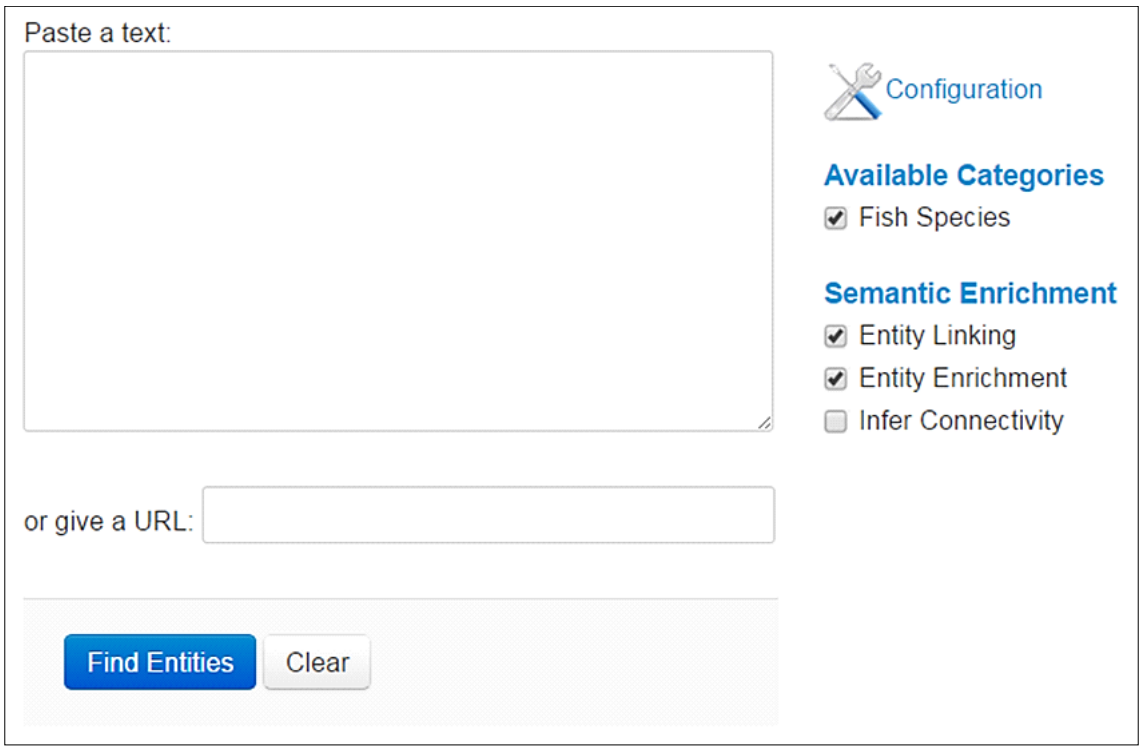


Figure 4.22: Indicative screenshot of the `X-Link` evaluation prototype.

T3 Specify how to *link* the identified entities of a category
T4 Specify how to *enrich* the entity URIs of a category
T5 *Perform* entity mining and inspect the connectivity of the entity URIs

The tasks are based on the following scenario:

*"Consider that you are the administrator of an application that can identify Fish names (currently supporting only the English language) in Web pages. You have been asked to perform some changes. Specifically, by exploiting DBpedia, the application must also identify European Countries **(T1)** as well as fish names in Spanish **(T2)** (because the application will be used mainly by Spaniards). Also, the identified fishes must be linked with resources from DBpedia **(T3)** and must be enriched with all their outgoing properties **(T4)**. Finally, in order to test that the system has been properly configured, perform entity mining in the Spanish version of Salmon's Wikipedia page and then inspect the connectivity of the identified entities **(T5)**"* ⋄

We also provided the participants with the following data:

- DBpedia's SPARQL endpoint (required for T1, T2, T3 and T4): `http://dbpedia.org/sparql`
- URI of the resource class `European Country` (required for T1):

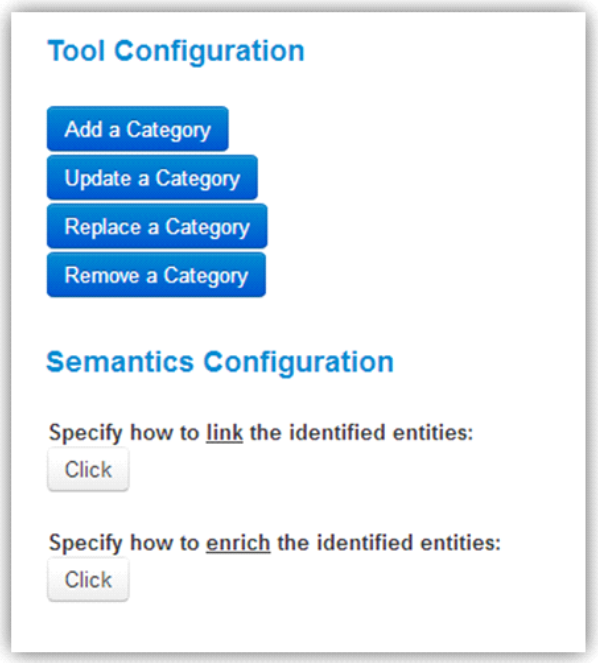


Figure 4.23: The configuration options of the `X-Link` evaluation prototype.

`http://dbpedia.org/class/yago/EuropeanCountries`

- Language code of Spanish (required for T2): `es`
- URI of the resource class `Fish Species` (required for T2 and T3): `http://dbpedia.org/ontology/Fish`
- URI of the property "label" (required for T2 and T3): `http://www.w3.org/2000/01/rdf-schema#label`
- Spanish version of Tuna's Wikipedia page (required for T5): `http://es.wikipedia.org/wiki/Thunnus`

We provided this data because we do not want to evaluate the ability of a participant to locate information on the Web and in DBpedia, since such a task depends on many complex factors (user experience, knowledge of DBpedia, etc.).

For the tasks T1 to T4, the participants could also load an example of a SPARQL query and modify it (instead of writing it from scratch). We recorded whether they succeeded to complete each task of the above scenario, as well as the time to successfully accomplish each task. In addition, at the end we asked them to complete a questionnaire. Specifically, they had to answer the following questions:

Q0 How easy was to configure the system according to the scenario?
Q1 How easy was to add the new category of entities?
Q2 How easy was to update the existing category?
Q3 How easy was to specify how to link the identified entities?

Q4 How easy was to specify how to enrich the identified entities?
Q5 How easy was to perform entity mining and inspect the connectivity of the identified entities?
Q6 What was difficult for you during the execution of the scenario?
Q7 How familiar are you with SPARQL?

Regarding the questions Q0 to Q5, the user could select one of the following answers: `very easy`, `easy`, `normal`, `difficult`, `very difficult`, `impossible`. As regards the question Q6 the user could write free text, while for the question Q7 the user could select a value between 1 (*I don't know SPARQL*) and 5 (*I am expert in SPARQL*).

Results

Figure 4.24 (left) depicts the success rate of each task. All participants managed to complete the tasks T1, T2 and T5. However, 18% of the participants (in particular two persons) failed to complete T3 and 9% (one person) failed to complete T4. The difficulty behind T3 and T4 is the comprehension of the *SPARQL template query*, specifically, the purpose of the template parameter inside the query and how it is used for constructing the template query (this was also made evident by the responses in Q6).

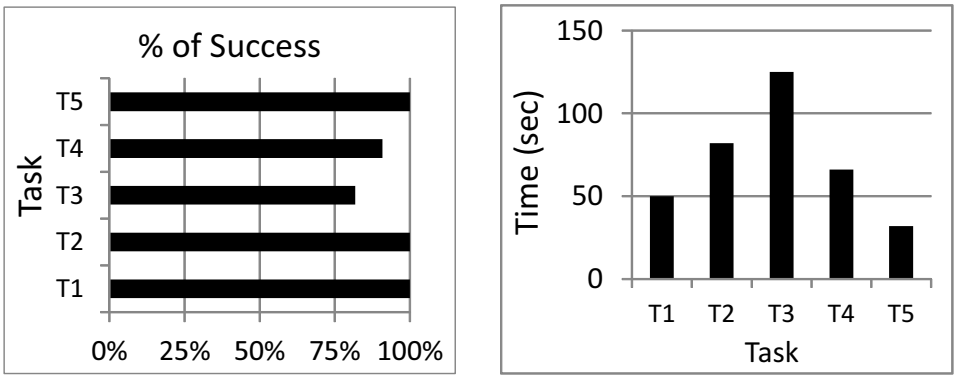


Figure 4.24: Success rate (left) and average time (right) for completing each task (results from 11 users).

Figure 4.24 (right) illustrates the average time for completing (successfully) each task. The reported times include the actual processing time, i.e., the time for running the corresponding SPARQL queries in tasks T1, T2 and T5, and the time for performing entity mining in T5. We notice that the most time consuming task was T3 which required about two minutes in average. This is a predictable result because T3 asked participants to construct (for first time) a SPARQL template query. In addition, the participants managed to totally configure the system according to the scenario (T1 to T4) in less than 6 minutes (in average).

Table 4.3: Evaluation of the difficulty in performing the scenario (results from 11 users).

Question	Very easy	Easy	Normal	Difficult	Very Difficult	Impossible
Q0	18%	82%	0%	0%	0%	0%
Q1	100%	0%	0%	0%	0%	0%
Q2	55%	27%	18%	0%	0%	0%
Q3	27%	45%	27%	0%	0%	0%
Q4	18%	55%	27%	0%	0%	0%
Q5	45%	45%	9%	0%	0%	0%

As regards the questionnaire, Table 4.3 depicts the results of the first 6 questions which refer to the difficulty in performing the tasks. None of the participants considered one of the tasks "difficult", "very difficult" or "impossible". 82% of the participants found the overall configuration (Q0) an "easy" task, while 18% found it "very easy". Regarding T3, which according to the success rates of Figure 4.24 was the most difficult task, 37% answered "normal", 45% answered "easy", while 27% answered "very easy". As regards T4, which was the second most difficult task according to the success rates, 27% answered "normal", 55% answered "easy", and 18% answered "very easy". Furthermore, all participants considered "very easy" the creation of a new category (T1). We notice that although some of the participants failed to complete T3 and T4, none of them found these tasks difficult. This is probably justified by the fact that after the evaluation, if a participant had completed unsuccessfully one or more of the tasks, we explained them their errors. Maybe, they then understood their errors and considered the task of "normal" difficulty.

Regarding Q6, a few participants mentioned a difficulty in understanding the notion of the SPARQL template queries (one also suggested to provide a user-friendly interface for constructing them). This can be justified by the fact that we did not explain it with many examples during the initial (5-minute) demonstration of `X-Link`. In addition, a participant commented that she/he would like to get informed with more details about the result of each action. For example, when updating a category it would be nice if the system reported the number of the added entity names. Finally, regarding Q7, 18% selected the answer "2", 36% selected "3", 36% selected "4", and 9% selected "5", meaning that about half of the participants were not very experienced with SPARQL, however most of them managed to configure the system.

Synopsis

Concluding the above results, we can say that by adopting the LOD-based approach that we propose and understanding the notion of SPARQL template queries, one can easily configure a NEE system within a few minutes. We should also stress that if we had dedicated more time for explaining the notion of the template queries (e.g., with more examples), perhaps all the participants would have also successfully completed T3 and T4.

4.6.2 Case Study: Querying Online DBpedia

We performed a case study for testing the feasibility of the proposed LOD-based approach. We used online DBpedia as the underlying SKB and we measured the average time for:

- creating a new category
- linking an identified entity with semantic resources
- enriching an entity URI

For improving the accuracy of the results, and since we were querying an online SKB at *real-time*, we repeated the experiments 20 times (specifically, about 2 times per day for 10 days) and here we report the average values (including the network's delay time). This case study can be also considered an evaluation of a publicly available SKB, since we ran many queries at DBpedia's SPARQL endpoint. Notice however that the results highly depend on the efficiency of the queried endpoint, which means that the results may vary a lot if we run the same experiments in another endpoint. The experiments were carried out using an ordinary computer with processor Intel Core i7 @ 3.4Ghz CPU, 8GB RAM and running Windows 7 (64 bit). The implementation is in Java 1.7.[1]

Creating a New Category

We used 7 sets of DBpedia resource classes. Each set has 5 different resource classes containing a particular number of entities (thus, totally 35 different resource classes were used). Each resource class actually corresponds to the new category that we want to create in `X-Link`. We measure a) the time for running the SPARQL query at DBpedia's SPARQL endpoint (which retrieves the labels of the entities belonging to the corresponding resource class, like the query in Figure 4.2), and b) the time for reading the answer and creating the category in `X-Link`.

Figure 4.25 depicts the average times for each set of resource classes. As expected, the time consuming task is the execution of the SPARQL query, since we query DBpedia's SPARQL endpoint at real time (the other task costs less than 10 seconds in all cases). We see that for resource classes with small number of entities (up to 10,000) the time is less than 20 seconds, while for resources classes with about 100,000 entities the time is about 5 minutes. A limitation regarding DBpedia's SPARQL endpoint is that it does not return more than 50,000 results at once, thus we had to run multiple queries for resource classes with more than 50,000 entities (using SPARQL's `LIMIT` and `OFFSET`). For this reason, adding a category with one million entities takes about 50 minutes. However note that this task is performed once (in a preprocessing step) or every time we want to update the entities of the corresponding category.

[1]The data used in the experiments (queries, resource classes, entity names, URIs, etc.) are accessible at `http://www.ics.forth.gr/isl/X-Link/files/exper_data.zip`.

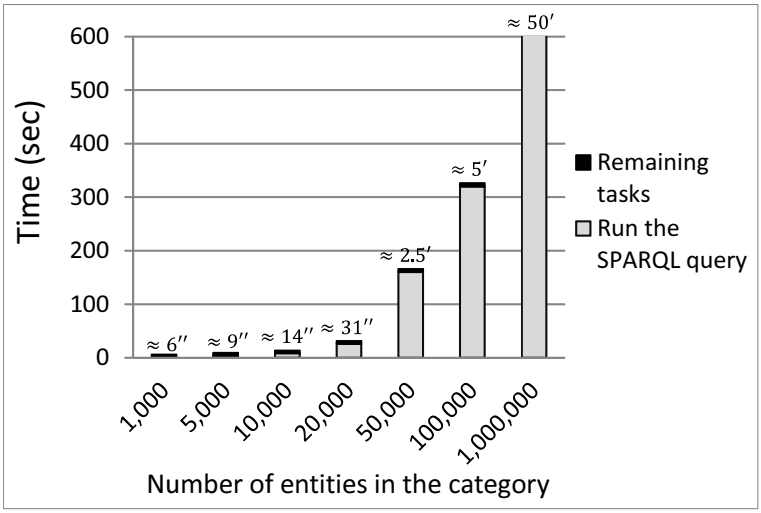


Figure 4.25: Time for adding a new category.

```
SELECT DISTINCT ?URI WHERE {
   ?URI rdf:type <[URI_OF_RES_CLASS]> .
   ?URI rdfs:label ?Name FILTER(regex(str(?Name),"[ENTITY]","i")) }
```

Figure 4.26: The SPARQL template query used in the experiments for linking the entities with semantic resources.

Time for Linking an Identified Entity

The time highly depends on the total number of entities belonging to the corresponding category. We used 8 sets of DBpedia resource classes, each one containing classes of a particular number of entities. Each set has 5 different resource classes (thus, totally 40 different resource classes were used). Note that each resource class actually corresponds to the category of an identified entity. For every resource class, we randomly selected 10 labels of entities belonging to that class and measured the average time for running the SPARQL query shown in Figure 4.26 (`[URI_OF_ RES_CLASS]` corresponds to the URI of the resource class, while `[ENTITY]` corresponds to the randomly selected label).

Figure 4.27 depicts the average times. We notice that for entities belonging to categories with up to 100,000 entities, the average time is less than 1 second, while for entities in categories with up to 1 million entities, the linking time is about 5 seconds. In addition, for linking an entity belonging to a category with 6 million entities the time is about 25 seconds. We should stress here that, in some application scenarios, this functionality can be offered *on-demand*. For example, in the search system `X-Search` the user can request to inspect the semantic resources that match an entity by clicking a small icon next to the entity's name (more in 6.3).

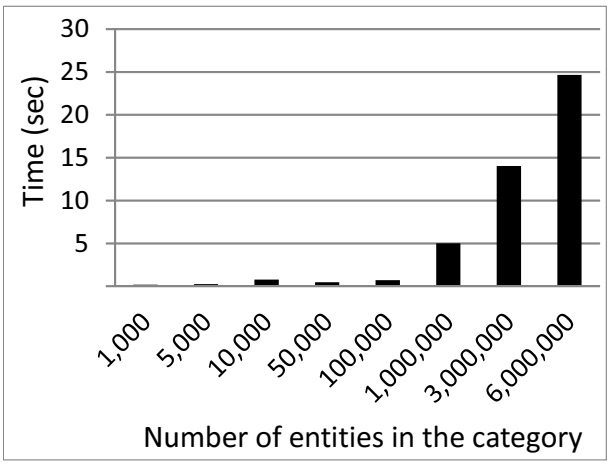


Figure 4.27: Time for linking an identified entity.

Time for Enriching an Entity URI

The time highly depends on the properties that we retrieve. We ran experiments for the following types of properties: i) incoming, ii) outgoing, iii) outgoing of a specific language, and iv) union of incoming and outgoing. We expect that the time will be very low since the entity URI is known (no many string comparisons are required) and the number of properties related to an entity is usually relatively small.

We randomly selected 160 URIs from DBpedia and measured the average time required for retrieving the properties. Figures 4.28-4.31 show the corresponding queries that we ran for each type of properties, and Figure 4.32 depicts the results. As expected, the time is very low (less than 300 ms) for all types of properties. Like in the case of entity linking, in some application scenarios this functionality can be offered on-demand. For instance, in the search system `X-Search` the user can explore the properties of an entity resource by clicking on its URI (more in 6.3).

Synopsis

The experimental results showed that even if we query an online SKB at real-time we can support the exploitation of LOD. This is very important since the "real-time" approach exploits the dynamic and "open" nature of LOD. In addition, we have seen that if an entity belongs to a category which contains millions of entities, then the time for linking a detected entity at real-time can be high. The same is true in case the underlying application requires to retrieve semantic information for numerous entities at once, i.e., when this functionality is not offered on-demand. In such cases, indexing the entire or part of the underlying SKB can highly improve efficiency, however with the cost of loosing the freshness of the results.

```
SELECT  ?propertyName ?propertyValue WHERE {
    ?propertyName ?propertyValue <[URI]> }
```

Figure 4.28: SPARQL template query for retrieving the incoming properties of a URI.

```
SELECT  ?propertyName ?propertyValue WHERE {
  <[URI]> ?propertyName ?propertyValue. }
```

Figure 4.29: SPARQL template query for retrieving the outgoing properties of a URI.

```
SELECT DISTINCT ?propertyName ?propertyValue WHERE {
   { <[URI]>  ?propertyName ?propertyValue FILTER(!isLiteral(?propertyValue)) }
   UNION
   { <[URI]> ?propertyName ?propertyValue FILTER(lang(?propertyValue)="en") } }
```

Figure 4.30: SPARQL template query for retrieving the outgoing properties of a URI, filtered by language.

```
SELECT DISTINCT ?propertyName ?propertyValue WHERE {
   { <[URI]> ?propertyName ?propertyValue }
   UNION { ?propertyName ?propertyValue <[URI]> } }
```

Figure 4.31: SPARQL template query for retrieving both the incoming and the outgoing properties of a URI.

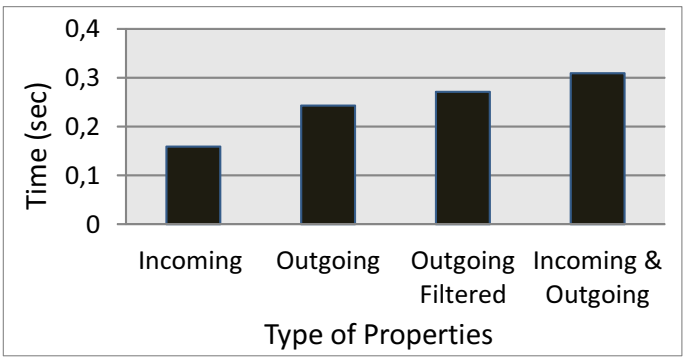


Figure 4.32: Time for enriching an identified entity.

4.6.3 Other Aspects

Effectiveness of NEE

There are various works that aim at evaluating the effectiveness of NEE tools [99, 177–179]. The effectiveness of `X-Link` highly depends on how the user/developer has configured it,

i.e., on the completeness of the specified categories, the quality of the underlying SKBs, the specified SPARQL template queries, etc. In this thesis we have focused on the *configurability* of a NEE system and on how we can exploit the LOD even at real-time; we have not proposed a new NLP/ML algorithm for NER or any disambiguation method. `X-Link` is a framework that supports the proposed configuration model and is currently relies on Gate ANNIE.

Notice also that the proposed configuration model can be applied by existing systems. For instance, a system that only performs NER and word-sense disambiguation can be LOD-aware by offering entity linking and entity enrichment capabilities. Likewise, a NER system that performs entity linking can describe its functionality by providing the categories it supports and the SKBs it exploits. Of course, in this case specifying linking template queries is not needed since the system can directly return the corresponding URI derived by its (internal) entity-linking process.

Formulation of SPARQL Queries

There are many tools that can facilitate the construction of SPARQL queries [22, 184]. Furthermore, there are natural language approaches that guide users in formulating queries in a language seemingly akin to English and translate them to SPARQL [81]. In this thesis, we consider that the administrator of the system knows the SPARQL query language.

4.7 Epilogue

We have proposed a generic (abstract) model that exploits the LOD for configuring a NEE system. This model allows specifying the EoI as well as the related (and useful for the application context) semantic information. We also introduced the Open NEE Configuration Model, an RDF/S vocabulary that allows exchanging configurations, and an extension of the Open Annotation Data Model that, amongst others, allows relating the output of a NEE process with an applied configuration. Finally, we have presented X-Link, a configurable LOD-based NEE framework that implements the proposed configuration model.

The key results of the evaluation can be summarized as follows:

- The results of the conducted user study showed that, by adopting the proposed LOD-based approach, one can configure a NEE system within a few minutes. In addition, the majority (80%) of the participants managed to successfully configure the system according to the specified scenario and also found it an easy task.

- The results of the conducted case study over online DBpedia demonstrated that even if we query a publicly available SKB we can support the exploitation of LOD at real-time, while to improve efficiency one can index the entire or part of the underlying SKB however with the cost of loosing the freshness of the results.

Chapter 5
Stochastic Ranking of Entities, Properties and Search Results

In this chapter, we initially focus on the problem of selecting and ranking the identified entities and their related structured information, i.e., on the Step 4 of the process described in 2.4 (Figure 2.6). Recall that the number of identified entities can be high and the amount of structured information that has been retrieved for these entities can be very high too, i.e., their associations and properties. Therefore, there is a need for methods that *rank* all this semantic information in order to promote and present to the end-users the most important (for the search context) entities, associations and properties. Furthermore, since common IR techniques often do not work well for more complex search tasks that go beyond fact-finding, we elaborate on whether and how such semantic information related to the retrieved results can be exploiting for re-ranking the list of results, aiming to promote relevant but low-ranked hits associated with important semantic information.

Below, we first define the required notions and notations (5.1) and we introduce the notion of "entity importance" (5.2). Then (in 5.3), we detail the stochastic process used for ranking the extracted entities and the related semantic information (and which is also exploited for constructing and showing top-K semantic graphs). In 5.4, we describe the Random Walk model used for re-ranking the retrieved results. In 5.5 we present evaluation results, and finally (in 5.6) we conclude the chapter.

5.1 Notions and Notations

Assuming that we are in the context of a submitted query q, we define the following notions and notations:

Documents (Search Results) (Step 1 of the process described in 2.4)

- L: the number of top documents (hits) to retrieve from the underlying search system for the query q.

- A: the set of the top-L documents of the answer of q.
- $score(a)$: the score (value in the range $[0, 1]$) of a document $a \in A$ as returned by the underlying search system for the query q.
- $rank(a)$: the position of a document $a \in A$ in the answer for the query q (i.e., the first hit has rank equal to 1, the second 2, etc.).

Document parts

- P: the set of different "parts" that constitute a document, e.g., $P = \{title, abstract, body\}$.
- a_p: the part of document $a \in A$ of type $p \in P$ (e.g., its *abstract*).
- $w(p)$: the weight expressing the importance of a part $p \in P$, where $\sum_{p' \in P} w(p') = 1$. For example: $w(title) = 0.5$, $w(abstract) = 0.3$, $w(body) = 0.2$.

Mined entities (Step 2 of the process described in 2.4)

- $ent(a_p) \subseteq EoI$: the set of entities identified in the part p of a document $a \in A$ by applying NER.
- $ent(a) = \cup_{p \in P} ent(a_p)$: the set of all entities identified in a document a (in all its parts).
- $E = \cup_{a \in A} ent(a)$: the set of all entities identified in A.
- $docs(e) = \{ a \in A \mid e \in ent(a)\}$: the elements of A in which e has been identified (inverse of $ent(a)$).
- $ef(e, a_p)$: the frequency (number of occurrences) of the entity e in the part p of the document a.

Entity URI and Related Semantic Data (Step 3 of the process described in 2.4)

- $U(e)$: the URIs related to the entity e (derived by running the *linking* template queries, as described in 4.1 and 4.2).
- $Descr(u)$: set of RDF triples that express information about the URI u (derived by running the *enrichment* template queries, as described in 4.1 and 4.2).
- R: the set of entities and properties related to the identified entities but not detected in the search results.

5.2 Entity Importance

We now define the notion of *"entity importance"*. As regards a single document, we consider that the more frequent entities are the more important. The term frequency (in our case entity frequency) is a classic numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [172]. We also take into account the different parts of a document and promote the entities identified in the important parts (i.e., in those with the highest weight). For example, regarding a scientific article, we may

consider that the entities detected in the title of the article are more important than those detected in the main body.

Precisely, the importance of an entity e within a document a is defined as:

$$imp(e, a) = \sum_{p \in P} \left(\frac{ef(e, a_p)}{\max_{e' \in ent(a_p)} ef(e', a_p)} \cdot w(p) \right) \tag{5.1}$$

We notice that the score takes into account both the number of entity occurrences and the part(s) in which the entity has been identified.

Now, as regards the importance of an entity in the whole set of top-L retrieved documents, we consider that the top-scored results probably contain more useful entities than the low-scored results (since they are considered better results for the user query). Thereby, the importance of an entity e in the set of retrieved documents is defined as:

$$HitScore_s(e) = \sum_{a \in docs(e)} (imp(e, a) \cdot score(a)) \tag{5.2}$$

We notice that the score is higher if the entity has been identified in the top-scored documents. In case the document scores are not given by the underlying search system, we can use the following more generic formula which takes into account only the ranking of the retrieved documents.

$$HitScore_r(e) = \sum_{a \in docs(e)} (imp(e, a) \cdot (1 - \frac{rank(a)}{|A| + 1})) \tag{5.3}$$

The advantage of this formula is that it is applicable also at a meta (uncooperative) search level where the document scores are usually not provided by the search system. Notice that this formula considers linear ranking of the retrieved documents. However the relation between scores and ranks usually is not linear (the scores of two consecutive hits may highly differ). Scores provide precise information for the actual relativeness of a document in regard to the user query. Thus, it is preferable to use scores instead of ranks (of course in case the underlying search system provides them). Moreover, in case the scores are not given, one could apply a different formula that considers, for example, exponential relation between the importance of an entity and the ranks of the documents that contain that entity.

5.3 Ranking of Entities and Properties

Here we detail the stochastic process for identifying the important semantic information related to the set of retrieved results.

5.3.1 The Semantically-Enriched Graph of Identified Entities

We first construct what we call Semantically-Enriched Graph of Identified Entities (SEGIE), which is an RDF graph, denoted by $\mathcal{X}$. We consider all entities in E as vertices in $\mathcal{X}$. Then, for each entity $e \in E$, each URI $u_e \in U(e)$, and each triple $(u_e, p, o) \in Descr(u_e)$ or $(s, p, u_e) \in Descr(u_e)$, we add to $\mathcal{X}$ the vertex o and the edge $e \rightarrow o$, or the vertex s and the edge $s \rightarrow e$, respectively. Furthermore, for two entities $e1, e2 \in E$ and two corresponding URIs $u_{e1} \in U(e1)$ and $u_{e2} \in U(e2)$, if $(u_{e1}, p, u_{e2}) \in Descr(u_{e1}) \cup Descr(u_{e2})$ or $(u_{e2}, p, u_{e1}) \in Descr(u_{e1}) \cup Descr(u_{e2})$, then we draw the corresponding edge p that connects the two entities. In addition, if the object o is a *blank node* b, and to avoid losing information that may be important, we include in the graph the set $out(b)$ and not the blank node b (if $out(b)$ contains a blank node, we then ignore it). In this case, for labeling the edge that connects an entity e with an entity e' in $out(b)$, we concatenate the names of the properties. For example, if e *"birth place"* b and b *"city"* e', then the name of the edge is "*birth place city*". Finally, we specially treat the `same-as` OWL property because it states explicitly that the corresponding object and subject refer to the same real-world object. Specifically, for a triple (s, p, o), if p is a `same-as` property then we merge the nodes that correspond to s and o. One could also apply here other "data cleaning" methods, e.g., apply an entity resolution/matching approach [192] aiming to identify entities that refer to the same real-world object (but not connected through a `same-as` property).

Consider the following example in the *marine* domain (which for now on it will be our running example): The user submits a query to a marine-related search system, e.g., the query "bonito". The search system applies NEE (with *fish species* as the EoI) in the snippets of the top-10 hits and identifies the following three entities:

- "Blackfin tuna" (`http://.../Blackfin_tuna`)
- "Striped bonito" (`http://.../Striped_bonito`)
- "Sarda" (`http://.../Sarda`)

Now, by accessing a SKB we retrieve the outgoing properties of each entity. Consider that for the entity "Blackfin tuna" the following properties are retrieved:

- `<http://.../isPredatorOf>  <http://.../Striped_bonito>`
- `<http://.../family>  <http://.../Scombridae>`
- `<http://.../bionomial>  _:b`
  `_:b  <http://.../authority>  <http://.../Lesson>`
  `_:b  <http://.../name>  "Thunnus atlanticus"@en`

For the entity "Striped bonito", the following 2 properties are retrieved:

- `<http://.../genus>  <http://.../Sarda>`
- `<http://.../family>  <http://.../Scombridae>`

While for the last entity, no properties are retrieved (the SKB does not contain triples describing information for this entity).

Figure 5.1(a) depicts the RDF graph of the above example. Node "`b`" corresponds to a

blank node, node "d" (*Thunnus atlanticus@en*) is a literal (specifically a string in English), while the other nodes are URIs (for improving readability we have omitted the namespaces). The corresponding SEGIE $\mathcal{X} = (\mathcal{E}_{\mathcal{X}}, \mathcal{P}_{\mathcal{X}})$ is shown in Figure 5.1(b). Informally, a SEGIE is an RDF graph which however does not contain blank nodes and `same-as` edges.

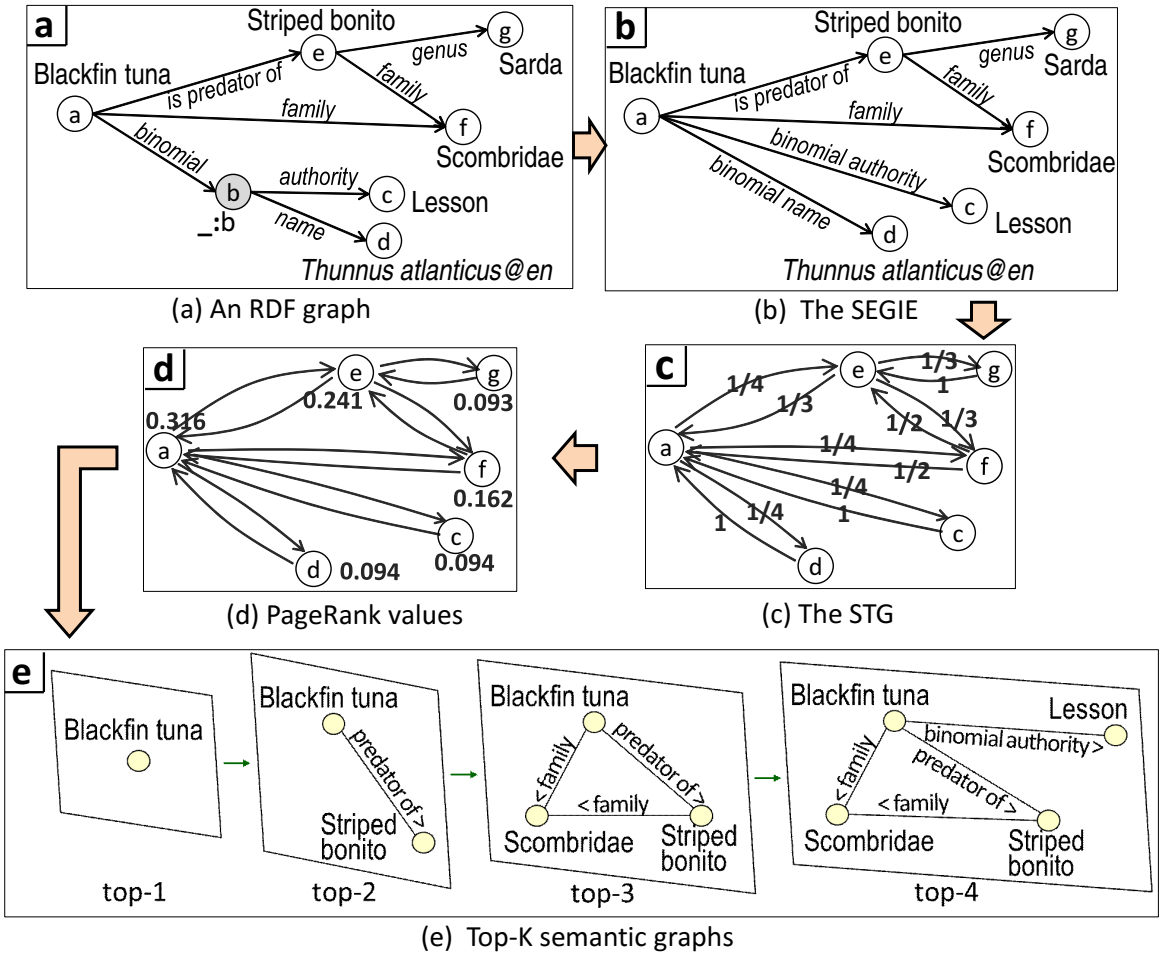


Figure 5.1: Stochastic ranking of entities and properties: a marine-related example.

5.3.2 The State Transition Graph (STG)

We will now define a STG $\mathcal{G} = (\mathcal{E}_{\mathcal{G}}, \mathcal{P}_{\mathcal{G}})$ over which a Random Walk model can be applied, i.e., its nodes $\mathcal{E}_{\mathcal{G}}$ correspond to *states* and its edges $\mathcal{P}_{\mathcal{G}}$ to (state) *transitions*.

For each node in $\mathcal{X}$, we create a node in $\mathcal{G}$. For each directed edge $(e_1 \rightarrow e_2)$ in $\mathcal{X}$, we create *two* directed edges in $\mathcal{G}$; one of the same direction $(e_1 \rightarrow e_2)$ and one of the opposite direction $(e_2 \rightarrow e_1)$. We do that because we consider that if a property connects two enti-

ties, then the two entities are *semantically biconnected*. For example, in Figure 5.1(a) the entities "Blackfin tuna" and "Scombridae" are connected by the relation *family*, meaning that "Blackfin tuna" *"belongs to the family"* "Scombridae" and equivalently that "Scombridae" *"is the family of"* "Blackfin tuna", i.e., the difference lies in how we name the property. In addition, in $\mathcal{G}$ we collapse multiple directed edges that connect two nodes in $\mathcal{X}$ in one directed edge, but we also specify accordingly the *edge weights*. Note that the weights of the outgoing edges from a given entity must represent transition probabilities, i.e., they must sum to 1. For a given entity $e \in \mathcal{E}_{\mathcal{X}}$, let $o(e) \subseteq \mathcal{P}_{\mathcal{X}}$ be the set of outgoing (directed) edges of e, and $i(e) \subseteq \mathcal{P}_{\mathcal{X}}$ the set of incoming (directed) edges. Let also $props(e, e') \subseteq o(e)$ be the set of (directed) edges that connect e with e' in $\mathcal{X}$ (i.e., the properties that connect the two entities). Note that in our setting $|o(e)| = |i(e)|$ and $|props(e, e')| = |props(e', e)|$. The weight of the single outgoing edge that connects e with e' in $\mathcal{G}$ is $\frac{|props(e,e')|}{|o(e)|}$.

Figure 5.1(c) depicts the STG that corresponds to the SEGIE of Figure 5.1(b). The figure also shows the edge weights as described above.

5.3.3 Analyzing the STG

The next step is to apply on STG a PageRank-like [161] algorithm for identifying the more *important* entities and properties. We prefer to follow a PageRank-inspired method because the underlying theoretical framework is solid (random walks and stochastic processes) and it can be customized (biased) according to the needs of different types of applications. The intuition behind PageRank, which was proposed and has been very successful in Web Search, is that the important Web pages are pointed by several other important Web pages. Analogously, in our problem an entity is considered important (and thus it is worth presenting to the user) if several other important entities point to it. PageRank simulates user's behavior of browsing on the graph nodes and works by counting the number and quality of links to a node aiming to determine a rough estimate of how important the node is. It outputs a probability distribution used to represent the likelihood that a person (random walker), who randomly selects edges, will arrive at any particular node.

The PageRank-like value $r(e)$ of an entity e is defined as:

$$r(e) = q \cdot Jump(e) + (1 - q) \cdot \sum_{e' \in i(e)} \frac{|props(e', e)|}{|o(e')|} r(e') \tag{5.4}$$

where q is a decay factor (typically set to 0.1-0.2), while $Jump(e)$ expresses the probability of random jumps to e, and thus it can be defined as $Jump(e) = \frac{1}{|\mathcal{E}_{\mathcal{G}}|}$ if we assume uniform distribution ($\mathcal{E}_{\mathcal{G}}$ is the set of STG nodes). The initial PageRank value of each entity also equals $\frac{1}{|\mathcal{E}_{\mathcal{G}}|}$. The equivalent matrix equation form is:

$$\mathbf{r} = q \cdot \mathbf{J} + (1 - q) \cdot \mathbf{T} \cdot \mathbf{r} \tag{5.5}$$

where $\mathbf{J}[e_i] = Jump(e_i)$ and $\mathbf{T}$ is the transition matrix:

$$\mathbf{T}(e, e') = \begin{cases} 0 & \text{if } (e' \rightarrow e) \notin \mathcal{P}_{\mathcal{G}} \\ \frac{|props(e', e)|}{|o(e')|} & \text{if } (e' \rightarrow e) \in \mathcal{P}_{\mathcal{G}} \end{cases} \tag{5.6}$$

Notice that the value of an entity e is the sum of two components: one part of the value is equal for all entities and expresses the probability of a random jump to e, and the other part comes from entities that point to e. The values can be computed iteratively and iterations should be run to convergence. According to [161], the number of iterations required for convergence is empirically $O(\log n)$, where n is the number of edges. Algorithm 1 describes the corresponding PageRank-like algorithm.

Algorithm 1 A PageRank-based algorithm for detecting the important entities and properties.

Require: $\mathbf{T}$ (transition matrix), $\mathbf{J}$ (random jumps matrix), q (decay factor), N (number of PageRank iterations).
Ensure: $\mathbf{r}$ (PageRank scores).

```
1: r = J //initialize the PageRank score of all entities
2: for i = 1 to N do
3:     r = q · J + (1 − q) · T · r
4: end for
5: return r
```

Figure 5.1(d) shows the PageRank values regarding the STG of Figure 5.1(c) (setting decay factor 0.15 and performing 10 iterations). The ranking is the following:

1. Blackfin tuna (node "a")
2. Striped Bonito (node "e")
3. Scombridae (node "f")
4. Lesson (node "c") and Thunnus atlanticus (node "d")
5. Sarda (node "g")

We notice that the entities "Blackfin tuna" and "Striped bonito" have the highest PageRank values because they have many connections and are also interconnected.

5.3.4 Promoting the Important Entities

So far we have ignored the importance (as defined in 5.2) of the identified entities, i.e., *HitScore*. To capture this, here we introduce a *biased* version of the scoring scheme. Instead of assuming a uniform distribution for the random jumps, we will now bias it. Specifically:

$$Jump(e) = \frac{HitScore_s(e)}{\sum_{e' \in E} HitScore_s(e')} \tag{5.7}$$

This means that the probability of a random jump to e is higher if e has been identified in the top-scored documents. The probability of a random jump to an entity that has not been identified in the search results is zero. Notice that, in case the document scores are not provided by the underlying search system, the above scoring scheme could be adjusted to use the ranking of the retrieved documents, i.e., $HitScore_r(e)$ (Formula 5.3).

To grasp the effect of the biased approach, in our running example consider that: *Striped bonito (node "*e*") was detected in the 1st, 2nd and 3rd result, Sarda (node "*g*") was detected in the 1st and 3rd result, while Blackfin tuna (node "*a*") was detected in the 8th result only.* For simplicity, in this example we exploit the ranking of the retrieved results, i.e., $HitScore_r(e)$. By running the biased version of PageRank as described above (with decay factor 0.15 and performing 10 iterations), we now get the following ranking:

1. Striped bonito (node "e"), with score 0.331
2. Blackfin tuna (node "a"), with score 0.260
3. Sarda (node "g"), with score 0.150
4. Scombridae (node "f"), with score 0.149
5. Lesson (node "c") and Thunnus atlanticus (node "d"), with score 0.055

Now the top-scored entity is *Striped bonito*, *Sarda* has gained two ranks, while *Blackfin tuna*, *Scombridae*, *Lesson* and *Thunnus atlanticus* have lost one rank. We notice that the entities identified in the top search results have been promoted.

One could exploit the biased version for supporting also various other kinds of *personalization*, e.g., promotion of entities of one or more specific categories (RDF classes) or those coming from specific SKBs, etc.

5.3.5 Top-K Semantic Graphs

Apart from producing and returning the top-K entities, say $\mathcal{E}_\mathcal{K}$ ($\mathcal{E}_\mathcal{K} \subseteq \mathcal{E}_\mathcal{G}$), as derived by the biased PageRank algorithm, we can return (at query time or on-demand) the top-K graph $\mathcal{G}_\mathcal{K} = (\mathcal{E}_\mathcal{K}, \mathcal{P}_\mathcal{K})$ for any K from 1 to $|\mathcal{E}_\mathcal{K}|$, allowing the user to increase or reduce the value of K. The set of edges $\mathcal{P}_\mathcal{K}$ of this graph consists of those elements of $\mathcal{P}_\mathcal{G}$ that connect elements of $\mathcal{E}_\mathcal{K}$, i.e., it is the *restriction* of $\mathcal{P}_\mathcal{G}$ on $\mathcal{E}_\mathcal{K}$, i.e., we can write $\mathcal{P}_\mathcal{K} = P_{\mathcal{G}|\mathcal{E}_\mathcal{K}}$.

Figure 5.1(e) depicts several top-K semantic graphs of our running example. The value of these graphs is that they show how the identified entities are connected as well as several important properties of the top-scored entities.

The Anatomy of a (Top-K) Semantic Graph

The possible different *types of vertices* in a semantic graph is an important information which must be taken into account in an exploratory search process, since it characterizes *groups of information* with some common properties. Since such a graph is actually an RDF graph, its vertices are either RDF resources or literal values (in our setting, by con-

struction a semantic graph does not contain blank nodes). However, some resources correspond to *categories of entities*, including resource classes (e.g., `dbpedia-owl:Fish`, `dbpedia-owl:Country`) and SKOS-like [10] concepts (e.g., `category:Fish_of_Hawaii`). Moreover, a resource may actually be a *Web address*, i.e., a URL that represents a Web page, photo, etc. We distinguish the vertices of a semantic graph as follows:

- $\mathbf{V}_{\texttt{ans}}$: Vertices corresponding to entities detected in the query answer. Each entity is associated with one or more URIs ($\mathsf{V}_{\texttt{ans}} \subseteq E$).
- $\mathbf{V}_{\texttt{rel}}$: Vertices corresponding to entities related to the detected entities. These resources have not been detected in the search results, but they were derived by exploiting the LOD. Each entity is associated with one or more URIs ($\mathsf{V}_{\texttt{rel}} \subseteq R$).
- $\mathbf{V}_{\texttt{ctg}}$: Vertices corresponding to categories. These resources have been derived by exploiting the LOD and each one is associated with a URI ($\mathsf{V}_{\texttt{ctg}} \subseteq R$).
- $\mathbf{V}_{\texttt{lit}}$: Vertices corresponding to literals, i.e., numeric values, strings, dates, or boolean value, derived by exploiting the LOD, e.g., birth date ($\mathsf{V}_{\texttt{lit}} \subseteq R$).
- $\mathbf{V}_{\texttt{web}}$: Vertices corresponding to Web addresses, i.e., URLs that represent Web pages, photos, etc. These URLs have been derived by exploiting the LOD ($\mathsf{V}_{\texttt{web}} \subseteq R$).

We call *clusters* these "groups" of vertices. A vertex can be included in only one of the above clusters and their union constitutes the set of vertices $\mathcal{E}_{\mathcal{K}}$, i.e., $\mathsf{V}_{\texttt{ans}} \cup \mathsf{V}_{\texttt{rel}} \cup \mathsf{V}_{\texttt{ctg}} \cup \mathsf{V}_{\texttt{lit}} \cup \mathsf{V}_{\texttt{web}} = \mathcal{E}_{\mathcal{K}}$. Figure 5.2 depicts an example of a graph containing vertices of these clusters. Considering that only the entity "Yellowfin tuna" was detected in the search results, vertex A is of type $\mathsf{V}_{\texttt{ans}}$, vertex B is of type $\mathsf{V}_{\texttt{rel}}$, vertex C is of type $\mathsf{V}_{\texttt{ctg}}$, vertex D is of type $\mathsf{V}_{\texttt{lit}}$, and vertex E is of type $\mathsf{V}_{\texttt{web}}$.

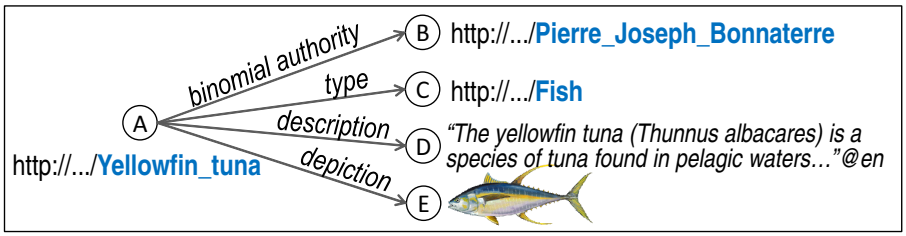


Figure 5.2: The main vertex types of a top-K semantic graph.

Of course, one could refine some clusters, e.g., dichotomize Web addresses to *images* and *Web pages*. One could even specify clusters according to the application context, e.g., a cluster that represents entities detected in a particular "interesting" set of results, etc.

The clustering of vertices in a semantic graph provides useful information which can be exploited by the interaction model and the underlying visualization/layout algorithm.

In 5.5.2, we report the results of a user study regarding the importance of these clusters in an exploratory search process. We also report experimental results regarding the distribution of vertices in these clusters for several top-K semantic graphs produced by a prototype system.

5.4 (Re-)Ranking of Search Results

Here we elaborate on whether and how we can exploit the identified entities and the related semantic information for *re-ranking* the list of retrieved results, aiming to promote relevant but low-ranked hits. The idea is to incorporate the documents in the graph of identified entities, and then to analyze the graph probabilistically.

Below we first present an exploratory searching scenario (from the user side) which allows to better motivate the Random Walk modeling that we propose, and then we detail the probabilistic analysis.

5.4.1 Modeling a Random Walker

We model the exploratory search process as a *random walker* of the graph defined by the documents, the mined entities and their connections. Specifically:

Whenever the walker is at a document hit d:

(a) With probability p_1 he jumps to another hit. The higher the relevance score/rank of a hit is, the higher is the probability to jump to that hit.

(b) With probability $1-p_1$ he moves to a node corresponding to an entity mined from d. The higher the entity importance score is (i.e., *HitScore*), the higher is the probability to move to that entity.

When at an entity e:

(c) With probability p_2 he jumps to a hit (based on the document scores/ranks as in (a)).

(d) With probability $1-p_2$ he follows one edge from e, specifically:

  (e) With probability p_3 he moves to a hit that contains e (based on the document scores/ranks as in (a) and (c)).

  (f) With probability $1-p_3$ he moves to a connected entity/property (equiprobably).

Figure 5.3 depicts the Markov chain of the corresponding stochastic process.

The above process actually models the behavior of a user in a Faceted Search-like environment and how the search results are consumed: the user submits a query and the system returns a list of results as well as entities identified in the results (e.g., in a left sidebar). The user can now open a result (it is more probable to open a highly scored/ranked

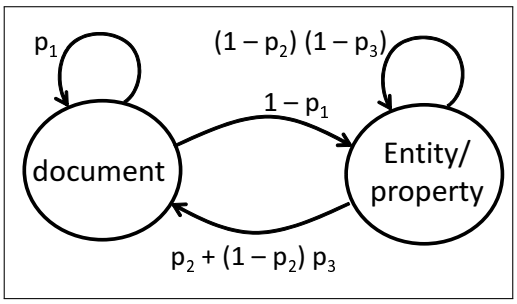


Figure 5.3: The Markov chain of the stochastic process.

result), or click on an entity and display only the results that contain the selected entity (it is more probable to click on a highly scored entity). In the latter case, the user can now either i) open one or more of the displayed results, or ii) click on some other entities and update the displayed list of results correspondingly, or iii) clear his selection (reset) and look again into the results.

5.4.2 The Semantic Graph of Documents and Entities

We describe how from a set of documents A and a set of identified entities E, we construct what we call Semantic Graph Of Documents and Entities (SGODE), denoted by $\mathcal{Z}$. We consider both the documents and the entities as vertices in $\mathcal{Z}$, while for drawing the edges we take into account the documents in which an entity has been identified. Specifically, we draw an edge (with name `identifiedIn`) starting from an entity e and ending to a document a, if $e \in ent(a)$ (i.e., e has been identified in a). Now, for each entity $e \in E$, each URI $u_e \in U(e)$, and each triple $(u_e, p, o) \in Descr(u_e)$ or $(s, p, u_e) \in Descr(u_e)$, we add to $\mathcal{X}$ the vertex o and the edge $e \rightarrow o$, or the vertex s and the edge $s \rightarrow e$, respectively. Furthermore, for two entities $e1, e2 \in E$ and two corresponding URIs $u_{e1} \in U(e1)$ and $u_{e2} \in U(e2)$, if $(u_{e1}, p, u_{e2}) \in Descr(u_{e1}) \cup Descr(u_{e2})$ or $(u_{e2}, p, u_{e1}) \in Descr(u_{e1}) \cup Descr(u_{e2})$, then we draw the corresponding edge p that connects the two entities.

Figure 5.4 depicts an example of a SGODE graph. The black nodes correspond to documents (set A), the gray to entities identified in the documents (set E), while the white to related properties/entities retrieved from a SKB (set R). Correspondingly, the solid edges connect entities to documents while the dashed edges connect entities with related properties/entities.

5.4.3 The STG

Now we describe how from $\mathcal{Z}$ we define a STG $\mathcal{H} = (\mathcal{E}_{\mathcal{H}}, \mathcal{P}_{\mathcal{H}})$. For each node n in $\mathcal{Z}$, we create a node in $\mathcal{H}$. For each directed edge $(n \rightarrow n')$ in $\mathcal{Z}$ we create *two* directed edges in $\mathcal{H}$; one of the same direction $(n \rightarrow n')$ and one of the opposite direction $(n' \rightarrow n)$. We do that

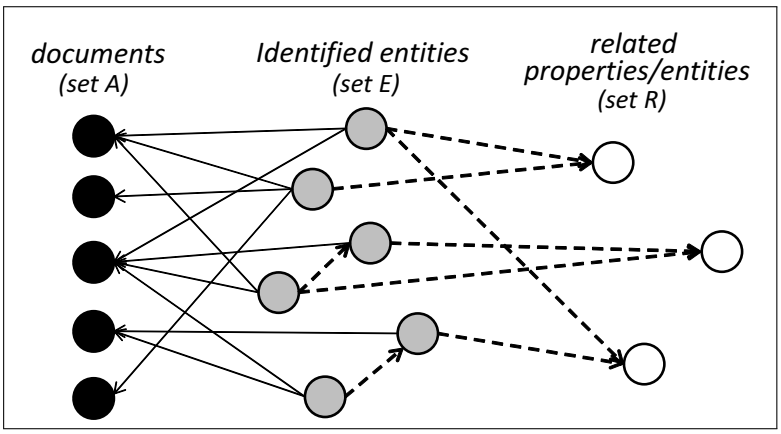


Figure 5.4: An example of a SGODE graph.

because we want the random walker to can be transferred both from a document-node to an entity-node and vice versa, as well as from an entity-node to another entity-node (or to a related property/entity-node) and vice versa.

Weighting the edges

In case the random walker lies in a document-node a, we consider the following formula for specifying the weights of the edges from a to entity-nodes $e \in ent(a)$:

$$weight(a \rightarrow e) = \frac{HitScore_s(e)}{\sum\limits_{e' \in ent(a)} HitScore_s(e')} \tag{5.8}$$

We notice that the transition probabilities are affected by the "importance" of the identified entities. Specifically, the higher the score of an entity is, the higher is the probability to move to that entity. Figure 5.5 depicts an example of a small STG of documents and entities showing also the edge weights as they are derived from the above formula. For simplicity and ease of comprehension, the graph includes only the edges from documents to entities.

Similarly, in case the random walker lies in an entity-node e, we consider the following formula for specifying the weights of the edges from e to document-nodes $a \in docs(e)$:

$$weight(e \rightarrow a) = \frac{score(a)}{\sum\limits_{a' \in docs(e)} score(a')} \tag{5.9}$$

We notice that now the transition probabilities are affected by the similarity scores given to the documents by the underlying search system.

An entity-node may also be connected with other identified entities or with related properties/entities (result of entity enrichment process). In this case the weights of the

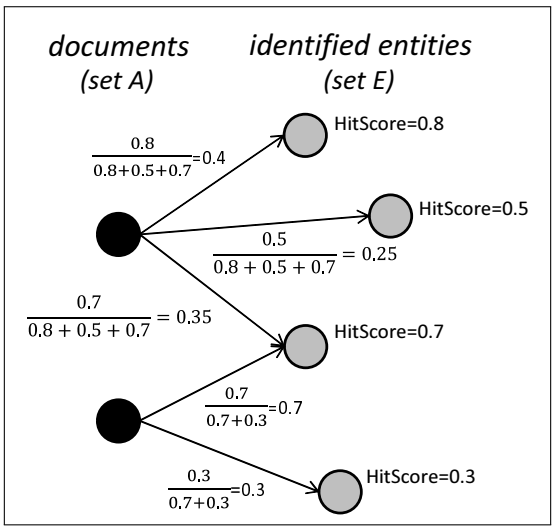


Figure 5.5: Biasing the link selection from a document-node.

edges are defined equiprobably as follows:

$$weight(e \rightarrow e') = \frac{1}{|edges_{out}(e)|} \tag{5.10}$$

where $edges_{out}(e)$ is the directed outgoing edges from the entity-node e to nodes that do not correspond to documents. However, the weights of the outgoing edges of a single node must represent transition probabilities, i.e., they must sum to 1. Thus, the weight from an entity-node e to a connected node n can be generally defined as:

$$weight(e \rightarrow n) = \begin{cases} p_3 \cdot \frac{score(n)}{\sum_{a' \in A} score(a')} & n \in A \\ (1 - p_3) \cdot \frac{1}{|out(n)|} & n \notin A \end{cases} \tag{5.11}$$

where p_3 is the probability to select a document-node. Figure 5.6 depicts an example of a small STG showing also the edge weights as they are derived from Formula 5.11 with $p_3 = 0.8$. For simplicity and ease of comprehension, the graph includes only the outgoing edges of the gray entity-nodes.

Finally, when the walker lies in a related property/entity node, he can move to the connected entity-nodes equiprobably. Figure 5.7 shows a corresponding example.

To sum up, the edge weight from a node n' to a connected node n is defined as:

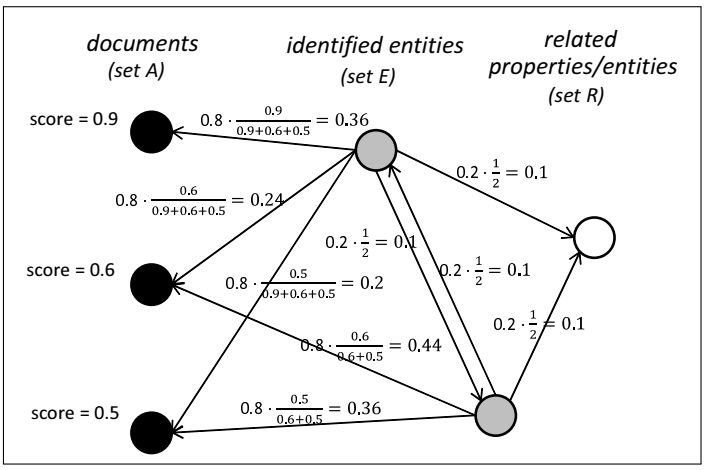


Figure 5.6: Biasing the link selection from an entity-node.

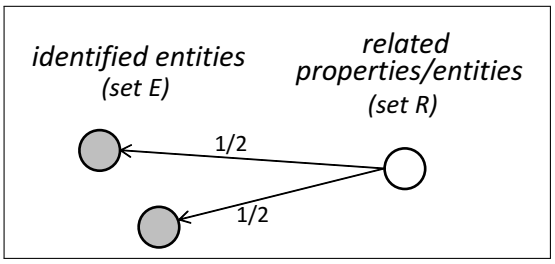


Figure 5.7: Equiprobable link selection from a related property/entity.

$$weight(n' \rightarrow n) = \begin{cases} \frac{HitScore_s(n)}{\sum\limits_{e' \in ent(n')} HitScore_s(e')} & n' \in A,\ n \in E \\ p_3 \cdot \frac{score(n)}{\sum\limits_{a' \in A} score(a')} & n' \in E,\ n \in A \\ (1-p_3) \cdot \frac{1}{|out(n')|} & n' \in E,\ n \notin A \\ \frac{1}{|out(n')|} & n' \in R,\ n \in E \end{cases} \tag{5.12}$$

5.4.4 Analyzing the STG

The objective is to find the probability the random walker to be in a specific document-node. For a node n, let $in(n)$ be the set of nodes that point to n. The PageRank-like value $r(n)$ is defined as:

$$r(n) = d \cdot Jump(n) + (1 - d) \cdot \sum_{n' \in in(n)} (weight(n' \rightarrow n) \cdot r(n')) \tag{5.13}$$

where d is the probability (decay factor) the walker to perform a random jump, $Jump(n)$ expresses the probability the walker to jump to the node n, and $weight(n' \rightarrow n)$ (as defined in Formula 5.12) is the probability the walker to visit n when being in a node n' connected to n (i.e., there is an edge from n' to n). The values can be computed iteratively and iterations should be run to convergence.

Random Jumps

As we have seen in our exploratory search scenario, we allow the random jumps only to nodes corresponding to documents. This means that the decay factor d in Formula 5.13 actually corresponds to the probabilities p_1 and p_2 of our modeling described in 5.4.1, i.e., $p_1 = p_2 = d$. In addition, we adjust the jump probabilities according to the document scores (instead of assuming a uniform distribution). Specifically, for a node n we consider the following formula for the random jumps:

$$Jump(n) = \begin{cases} 0 & n \notin A \\ \frac{score(n)}{\sum_{a' \in A} score(a')} & n \in A \end{cases} \tag{5.14}$$

which means that the probability the random walker to jump to a document is higher if the document has received a high similarity score from the underlying search system.

We should stress here that, in case the similarity scores given by the search system are not available, the above scoring formulas (Formulas 5.12 and 5.14) could be adjusted to use the ranking scores, i.e., $HitScore_r(e)$ and $rank(a)$, instead of similarity scores.

Tuning

To run the above Random Walk-based algorithm, it remains to tune some of its parameters:

Decay factor, p_3. At first we must specify a value for the decay factor, i.e., for the probability d the random walker to perform a random jump. In Web Searching, this value is typically set to $0.1 - 0.2$. In our context, a large d value favors the highly ranked documents and thus their final PageRank score, while a small value favors the connected entities which in turn favors the documents associated with highly scored entities. Note that for $d = 1.0$, the connectivity of the graph nodes is not considered (i.e., the identified entities are not taken into consideration), which means that the scoring is affected only by the random jumps,

i.e., by the document scores. Since we want to favor documents associated with important (highly scored) entities we can define a small value, e.g., $d < 0.4$.

Regarding p_3, i.e., the probability to select a document-node or a related entity/property node from an entity-node, we consider that when the walker is in an entity-node, it is more probable to move to a document node (that contains/refers this entity) since the final user target is to locate one or more documents that satisfy his information need. Thereby, we can define a big p_3 value, e.g., $p_3 > 0.6$. In case we want to allow only the selection of document nodes, we can define $p_3 = 1.0$.

Initial PageRank values. PageRank requires some initial values for the graph nodes. In Web Searching, PageRank is usually initialized with the same value for all Web pages. We also define a uniform distribution for the initial values. Specifically, for each node n we set $r(n) = 1/|\mathcal{E}_{\mathcal{H}}|$ ($\mathcal{E}_{\mathcal{H}}$ is the set of STG nodes).

Number of iterations. According to [161], the number of iterations required for convergence is empirically O(log$n$), where n is the number of edges.

5.4.5 Exploiting the Outcome

After running the above algorithm, all graph nodes receive a final PageRank score. The higher the score of a node is, the most important (and relevant to the search context) that node is considered. Documents with important (highly-scored) entities will receive a high score. The documents are finally re-ranked according to their PageRank scores.

We should stress here that such an approach to improve the search results (by re-ranking the top-L retrieved hits) is safer than other approaches that involve the risk of unexpected results (e.g., pseudo-relevance feedback [144]). In our case, an already retrieved list (containing documents that match the submitted query according to a retrieval model) is just re-ranked aiming to promote low-ranked but maybe relevant hits containing entities that are important in the current search context. Of course, the drawback in our case is that we cannot expect increment in the number of relevant hits in the top-L list (i.e., in the list that we analyze) since no new documents are retrieved.

5.5 Evaluation

At first (in 5.5.1) we report the results of a survey regarding the usefulness of top-K semantic graphs in the context of Marine (Professional) Search. We then (in 5.5.2) comparatively evaluate the effectiveness of the proposed probabilistic scheme for ranking entities and the related structured information. In 5.5.3 we experimentally evaluate the effectiveness of the proposed Random Walk model used for re-ranking a list of retrieved results. In 5.5.4 we report extensive experimental results regarding the applicability and the efficiency of the entire semantic analysis process. Finally, in 5.5.5 we compare the proposed

LOD-enriched ranking approach with the plain NER method, and we present the evaluation results of a related work that adopts our entity ranking method on the problem of constructing semantic snippets.

5.5.1 Usefulness of Top-K Semantic Graphs

In order to get a first feedback for the usefulness of the top-K semantic graphs derived by the entity and property ranking process described in 5.3, we performed a survey in the *marine* domain. The objective is to study whether the depiction of associations among the derived semantic information (through a top-K semantic graph related to the search results) can help the users (in our case marine biologists) in an exploratory search process.

Survey Setting

The survey is based on a questionnaire[1] (Google Form) in which we ask participants related to the marine domain (mainly marine biologists) to answer a few questions related to five particular queries (each query corresponds to a different *query type*). At first, for each query we derive the top-5 semantic information (i.e., entities and properties) by applying the proposed approach (described in 5.3), specifically by performing NER in the top-100 snippets as returned by Bing Web Search engine, with Fish Species (coming from DBpedia) as the EoI and using DBpedia as the SKB for retrieving the incoming and outgoing properties of the identified entities. Then, we depict this semantic information in two different ways: as a *top-5 list* and as *top-5 graph*. We select to show the top-5 list and graph (and not for example the top-10 or the top-20) because we do not want the quality of the visualization of the graph to affect the participants' opinion (since this is beyond the scope of this thesis). Then, the participant must answer the following question:

Q1. *"In an exploratory search process regarding the query* `<here the query>`*, how would you prefer to see the top-5 entities and properties related to that query?"*.

The participant can select one of the following options: *Only the LIST is enough*, *Only the GRAPH is enough*, *I would like to see BOTH*, *I do not want to see neither the list nor the graph*. Figure 5.8 depicts an indicative screenshot of the questionnaire in which the participant must answer Q1 for the query "yellowfin tuna". The participant must answer the above question for each one of the five queries. In the next step (in a new page), we ask the participant to answer the following question (again for each one of the five queries):

Q2. *"In an exploratory search process regarding the query* `<here the query>`*, do you believe that the appearance of a graph of semantic information related to the search results can help the user during his/her search process?"*

[1]The questionnaire is accessible at:
`https://docs.google.com/forms/d/1_qaKS2aYYgV6Xo2HBJ7AIXz79OQ936UwJeS3vI53bec/viewform`

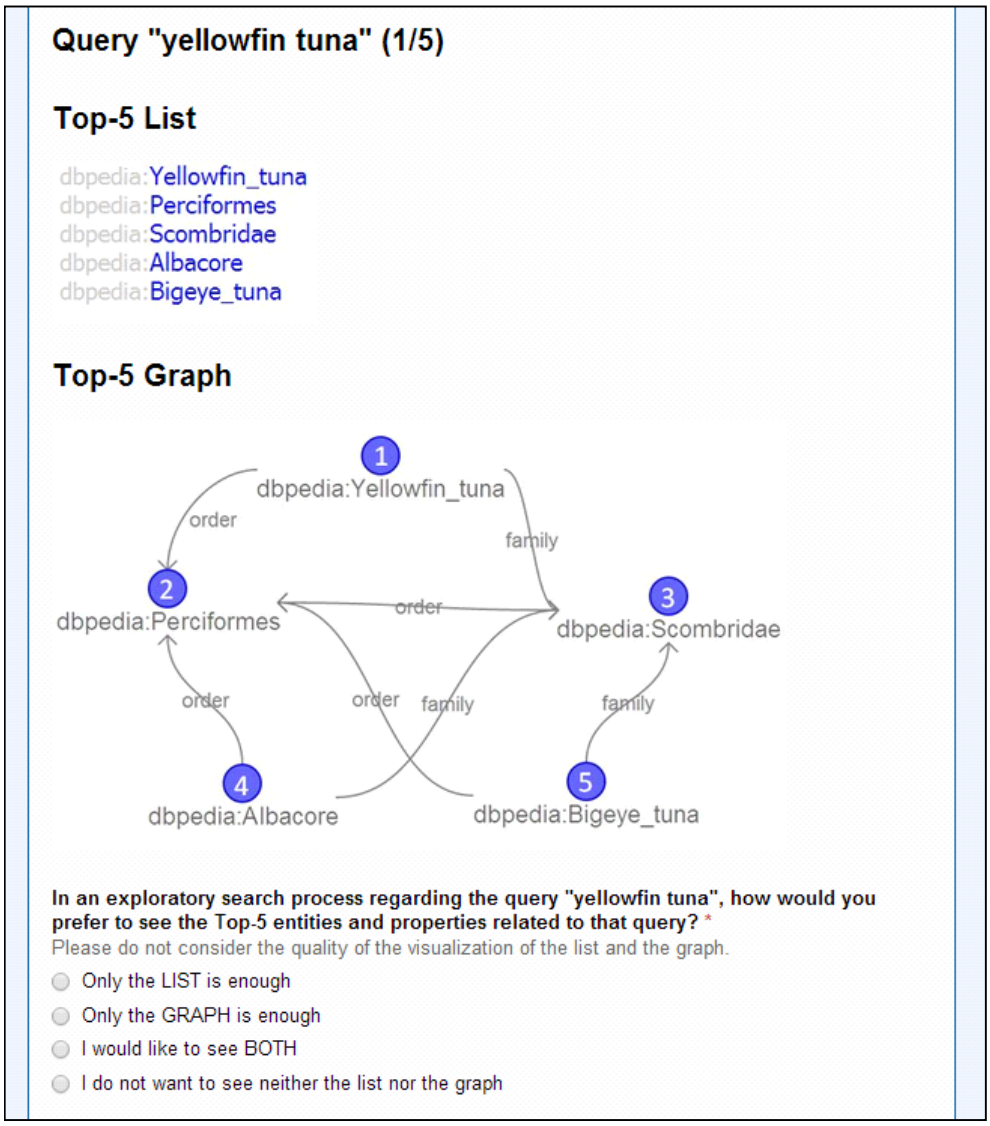


Figure 5.8: Indicative screenshot of the questionnaire in which the participant must answer the first question for the query “yellowfin tuna”.

The participant can select one of the following options: *Yes, Maybe Yes - it depends on the interaction model and the quality of the visualization of the graph, Maybe No, No.*

Clearly, the type of result expected depends on the *type* of the query. For example, a query such as `tuna species` is looking for instances of a class of entities, while a query like `yellowfin tuna` is looking for information for one particular entity, in this case a certain tuna species. Pound et al. [169] proposed a classification of queries from a semantic search point of view by expected result:

- *Entity query:* find information about a particular entity.

Table 5.1: Survey results.

Query	Q1					Q2				
	only list	only graph	both	no list, no graph	χ^2 test $p =$	yes	maybe yes	maybe no	no	χ^2 test $p =$
yellowfin tuna **(entity query)**	23%	30%	43%	3%	0.01857	23%	63%	10%	3%	$9.537e^{06}$
jack fishes **(type query)**	13%	37%	47%	3%	0.002262	27%	67%	7%	0%	$4.31e^{07}$
chum salmon genus **(attribute query)**	17%	37%	43%	3%	0.00694	33%	57%	7%	3%	$5.052e^{05}$
zander and walleye **(relation query)**	13%	43%	40%	3%	0.002905	33%	53%	13%	0%	0.000205
fishing in Hawaii **(other query)**	23%	37%	30%	10%	0.1979	30%	43%	23%	3%	0.01857

- *Type query:* find entities of a particular type or class.
- *Attribute query:* find values of a particular attribute of an entity or type.
- *Relation query:* find how two or more entities or types are related.
- *Other keyword query:* its intention is described by some keywords that do not fit into any of the above categories.

The existence of these query types is very important in our problem because each type may require a different result, thus it must be evaluated differently by the human judge. Thereby, the participants must answer the aforementioned two questions for five different queries, each one belonging to a different type. Specifically, we selected the following queries: `yellowfin tuna` (entity query), `jack fishes` (type query), `chum salmon genus` (attribute query), `zander and walleye` (relation query), `fishing in Hawaii` (other keyword query). Note that the selection of the queries to use in the questionnaire does not affect the purpose of this survey since this is not a task-based evaluation.

Survey Results

We distributed the questionnaire to marine biologists and to persons working on marine-related projects who have a basic knowledge on marine species. 30 subjects participated in the user study (22 to 60 years old), from 6 countries and 12 organizations[2]. Table 5.1 depicts the results. We should stress here that for a 95% confidence level (which means that there is only a 5% chance of our sample results differing from the true population average), a good estimate of the margin of error (or confidence interval) is given by $1/\sqrt{N}$, where N is the number of participants or sample size. Since 30 subjects participated in our survey, the margin of error is about 18%. This means that the reported results could vary by ±18%.

As regards Q1, we notice that the majority of the participants (67% to 84%) would like to

[2]Hellenic Centre for Marine Research, University of Carthage, University of Salento, University of Patras, University of Montpellier, TEI of Western Greece, University of the Aegean, AgroKnow Technologies, Foundation for Research and Technology - Hellas, University of Crete, CNRS-NCSM, EIHP.

see a graph representation (ONLY GRAPH or BOTH) of the top-5 entities. As expected, the biggest percentage is for the query `zander and walleye` which is a relation query, and the smallest is for the query `fishing in Hawaii` which does not belong to a particular type. In addition, we notice that for the first three types of queries (*entity*, *type* and *attribute*), more participants prefer to see both a list and a graph which means that the graph representation can be offered *on-demand* as a complementary representation which enables the users to inspect the connectivity of the identified entities.

Regarding Q2, we notice that the majority of the participants (73% to 94%) believe that the appearance of a graph of semantic information related to the search results can help users during a search process (YES or MAYBE YES). In addition, most of them consider that the success of this approach depends on the interaction model and the quality of the visualization of the graph (MAYBE YES). Notice also that if we consider both the percentages of users that selected ONLY GRAPH or BOTH in Q1 and the percentages of users that selected YES or MAYBE YES in Q2, the latter is always bigger. The above are a strong rationale for elaborating in the future on the interaction model and on the visualization of the top-K graphs. We also notice that for the last query (of type *other query*) which is a quite general query, a high percentage of participants (26%) selected MAYBE NO or NO.

Statistical significance. In order to check for the randomness of our results, we conducted a *Pearson's χ^2 statistical test (Goodness of Fit)* [167] which is appropriate for categorical data, testing the null hypothesis that the different answers of Q1 and Q2 respectively have been equally represented in our results. We used $\alpha = 0.05$ which is widely used in the bibliography. As regards Q1, the results depicted in the respective column of Table 5.1 show that we can reject the null hypothesis for the first four queries with a Type-I statistical error of 5%, while we cannot reject it for the last one, i.e., for the query *fishing in Hawaii* of type "other query". However, we should note that this query does not seem to fit a particular information need for marine biologists (it is a quite general query), and probably this is the reason for receiving many negative selections in both Q1 and Q2. As regards Q2, the results in the respective column of Table 5.1 show that we can reject the null hypothesis for all the queries. Here, again the last query has the highest p value. From the above results, we can support that the reported results are statistically significant for the first 4 types of queries, i.e., it is unlikely to have occurred by chance alone.

5.5.2 Effectiveness of Entity and Property Ranking

At first we should stress that there is not any standard evaluation procedure and collection for our purpose. Initiatives like INEX [76,105], TREC [27], and the SemSearch Challenge [9] cannot be applied in our problem because they focus on retrieving and ranking entities from a *structured* repository based on free text queries. On the contrary, our approach *semantically analyzes* (through NEE) a list of search results and focuses on identifying

the semantic information (entities and properties) that better characterizes these results. Thus, we designed and performed a user study in a *Marine Search* scenario for evaluating (comparatively with other ranking approaches) the effectiveness of the proposed ranking scheme.

User Study Setting

The objective is to evaluate the effectiveness of the proposed Random Walk scheme in ranking entities and properties (described in 5.3). We comparatively evaluated the proposed biased PageRank algorithm (`BiPR`) with i) the plain PageRank algorithm (`PR`) (assuming uniform distribution for the random jumps), and ii) a Spreading Activation [65] algorithm (`SA`). As regards the PageRank-based algorithms (BiPR and PR), we set decay factor 0.15 and we performed 50 iterations. Regarding `SA`, we adopted a similar approach with [183] and [17]. However, we set the initial activation of a node that represents an identified entity to be the score of this entity as derived by Formula 5.7, while the remaining nodes have zero activation. In addition, we set the decay factor a to equal 0.85 which in our setting appears to produce the best results (the decay factor corresponds to the percentage of activation that is lost every time an edge is processed). We also set the *firing threshold* to equal a very small real number (0.00001) because we want all nodes representing entities identified in the results to fire even if their ranking score is very small.

We deployed a Web application which implements the proposed functionality. Specifically, the system accepts keyword-based queries and performs NER in the top-100 snippets as returned by Bing, with Fish Species (from DBpedia) as the EoI and using DBpedia for retrieving the incoming and outgoing properties of the identified entities. We allowed the participants to submit their own queries so that they can better judge the results. We also stored the results for each submitted query as well as several statistics and features.

For each submitted query, the system presents three top-10 lists (the one next to the other with random display order) of ranked semantic information related to the results (entities and properties), each one produced by one of the aforementioned ranking schemes (`BiPR`, `PR` and `SA`). As regards `BiPR` and `SA`, and since the underlying search system (Bing) does not provide the document scores, we use the ranks of the retrieved documents, i.e., $HitScore_r(e)$ (Formula 5.3). The participant can evaluate the ranking of each list by selecting one of the following options: 1 (poor), 2 (not bad), 3 (good), 4 (very good), 5 (excellent). In addition, the participant can inspect many top-K lists, for several values of K ($5 \leq K \leq 50$), in order to better judge each ranking. We also included guidelines and a brief description of the proposed functionality allowing the participants to better understand the context of the evaluation. Figure 5.9 depicts a screenshot of the evaluation prototype showing the guidelines and the three top-10 lists for the query "tuna species".

Furthermore, we asked participants to answer a question regarding the *importance* of the different types of displayed semantic information. The objective is to get a first feed-

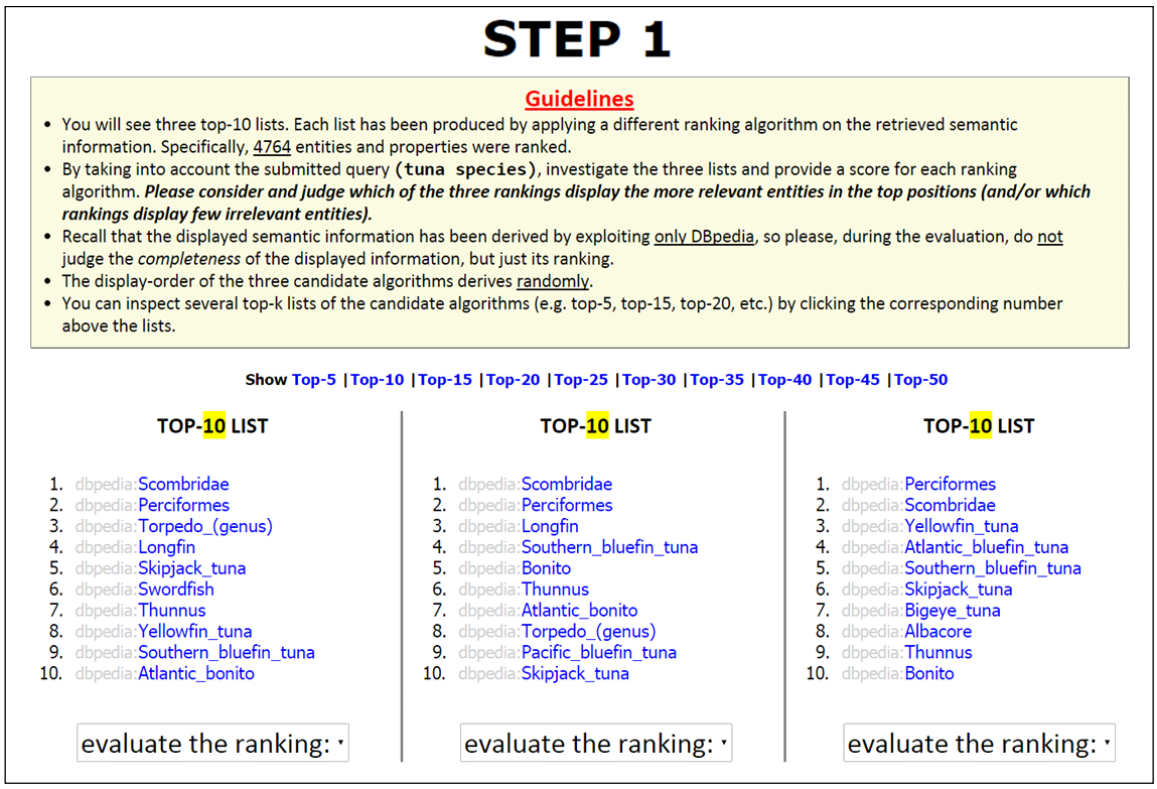


Figure 5.9: Indicative screenshot of the evaluation prototype showing the guidelines and the three top-10 lists for the query "tuna species".

back regarding the value of the entities and properties (that are derived by exploiting the LOD) in an exploratory search process. Specifically, we asked them to answer the following question:

"Taking into account the submitted query `<here the query>`*, please vote the importance of each category of displayed entities:*
- Fish Species detected in the search results
- Fish Species related to the detected species, but not detected in the search results
- Properties (literal and numeric values) related to the detected species
- Categories related to the detected species
- Web addresses (e.g., Web pages) or photos related to the detected species".

These types of entities correspond to the five clusters described in 5.3.5 (V_{ans}, V_{rel}, V_{lit}, V_{ctg}, V_{web}). The user can select a value between 0 (useless) and 5 (absolutely important). Figure 5.10 depicts a screenshot of the evaluation prototype showing the above question for the query "tuna species".

Figure 5.10: Indicative screenshot of the evaluation prototype showing the question for the query "tuna species".

User Study Results

Totally, 51 queries were submitted by 17 subjects (part of those who completed the survey in 5.5.1). For each submitted query, on average 11.5 entities were detected in the search results, 4,687 triples were derived from DBpedia, and 2,031 entities and properties had to be ranked.[3]

Figure 5.11 depicts the results. We notice that `BiPR` outperforms `PR` and `SA`, receiving many "4" (very good) scores. Specifically, the average score of `BiPR` was 3.53/5.0 (good to very good) and the median 4 (very good), the average score of `PR` was 2.47/5.0 (not bad to good) and the median 2 (not bad), while the average score of `SA` was 2.90/5.0 (almost good) and the median 3 (good). We can conclude that promoting the entities identified in the top results affects positively the ranking of the semantic information and thus the elements of the top-K graphs. Moreover, a biased PageRank-inspired method appears to produce better rankings than a Spreading Activation algorithm. Thus, we can also support that the random jumps affect positively the ranking process and that the semantic information that is accessible by many highly-scored entities is useful for the users.

Regarding the importance of the different types of entities, users considered the entities detected in the search results (V_{ans}) the most important semantic information with score 4.3/5, which is a predictable result since this information derives by analyzing the

[3]The full results, the identified entities and the RDF triples retrieved from DBpedia for each submitted query, and the top-200 rankings as produced by each one of the three ranking algorithms, are available to download through `http://users.ics.forth.gr/~fafalios/rankingEval/fullEvalResults.zip`.

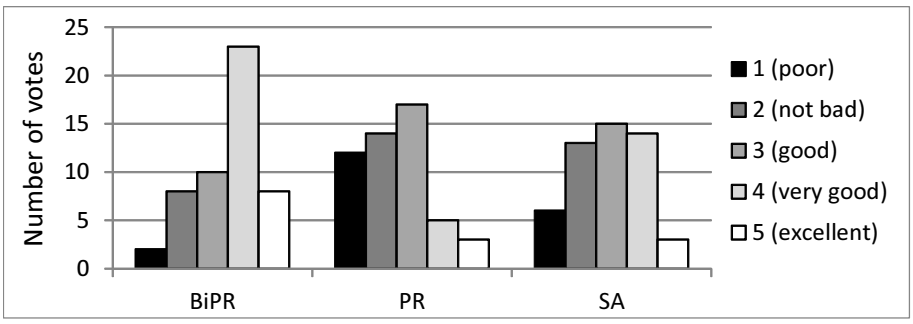


Figure 5.11: The scores given to the three ranking schemes.

results of a given user query. As regards the remaining clusters, their importance vary from $3.2/5$ to $3.7/5$ (V_{rel}= 3.3, V_{ctg}= 3.4, V_{lit}= 3.7, V_{web}= 3.2). We notice that the higher score is for the literal properties (V_{lit}) which correspond to characteristics of the detected entities. This is justified by the fact that the majority of the submitted queries where *entity queries* whose intention is to find information about a particular entity. In general, we notice that the score of all clusters is high meaning that users are interested not only in entities detected in the search results, but also in information related to the detected entities.

Graph Features and Efficiency

For each query submitted during the user study we measured several features of the semantic graphs derived by applying each one of the three ranking algorithms (`BiPR`, `PR` and `SA`). The objective is to inspect if the derived semantic graphs are well-connected as well as what types of vertices they contain, since such information is important for deciding how to visualize the graphs. Another objective is to inspect the difference of both the elements and the relative order of the common elements of several top-K graphs as derived by each one of the three ranking algorithms. If the difference is very low then the results of the evaluation may not be valid since the displayed lists look alike. Finally, and since the participants performed the evaluation in a real prototype system, we examined the average time for running each algorithm as well as the total time of the whole searching procedure.

Graph connectivity. Figure 5.12a depicts the average number of edges of several top-K graphs, for each one of the ranking algorithms. We notice that all algorithms produce graphs with many edges and, as expected, the more vertices the graph contains, the more ‘‘complex’’ the graph is because of the large number of edges. An interesting remark is that the semantic graphs produced by the SA algorithm have slightly more edges than the graphs produced by the PageRank-based algorithms. This can be justified by the fact that a PageRank algorithm applies random jumps, meaning that it can reach and score vertices

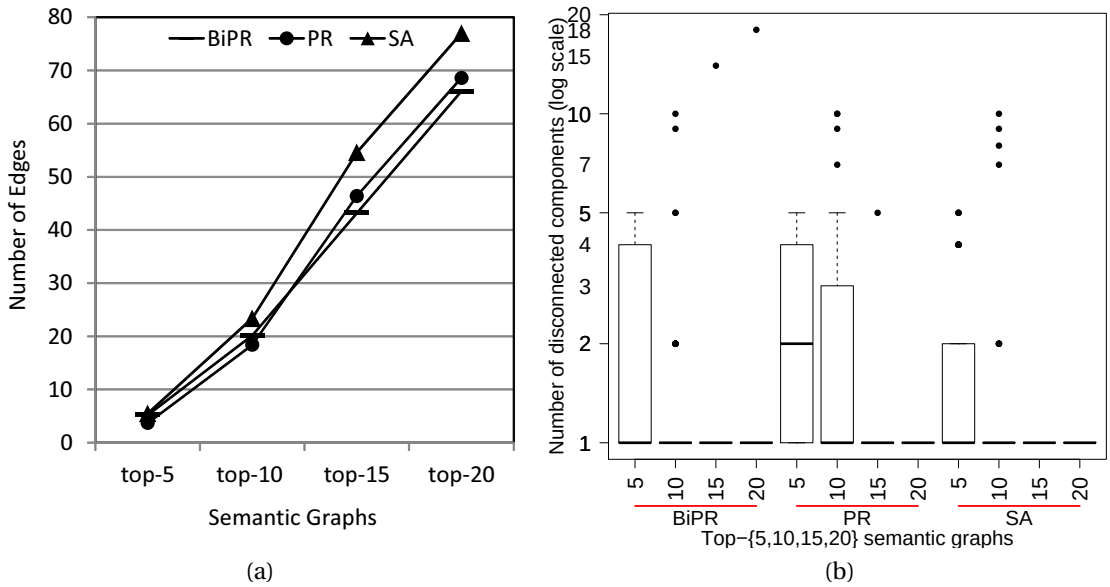
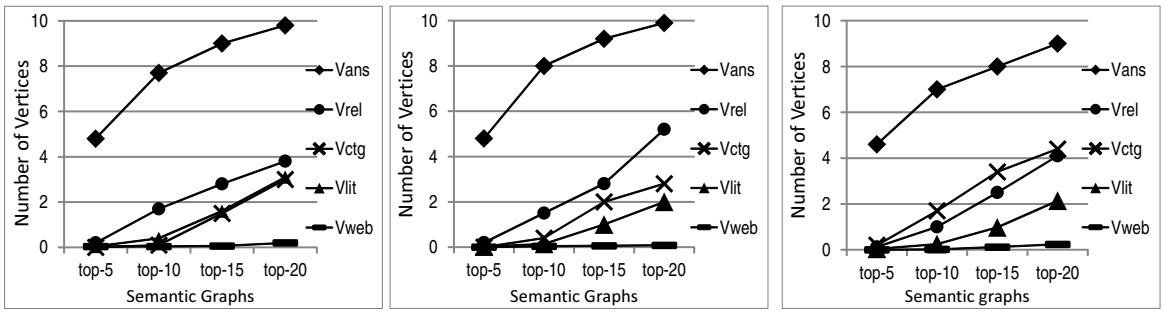


Figure 5.12: (a) Average number of edges and (b) number of disconnected components in top-5, top-10, top-15 and top-20 semantic graphs.

with few or no connections.

Figure 5.12b depicts a boxplot with the number of disconnected components of several top-K graphs (where one disconnected component means a well-connected graph). As shown, when $K > 5$ the graph is well-connected for the majority of cases. The only exceptions are the top-10 case of the PR algorithm where 50% of the queries result to a disconnected graph, and some other outlier queries for all algorithms (e.g., there is one query with 18 disconnected components). Regarding the top-5 answer, the PR algorithm seems to create the most disconnected components in the returned graph, with only 25% of the queries returning a connected graph, and more than 50% of the queries returning graphs with more than 2 disconnected components. It is interesting also that the average number of disconnected components is smaller in SA. This is justified by the ''nature'' of the spreading activation algorithm; the node weights decay as activation propagates through the graph. We should stress here that the *median* is 1 in all cases except for the top-5 graphs produced by PR where the median is 2 (i.e., in most cases the returned graph is well-connected).

Distribution of vertices. Figure 5.13 depicts the average distribution of the top-K entities and properties in the five clusters described in 5.3.5. We notice that in all cases the majority of vertices correspond to entities detected in the search results (type V_{ans}). We also see that in the top-5 graphs, almost all vertices are of type V_{ans}. This is a predictable re-



(a) BiPR (b) PR (c) SA

Figure 5.13: Distribution of vertices in the top-K semantic graphs.

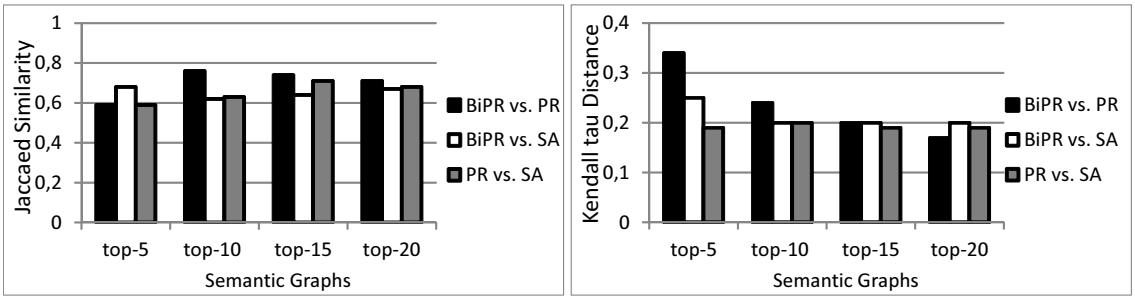
sult since: i) in the case of BiPR the vertices that correspond to the detected entities are promoted since they have a non-zero probability of random jump, ii) in the case of PR the vertices that correspond to the detected entities have much more connections compared to the other vertices (since we retrieved their properties from the underlying SKB), and iii) in the case of SA the vertices that correspond to the detected entities have a non-zero activation value.

We should stress here that both the connectivity of the top-K graphs and the distribution of vertices highly depend on the contents and the quality of the SKBs that we exploit. The more information a SKB contains, the more data we can retrieve for the detected entities, while if the resources of a SKB are well interconnected, the top-K graphs will be also well-connected. In addition, in the same SKB a category of entities may contain a lot of information about its instances, while another category may not be "rich" enough. Furthermore, some categories may contain many related entities (i.e., big number of $\mathsf{V}_{\mathtt{rel}}$), while others many literals (i.e., big number of $\mathsf{V}_{\mathtt{lit}}$). Therefore, the quality and the contents of the underlying SKBs highly affect the quality and the contents of the top-K graphs.

Jaccard Similarity and Kendall tau Distance. We compared the three ranking algorithms in terms of a) their difference in the top elements, and b) their difference in the order of their common elements.

As regards the difference in the top elements, we compared the algorithms using the *Jaccard* similarity coefficient. If A is the set of the top-k entities and properties returned by the one ranking algorithm and B the set of the top-k entities and properties returned by another algorithm, then their Jaccard similarity is $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Figure 5.14a depicts the results for all pairs of algorithms. We notice that in all cases the Jaccard similarity is between 0.6 and 0.75 meaning that the top-K entities and properties differ significantly.

As regards the difference in the order of the elements, we compared the algorithms using the normalized *Kendall tau* distance measure as described in 3.5.4. In our case we compared the linear orders produced by two ranking algorithms. In this comparison, we



(a) Jaccard Similarity (b) Kendall tau Distance

Figure 5.14: Similarity of the top-K semantic graphs.

considered as set E, the set of elements that exist in the corresponding top-K sets of both linear orders. Figure 5.14b depicts the results for all pairs of algorithms. We notice that the Kendall tau distance ranges from about 0.17 to 0.35, meaning that the difference in the linear orders is not negligible even when comparing only the order of the common elements.

Average Time. For each submitted query, we recorded the time required to run each one of the three algorithms. The time for running the PageRank-based algorithms (BiPR and PR) was about 80 ms, while the time for running SA was about 5 ms. We notice that the SA algorithm is more efficient, however the time for running the PageRank-based algorithms is also very low. We should also note that the total time (including the time to fetch the results, the time to perform entity mining in the top-100 snippets, and the time to retrieve the properties of the detected entities) was on average less than 3 seconds. This demonstrates the efficiency of the proposed approach for a marine-related scenario. In 5.5.4 we report the results of an *extensive* evaluation regarding the efficiency of each task of the proposed approach.

5.5.3 Effectiveness of Results (Re-)Ranking

Corpus and Setup

We evaluated the proposed probabilistic approach for re-ranking the list of retrieved results (described in 5.4) using the 2014 and 2015 datasets of the TREC Clinical Decision Support Track[4] [191]. The track focuses on retrieving biomedical articles relevant for answering generic clinical questions about medical records. The target document collection is a snapshot (containing 733,138 articles) of the Open Access Subset of PubMed Central (PMC)[5]. PMC is an online digital database of freely available full-text biomedical literature.

[4] `http://trec-cds.appspot.com/`
[5] `http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/`

We used Apache Lucene[6] 4.10.3 for indexing the collection (using its Standard Analyzer which finds word boundaries, downcases the words, and filters out stopwords) and we indexed the *title*, the *abstract* and the *body* of each document. As regards query evaluation and scoring, Lucene uses a combination of the Vector Space Model (VSM) and the Boolean model [144]. We selected to use Lucene as the baseline search system in our experiments because it is a classic, easy to use and widely applicable search engine. Of course, one could use another retrieval model that may be more effective in our medicine-related scenario (e.g., [220], [148] or [60]). However, our focus is to improve a list of retrieved results when IR has not performed very well, i.e., when the list of results contains some relevant hits only in the top positions of the answer (besides, a perfect list does not need any improvement). Note that IR can fail due to several reasons. For example, the user may have not accurately described her/his information need (this is very common in exploratory search needs). For this reason, in our experiments we use a widely applicable search system in its default settings, without paying particular attention to the retrieval model and the used queries.

We used the full *description* of each of the provided 60 topics for querying the collection. A topic is actually a medical case narrative serving as an idealized representation of an actual medical record and it describes information such as a patient's medical history, the patient's current symptoms, tests performed by a physician to diagnose the patient's condition, the patient's eventual diagnosis as well as the steps taken by a physician to treat the patient. For each provided topic, an effective IR system must find documents that can help the physician to answer a common generic clinical question such as what is the patient's diagnosis or what tests should the patient receive based on the medical report.

We performed NER in the indexed fields of top retrieved documents, and we used diseases, drugs, proteins, and chemical substances from DBpedia as the EoI. Regarding the weight of each indexed document part, we give 0.5 to the *title*, 0.3 to the *abstract*, and 0.2 to the *body*. We initially run experiments for different number of retrieved results (L= 100, 250, 500 and 1000), for two different decay factor values ($d = 0.0$ and 0.2) for the random jumps (which empirically provide better results), and without entity enrichment (i.e., p_3= 1.0). We then tested the case of entity enrichment for three different p_3 probabilities (0.25, 0.5, 0.75) by enriching the identified entities with the property `dct:subject`[7] from DBpedia (which seems to provide useful information for the specified EoI). We should also note here that a big decay factor (i.e., big probability of random jumps to document-nodes) does not make sense because the ranking will be mainly affected by the document scores/ranks and thereby the re-ranked and the initial lists are expected to be quite similar.

We compared the following lists of top-100 results:

[6] `https://lucene.apache.org/`

[7] In DBpedia, the `dct:subject` property (`http://purl.org/dc/terms/subject`) provides categories/groups in which the corresponding entity belongs.

- `BEFORE`: Initial top-100 list returned by Lucene
- `AFTER-d0`: Top-100 list after applying the proposed re-ranking approach with decay factor $d = 0.0$
- `AFTER-d2`: Top-100 list after applying the proposed re-ranking approach with decay factor $d = 0.2$
- `RANDOM`: Top-100 list after shuffling randomly the initial list returned by Lucene

For evaluating the results, we used the following evaluation metrics:

- `bpref`: as proposed in [43]
- `AveP'`: Average Precision based on a condensed list (after removing all unjudged docs) as proposed in [187]
- `nDCG'`: Normalized Discounted Cumulative Gain based on a condensed list as proposed in [187]
- `Q'`: Q-Measure on a condensed list as proposed in [187]
- `rpref_relative2`: as proposed in [187]
- `P@10'`: Precision at rank 10 based on a condensed list

In Appendix B we describe and give the formulas of the above metrics. Note that these metrics have been specially designed for evaluation environments with incomplete relevance data. We should also stress here that we cannot use the inferred metrics `infAP` [215] and `infNDCG` [216] because these metrics require knowledge of all pooled documents.

Results

At first we should point out that the top-100 `BEFORE` lists contain in average 11 relevant, 36 non-relevant and 53 unjudged hits. This means that Lucene did not manage to retrieve many relevant-for-sure documents in the top-100 lists, although the number of unjudged hits is quite large. This enforces the need for an effective re-ranking approach that can bring these few relevant-for-sure documents in higher positions in the returned ranked list. Moreover, in case more than 100 results are retrieved and analyzed, an effective re-ranking approach could promote in the top-100 list relevant but very low-ranked documents (in positions > 100).

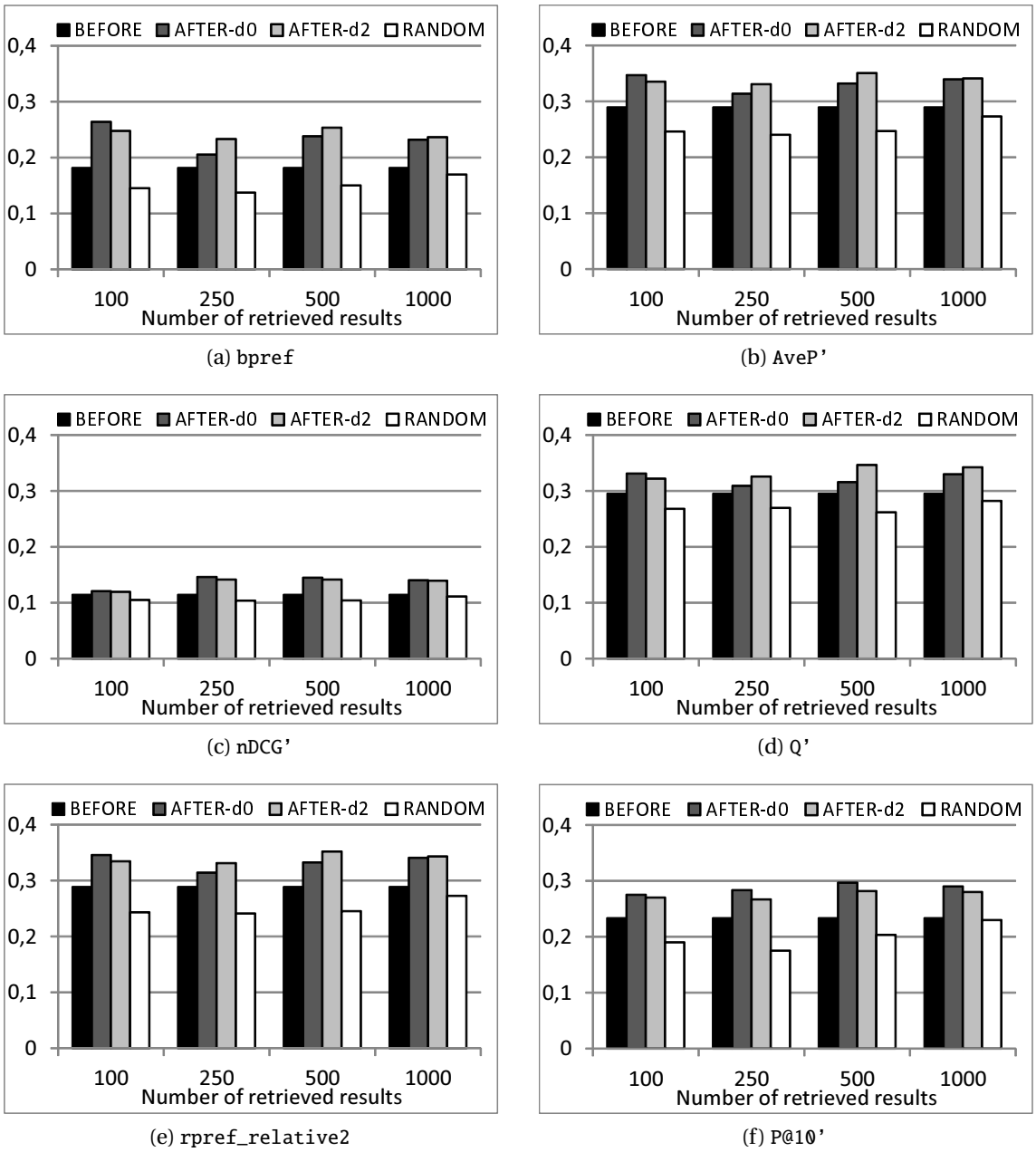
Figure 5.15 depicts the average results for all 60 topics. We notice that for all metrics and all numbers of retrieved results the top-100 lists are notably improved when the proposed re-ranking approach is applied for both $d = 0.0$ and $d = 0.2$. For instance, for $d = 0.2$ and $L = 500$ the average increment is about 40% in `bpref`, 21% in `AveP'`, 24% in `nDCG'`, 18% in `Q'`, 22% in `rpref_relative2`, and 21% in `P@10'`. This illustrates that the proposed method moved relevant hits in higher positions in the top-100 list. Furthermore, and for the cases where $L > 100$, re-ranking promoted in the top-100 list relevant hits which though had been ranked in positions > 100 in the `BEFORE` list. For example, and as regards the latter, for $d = 0.2$ and $L = 500$ the number of relevant-for-sure hits in the

top-100 list was increased for 43/60 topics (+1 for 13 topics, +2 for 6 topics, + >2 for 24 topics), was decreased for 7/60 topics (−1 for 5 topics, −2 for 1 topic, −3 for 1 topic), and remained the same for 10/60 topics. In addition, we notice that in all cases the random lists are worse than the initial lists. This somehow illustrates the non-randomness of our results. As regards the decay factor, we notice that for $L = 100$ the results are better when $d = 0.0$, while for $L > 100$, it seems that $d = 0.2$ performs better for the majority of metrics. For $L > 100$, re-ranking may bring new hits in the top-100 list (i.e., hits that did not exist in the initial top-100 list) and thus a non-zero decay factor somehow "limits" the movement of the highly ranked hits (recall that we compute the metrics for the top-100 lists).

We also conducted paired t-tests with α-level 5% for investigating if the improvement of the re-ranked top-100 list (`AFTER`) over the initial list (`BEFORE`) is statistical significant. The results showed that the improvement is statistical significant for the majority of cases (it does not pass the test when the improvement is small, less than 14%). Specifically:

- For $L = 100$, and for both $d = 0.0$ and $d = 0.2$, the results are statistical significant for all metrics.
- For $L = 250$ and $d = 0.0$ the results are statistical significant only for the metrics `nDCG'` and `P@10'`, while for $d = 0.2$ the results are statistical significant for all metrics except for `Q'`.
- For $L = 500$ and $d = 0.0$ the results are statistical significant for all metrics except for `Q'`, while for $d = 0.2$ the results are statistical significant for all metrics.
- For $L = 1,000$, and for both $d = 0.0$ and $d = 0.2$, the results are statistical significant for all metrics.

Improvement Failure. Although we saw that, in average, re-ranking improves the top-100 list returned by a classical IR system, by analyzing thoroughly the results for the case of $d = 0.2$ and $L = 500$ and for one of the evaluation metrics (specifically for `bpref`), we noticed that for 10 topics re-ranking failed to improve the top-100 list (i.e., the `bpref` value was reduced). By inspecting these cases, we noticed that for 7/10 topics the initial number of relevant-for-sure hits was above the average value. Moreover, only for 4 topics the reduction was above 0.05 and for 3 of these 4 topics, the number of relevant hits was above 30. From the above, we can conclude that when the number of relevant retrieved hits is high, re-ranking can affect negatively the results. Furthermore, surprisingly, for 8/10 topics the number of relevant-for-sure hits in the top-100 list was increased (+1 for 2 topics, +2 for 1 topics, + > 2 for 5 topics), while only for 2 of these topics it was decreased (−1 for the one and −2 for the other). This means that, although re-ranking moved some highly-ranked relevant-for-sure hits in lower positions (causing the reduction of `bpref` value), it nevertheless managed to bring more relevant hits in the top-100 list, increasing thereby recall. Notice that improvement in recall may be very important for search applications in professional domains where the main goal is to retrieve almost all documents that are



(a) bpref

(b) AveP'

(c) nDCG'

(d) Q'

(e) rpref_relative2

(f) P@10'

Figure 5.15: Average evaluation results for all topics.

relevant to an information need.

Effect of Entity Enrichment. We also tested the case of entity enrichment. Specifically, we used three different p_3 probabilities (0.25, 0.5, 0.75), keeping constant the decay factor ($d = 0.2$) and the number of retrieved results ($L = 500$), and we compared it with the no entity enrichment approach (i.e., for $p_3 = 1.0$).

Figure 5.16 depicts the results. We notice that entity enrichment did not improve the top-100 list and this is clear for all evaluation metrics. Furthermore, the smaller the value of p_3 is (i.e., the larger the probability for the random walker to select a related entity/property node), the worse the results are. This means that the specific semantic information about the identified entities (DBpedia `subject` property), although it might be quite useful in another context (e.g., for entity-based faceted search), it misleads the random walker and affects negatively the re-ranking of the retrieved results.

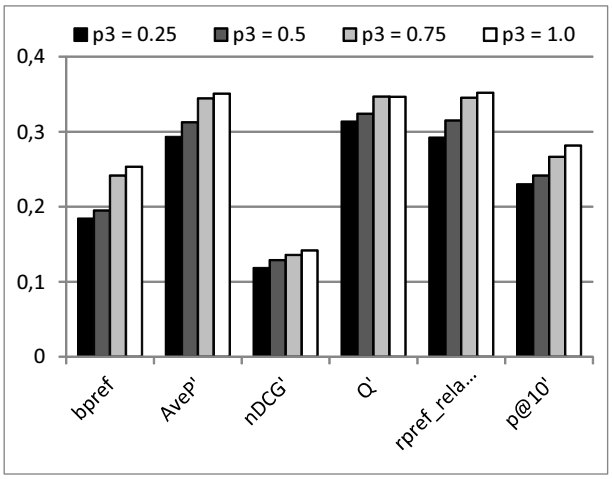


Figure 5.16: Effect of entity enrichment.

5.5.4 Efficiency of the Entire Semantic Analysis Process

Response Time

The time for performing entity mining in a set of search results depends on many parameters like the number of results we want to analyze, the size of the text of each result/document, the efficiency of the underlying NER algorithm, etc. For example, performing real-time entity mining (using `X-Link` and supporting 4 categories of entities) in the top-100 results returned by a Patent Search system costs about 1 second [86] (the cost is about 10 ms per result). Here we examine the average time for i) creating the SEGIE (with DBpedia as the underlying SKB), ii) creating the STG, iii) running PageRank, and iv) creating the top-500 graph, for various numbers of randomly selected entities. We run experiments for entities belonging to 10 randomly selected RDF classes (i.e., categories of entities): Tennis Player, Boxer, Country, Philosopher, Drug, Disease, Chemical Substance, Bacteria, Fish, and Golf Player. In a real setting, the randomly selected entities correspond to entities identified in the search results. For achieving accuracy, we repeated the experi-

Table 5.2: Graphs statistics and creation time, and PageRank execution time for several number of (randomly selected) entities.

# entities	SEGIE #vertices	SEGIE #edges	STG #edges	Top-500 Graph #edges	SEGIE creation time	STG creation time	Time for running PR	Top-500 creation time
50	2,573	3,790	7,580	889	1.4 sec	28 ms	194 ms	42 ms
100	4,133	6,193	12,386	1,493	2.9 sec	95 ms	329 ms	68 ms
500	20,743	34,816	69,632	3,471	13 sec	298 ms	1.7 sec	343 ms
1,000	49,954	84,893	169,786	3,411	27 sec	480 ms	3.9 sec	552 ms
10,000	528,815	995,981	1,991,962	3,421	258 sec	8 sec	58 sec	22 sec

ments 20 times and we computed the average values.[8]

Regarding i), i.e., the creation of the SEGIE, we decided to access DBpedia at *real-time* (and not to download its datasets and load them in one or more servers) because we want to preserve the dynamic nature of our approach (since the LOD constantly changes and increases). For each entity, DBpedia offers its data (properties and related entities) online in various forms: JSON, XML, triples and N3/Turtle. We access the data in the N3/Turtle form. As regards the objects in the triples that represent literals, we retrieve only those in English language. Although DBpedia offers a SPARQL endpoint, we decided to access its data by directly parsing the N3/Turtle pages because this fits our problem (for each entity we retrieve its incoming and outgoing properties) and because the parsing of pages turned out to be much more efficient than running SPARQL queries (allowing us to run concurrent threads). Table 5.2 reports the average values and also includes the main characteristics of the graphs in order to better understand how these characteristics affect the times.

As expected, the most time consuming task is the creation of the SEGIE since for each entity we access DBpedia at *real-time* and retrieve its related LOD. Notice that the time is linear in relation to the number of detected entities. For up to 100 detected entities (which is the common case for snippet-mining), the time is on average less that 3 seconds. For bigger number of detected entities the time is higher, e.g., for 1,000 entities about 30 seconds are required. However, such times are often acceptable in Professional Search, e.g., persons working in *patent* offices may spend many hours for a particular patent search request (the same is true in *bibliographic* and *medical* search).

The rest of the tasks require around one order of magnitude less time. We can conclude that the proposed functionality can be offered at real-time for about 100 identified entities even if we query an online SKB like DBpedia. In addition, as we have already said, this

[8]The experiments were carried out using an ordinary laptop with processor Intel Core i5 @ 2.4Ghz CPU, 4GB RAM and running Windows 7 (64 bit). The implementation was in Java 1.7 and for the creation and the management of the graphs we used the Java Universal Network/Graph Framework (JUNG) (`http://jung.sourceforge.net/`).

Table 5.3: Time dependencies table

Task	Time depends on:
Retrieving (top) results	underlying search system
Performing entity mining	i) num of results to analyze, ii) size of text of each result/document, iii) efficiency of the entity mining algorithm
Creating the SEGIE	i) num of detected entities, ii) underlying SKBs, iii) categories of the detected entities
Creating the STG	num of triples in SEGIE (i.e., number of edges)
Running PageRank	i) num of iterations, ii) number of edges in STG
Creating the top-K graph	num of vertices and edges in SEGIE

functionality can be also offered *on-demand* so the user can decide if she/he wants to pay the cost.

Time Dependencies

Table 5.3 synopsizes the time dependencies of each task of the proposed approach. The time for retrieving the top results depends only on the underlying search system. The time for performing entity mining in the top results depends on i) the number of the top results that we want to analyze, ii) the size of the text of each result/document, (e.g., in Web Searching mining the full contents requires much more time than mining the small snippets), and iii) the efficiency of the underlying entity mining algorithm which is affected by many parameters (e.g., number of supported categories, NLP/ML algorithm, etc.). The time for creating the SEGIE depends on the number of detected entities, the efficiency of the underlying SKBs and the categories of the detected entities (since some categories of entities may contain numerous incoming or outgoing properties). The time for creating the STG depends on the number of triples in SEGIE, while the time for running PageRank depends on the number of iterations and on the number of edges in the STG. Finally, the time for creating the top-K semantic graph depends on the size of SEGIE.

Scalability

As previously noted, the described process can be time consuming if the number of detected entities is big. Although such times (in the scale of minutes) are often acceptable in Professional Search, we can mitigate scalability issues as follows:

Bounding the response time. One reasonable (default) policy that we adopt for being scalable is to retrieve LOD (Step 3 of the process described in 2.4) only for the top-m (e.g., $m = 100$) identified entities based on their importance score (Formula 5.7). These top-m entities exist in most of the top-ranked results, therefore they are probably the more important. Thereby, we can bound the maximum response time.

5.5.5 Other Aspects

Plain Entity Mining vs. LOD-enriched

Below we compare the proposed LOD-enriched ranking approach (described in 5.3) with the plain NER method, i.e., with an approach that is based only on the *entity importance* (as defined in 5.2) and that does not exploit the structured knowledge that is available for the detected entities. Let A be the set of the top-K entities and properties as derived by the LOD-enriched approach, and let B denote the set of the top-K entities as produced by the plain entity mining approach. We want to answer the following questions:

(a) How different A and B are (if the difference is low then what we propose may have minor impact).

(b) Is the relative order of the entities which belong to both A and B equal or different? This allows testing whether the connectivity obtained through LOD affects the order; if only the biased jumps determine the ranking then A and B are expected to be quite similar.

(c) Understand the kind of elements of B which are not in A. Are they elements occurring in search results or not (if not then they were fetched from LOD).

We ran experiments using the 51 queries submitted in the user study described in 5.5.2, with Fish Species as the EoI and DBpedia as the underlying SKB. We performed NER in the top-200 snippets returned by Bing and, as regards the LOD-enriched approach, we retrieve the incoming and outgoing properties of each entity and we ran PageRank using decay factor 0.15 and performing 50 iterations.

Differences in the top-K entities. We compared the top-5, top-10, top-15 and top-20 entities using the *Jaccard similarity coefficient*. The results showed that on average the top-5 and the top-10 entities have Jaccard similarity about 0.65, the top-15 have Jaccard similarity about 0.60, while the top-20 have Jaccard similarity about 0.57. This means that they differ significantly.

Differences in the order of the common elements. Regarding the difference in the *order* of the entities, we use the *Kendall's tau* distance measure as described in 3.5.4. In our case we compared the linear order produced by the plain entity mining approach with the linear order produced by the LOD-enriched approach. In this comparison, we considered as set E, the set of entities derived by entity mining *only*, i.e., we ignored the entities that came from the LOD. The average *Kendall's tau* distance ranges from 0.23 to 0.3. If we recall that in the LOD-enriched approach the probability of a random jump to an entity is based on its importance score (Formula 5.7), it follows that the associations and the properties fetched from the LOD are responsible for a change of 23%-30% (which is not negligible) in the ranked order. Specifically, the stochastic analysis approach favors the entities that have many associations with other highly-scored entities.

In the same experiments we also compared the linear order of the top-5, top-10, top-15 and top-20 entities (not all the returned entities as previously), ignoring the entities that do not exist in the corresponding top-K entities of both approaches, with a view to clarify the positions in which there are differences in the linear order. On average, the top-5 entities have *Kendall tau* distance 0.08, meaning that often (but not always) there is a pair of entities in the top-5 list with different order in the two approaches. The top-10 entities have *Kendall tau* distance 0.10 (meaning that there are about 4-5 pairs of entities in the top-10 list with different order), while the top-15 and top-20 entities have *Kendall tau* distance about 0.20.

Biased Stochastic Analysis for Constructing Semantic Snippets

Alsarem et al. [21] elaborate on the same problem of ranking entities identified in a set of top results returned by a search system, focusing on the construction of semantic snippets. They introduce an algorithm, called `LDRANK`, which combines the biased ranking method that we propose in 5.3 (taking thereby into account the ranking of the search results) and a variation of the Singular Value Decomposition (SVD) algorithm that exploits textual data associated with the identified entities (specifically the abstract from DBpedia and a text window from the corresponding Web page).

The authors comparatively evaluated (using a dataset created through crowdsourcing): i) the biased PageRank algorithm that we propose in this thesis (they call it `HIT`), ii) a biased PageRank algorithm modified with SVD-based prior knowledge (called `SVD`), iii) an unmodified PageRank algorithm which applies equiprobable distribution (called `EQUI`), and iv) `LDRANK` which combines the previous three approaches using a consensual opinion pool algorithm [50].

Figure 5.17 (from [21]) shows the NDCG scores at all ranks from 1 to 100. The results illustrate that our method slightly outperforms `SVD`, `EQUI` is the worse strategy, while the consensual combination of all approaches (`LDRANK`) produces the best ranking. Finally, the authors report that `SVD` takes more time than `HIT` because it needs to compute the SVD, while the additional time spent by the combined strategy is due to the time necessary to converge towards a consensus.

5.6 Epilogue

We have introduced a biased stochastic method for identifying the semantic information (entities and properties) that better characterizes the search results. The proposed approach exploits both the ranking of the retrieved results, the extracted named entities and their connectivity, and is also exploited for producing and showing top-K semantic graphs. A top-K semantic graph can complement the query answer with useful information re-

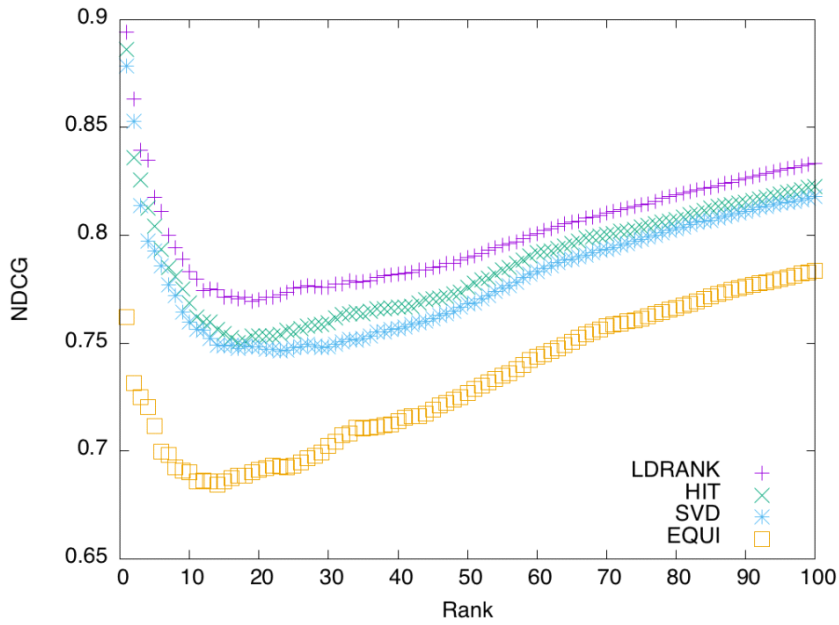


Figure 5.17: Comparison of the nDCG scores for different entity ranking strategies (Figure from [21]).

garding the connectivity of the identified entities. Furthermore, we have introduced an entity-based Random Walk model for re-ranking the results retrieved by a search system, aiming to promote relevant but low-ranked hits containing important (for the search context) entities. The key evaluation results can be summarized as follows:

As regards the usefulness of the top-K semantic graphs:

- The conducted survey in the marine domain showed that the majority (more than 70%) of the participants a) would like to see a graph representation of the top-5 entities regardless the type of the submitted query, and b) believe that the appearance of a graph of semantic information related to the search results can help them during an exploratory search process. The conducted statistical significance test rejects the randomness of the reported results with a 5% Type-I error.

As regards the effectiveness of the proposed method for ranking entity and properties:

- The results of the conducted user study illustrated that the proposed (biased Random Walk-based) ranking scheme produces more preferred rankings compared to other algorithms (about 22% better ranking compared to a Spreading Activation algorithm and 43% compared to the plain PageRank algorithm). In addition and regarding the importance of the different types of entities, the results showed that users are

interested not only in entities identified in the search hits, but also in data related to the detected entities.

As regards the effectiveness of the proposed results re-ranking method:

- Experimental results over the 2014 and 2015 datasets of the TREC Clinical Decision Support track showed that the proposed re-ranking approach can notably improve the list of results returned by a classical IR system by moving relevant but low-ranked hits in higher positions. However, additional semantic information about the entities (properties and related entities) can mislead the random walker and affect negatively the results. The same can happen when the initial list contains a big number of relevant hits. Nevertheless, the results showed that even if the initial answer is very good, re-ranking can increase recall.

As regards the efficiency of the entire semantic analysis process:

- The results of the conducted experimental evaluation showed that for up to 100 detected entities (which is the case for snippet-mining), we can offer the proposed functionality at real-time (in less than 4 seconds) even if we access an online SKB like DBpedia. Nevertheless, we have seen an approach on how we can bound the maximum response even for very big number of identified entities.

As regards the exploitation of the proposed biased ranking method in the problem of constructing semantic-snippets:

- The results of a comparative evaluation conducted by Alsarem et al. [21] showed that our method outperforms other approaches, while a consensual combination of three ranking methods (including the biased method that we propose) produces the best results.

Chapter 6
Interaction Model and Applications

In this chapter we elaborate on how the user can interact with the result of the proposed semantic analysis process (as introduced in 2.3) for supporting exploratory searching. As a proof of concept, we present related applications that we have designed and implemented as well as our experiences.

We focus on an interaction that:

(a) enables users to locate easily and fast even low ranked hits which though are relevant to their information need
(b) enables users to get "fresh" information related to the search context without having to disengage from their initial task (e.g., what are the main characteristics of an entity, how two or more entities are associated, etc.)

The first objective can be supported by extending the widely used faceted interaction scheme [185] with the results of NEE. For tackling the second objective, we investigate an approach where the user can on-demand inspect semantic information related to one or more of the identified entities.

Below, we first discuss how the user can interact with the elements of the derived semantic graph (6.1). We then show how the user can retrieve more (semantic) information about the identified entities (6.2). In 6.3 we present `X-Search`, a meta-search system we have designed and implemented which supports the proposed search process, configurability and interaction model, and which is currently used in a data infrastructure regarding fisheries management and conservation of marine living resources. In 6.4 we present `Theophrastus`, an entity-based annotation system that supports the automatic and configurable annotation of Web documents. Finally (in 6.5), we discuss the main weaknesses and limitations of our approach, and how we can cope with them.

6.1 Faceted Search and Graph Exploitation

One question is how to present the elements of the derived semantic graph (SEGIE with ranked vertices as described in 5.3). We have identified two basic forms (which can also be in conjunction):

(I) A top-K list of identified entities is provided (i.e., SEGIE's elements of type V_{ans}). The entities can be grouped in categories (facets), or alternatively, each entity can be qualified by its category name in parentheses, e.g., see Figure 6.1 (the number in parenthesis is the number of hits containing the corresponding entity name).

(II) The top-K semantic graph is provided (as described in 5.3.5), e.g., see Figure 6.2.

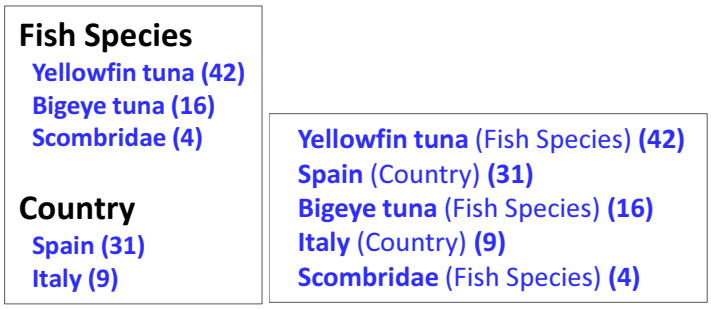


Figure 6.1: Identified entities grouped in facets (left) vs. list of identified entities (right).

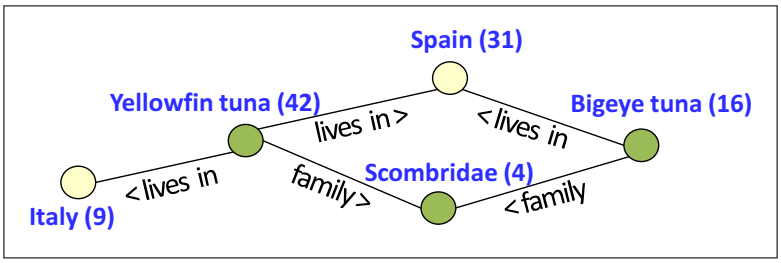


Figure 6.2: A top-5 semantic graph.

Below we first describe each one and then we dicuss their pros and cons.

In both cases, a default value of K can be used, e.g., K = 5. The user can specify this value through the User Interface (UI), and the UI can allow changing it dynamically. For case (I), the classical faceted search interaction scheme [185] can be used that allows users to restrict their focus or information need gradually and exploits the results of the previous steps through a session-based mechanism. Notice that this is actually an approach to *group* the search results. In 3.3 we discussed works which have shown that grouping the results improves the search experience especially in Professional Search.

For case (II), when the user clicks on a node of type V_{ans}, the hits that contain that entity can be shown. The user can also gradually increase or decrease K (as illustrated in Figure

6.3) allowing thereby to control the amount of information that she/he wants to display and consume. The semantic graph can be also provided on-demand, as a complementary representation of the identified entities which enables the user to inspect their connectivity. Figure 6.4 depicts several top-K graphs for the query "bonito" (of the marine-related scenario described in 5.3). For visualizing the graph, we can use a common force-directed (spring-style) layout algorithm [98, 219].

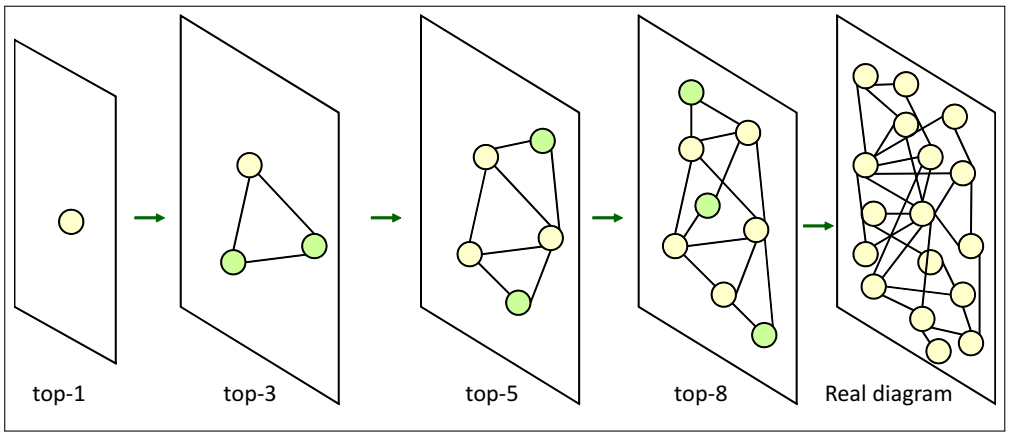


Figure 6.3: Progressive visualization based on top-K diagrams

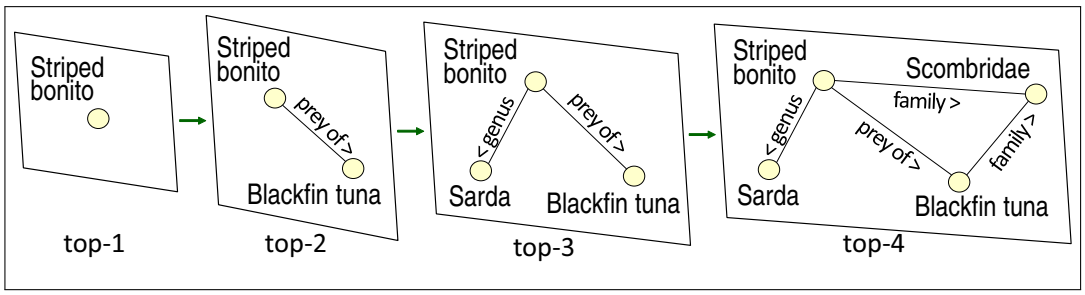


Figure 6.4: Progressive visualization of top-K semantic graphs in a marine-related scenario.

The strong benefit of case (I) is the fact that users are familiar with Faceted Search, while its drawback (in comparison to II) is that the user gets no information about the main characteristics and the connectivity of the displayed entities. Conversely, the benefit of case (II) is the informativeness of the top-K semantics graphs. However, visualizing such graphs in an aesthetic and effective way with high readability degree (especially for big values of K) is a challenging issue [114]. Furthermore, it requires specifying the interaction model between the graph and the end-user, which in turn requires training users on how to use it. These are strong rationales for offering case (II) *on demand* as a complementary representation which enables the users to inspect the connectivity and the main

characteristics of the top-K identified entities.

Exploiting the Semantic Graph for More Advanced Faceted Search

By exploiting the entity associations and properties, one could further enrich the information space to allow more sophisticated and expressive restriction of search results. Specifically, the results can be also restricted to those containing not only one or more particular entities (as in the classic Faceted Search), but to those results containing entities that share some common characteristics. This, for example, allows a user to browse for hits referring *fishes of genus "thunnus"*, or to those referring *sturgeon fishes* (i.e., fish belonging to the DBpedia category `http://dbpedia.org/resource/Category:Sturgeons`), or even using characteristics in bigger depth/radius (e.g., hits referring *fishes that have a predator of genus "thunnus"*). Figure 6.5 shows a screenshot of a prototype system that implements this "advanced" faceted browsing functionality. In this example, the user has restricted the search space to 10 results referring fish species of genus *"thunnus"*.

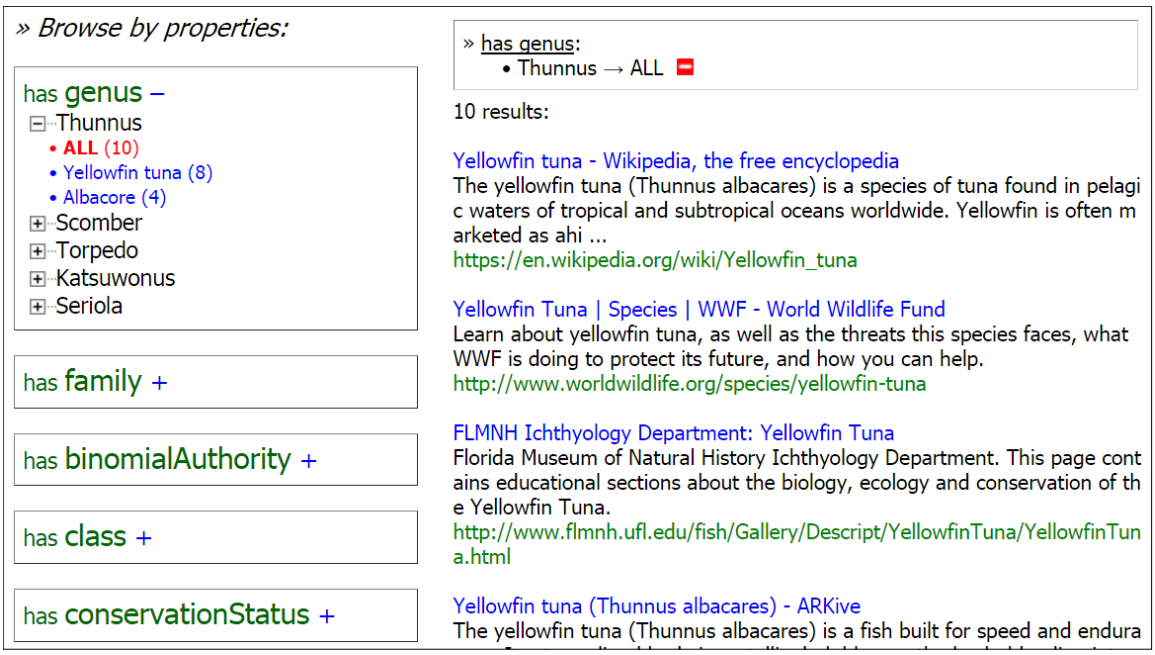


Figure 6.5: Restriction of search results based on entity properties.

Note that if we express the enhanced answer using the Open Annotation Extension (as we saw in 4.4), we can feed the resulting RDF triples to a system that supports faceted exploration of RDF data (like `Sparklis`[1] [96, 124] or SemFacet[2] [103]) and offer the aforementioned browsing capabilities. See [199] for a survey of such approaches.

[1] `http://www.irisa.fr/LIS/ferre/sparklis`
[2] `http://www.cs.ox.ac.uk/isg/tools/SemFacet/`

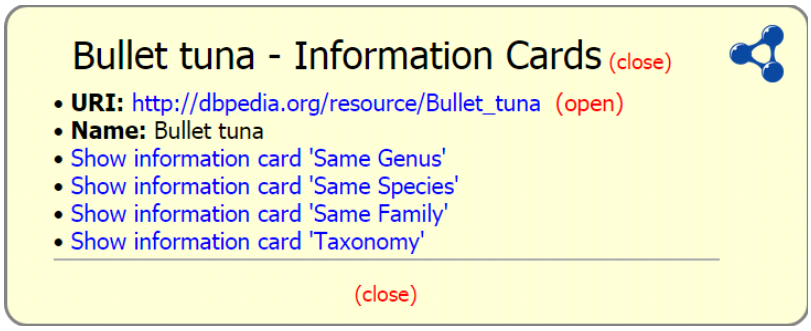


Figure 6.6: A list of available information cards for the fish species "Bullet tuna".

6.2 On-Demand Entity Enrichment and Exploration

Recall the objective (b), i.e., *"to enable users to get "fresh" information without having to disengage from their initial task"*. As we saw in Chapter 4, the availability of LOD datasets enables not only to configure easily the entity names that are interesting for the application at hand, but also the enrichment of the entities with more information about them. In this way, users not only can get useful information about an entity without having to submit a new query, but they can also start browsing the entities that are linked to that entity. Another important point is that exploiting LOD is more dynamic, affordable and feasible than an approach that requires each search system to keep stored and maintain its own SKB of entities and facts.

Returning to our setting, the main question is what information to retrieve for the identified entities. For each supported category of entities, the user/administrator can specify one or more *information cards*. An information card is a piece of information regarding an entity that is interesting to the community that uses the search application. For example, regarding a Fish Species, an information card could be its taxonomy, its predatory fishes, available synonyms, species that belong to the same family or genus, etc. Likewise, in the medicine domain, the information card of a Drug could be its chemical formula, interactions with other drugs, and so on. The semantic data that Google displays together with the returned results (in a right panel) when the submitted query corresponds to an entity, is very close to the notion of an information card. Figure 6.6 depicts a pop-up window displaying a list of available information cards for the fish species "Bullet tuna".

An information card actually corresponds to an *Enrichment Template Query*, as it has been described in the configuration model (4.2), thus it is associated with a name (card name) and a SPARQL template query. For example, the information card in Figure 6.7 depicts the taxonomy of the fish species "Chaunax" as returned by DBpedia (the corresponding template query is shown in Figure 6.8), while the information card in Figure 6.9 displays fish species that are of the same genus as the fish "Bigeye tuna" (the corresponding template query is shown in Figure 6.10).

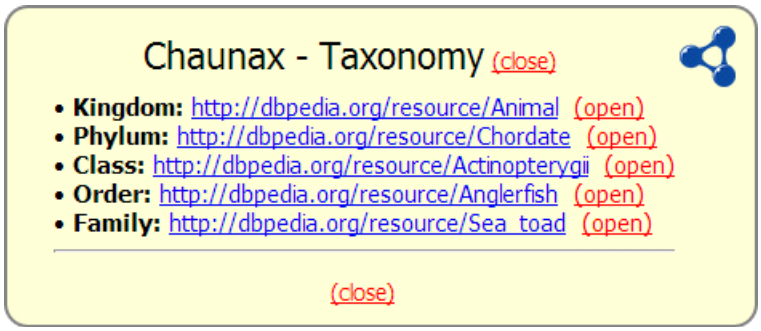


Figure 6.7: An information card displaying the taxonomy of the fish species “Chaunax”.

```
SELECT * WHERE {
  OPTIONAL { [URI] <http://dbpedia.org/ontology/kingdom> ?Kingdom . }
  OPTIONAL { [URI] <http://dbpedia.org/ontology/phylum> ?Phylum . }
  OPTIONAL { [URI] <http://dbpedia.org/ontology/class> ?Class . }
  OPTIONAL { [URI] <http://dbpedia.org/ontology/order> ?Order . }
  OPTIONAL { [URI] <http://dbpedia.org/ontology/family> ?Family . }
  OPTIONAL { [URI] <http://dbpedia.org/ontology/genus> ?Genus . }
  OPTIONAL { [URI] <http://dbpedia.org/ontology/subgenus> ?Subgenus } }
```

Figure 6.8: SPARQL template query for retrieving the taxonomy of a fish species from DBpedia.

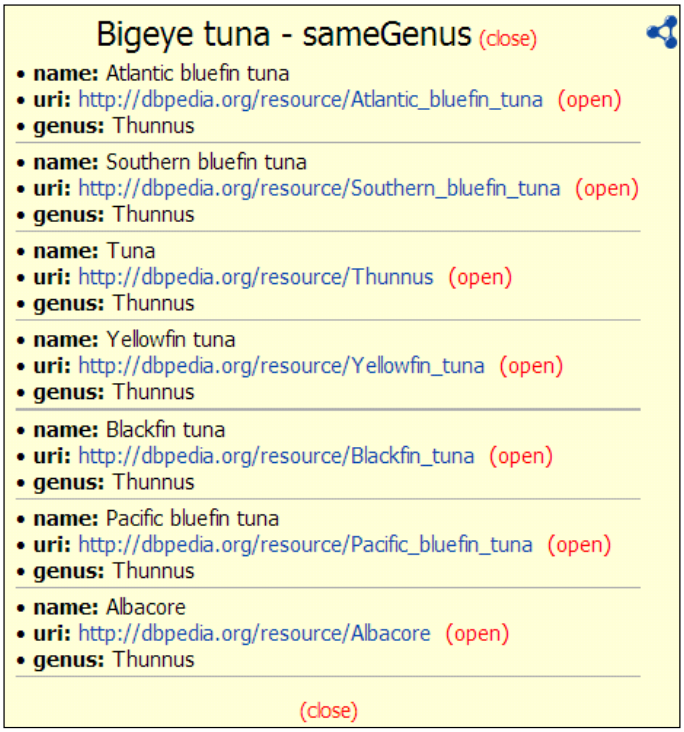


Figure 6.9: An information card displaying fish species that are of the same genus as “Bigeye tuna”.

```
SELECT ?name ?uri ?genus WHERE {
  [URI] <http://dbpedia.org/property/genus> ?genus .
  ?uri <http://dbpedia.org/property/genus> ?genus .
  ?uri <http://dbpedia.org/property/name> ?name }
```

Figure 6.10: SPARQL template query for retrieving fish species of the same genus from DBpedia.

Figure 6.11: Pop-up window showing some of the properties of the fish “Chum salmon”.

Running multiple SPARQL queries (for retrieving the information cards of all the identified entities) at query time would be a very expensive task, especially if the system has discovered a lot of entities. For this reason, one can offer this service *on demand*. Specifically, when the user requests to inspect an information card of an identified entity (e.g., by clicking a hyperlink), the system can, at that time, run the corresponding SPARQL query and display the answer in a popup window. Notice that since the LOD constantly changes and increases, this approach always serves fresh information and also preserves the dynamic nature of the proposed search paradigm.

To allow browsing the LOD related to the identified entities in any depth, we can define a SPARQL query that retrieves, for instance, all the outgoing and incoming properties of a resource. Thereby, the user can semantically explore entities that have not been identified in the search results, i.e., resources related to the identified entities. Figure 6.11 depicts an example of a pop-up window showing some of the outgoing properties of the fish “Chum salmon”.

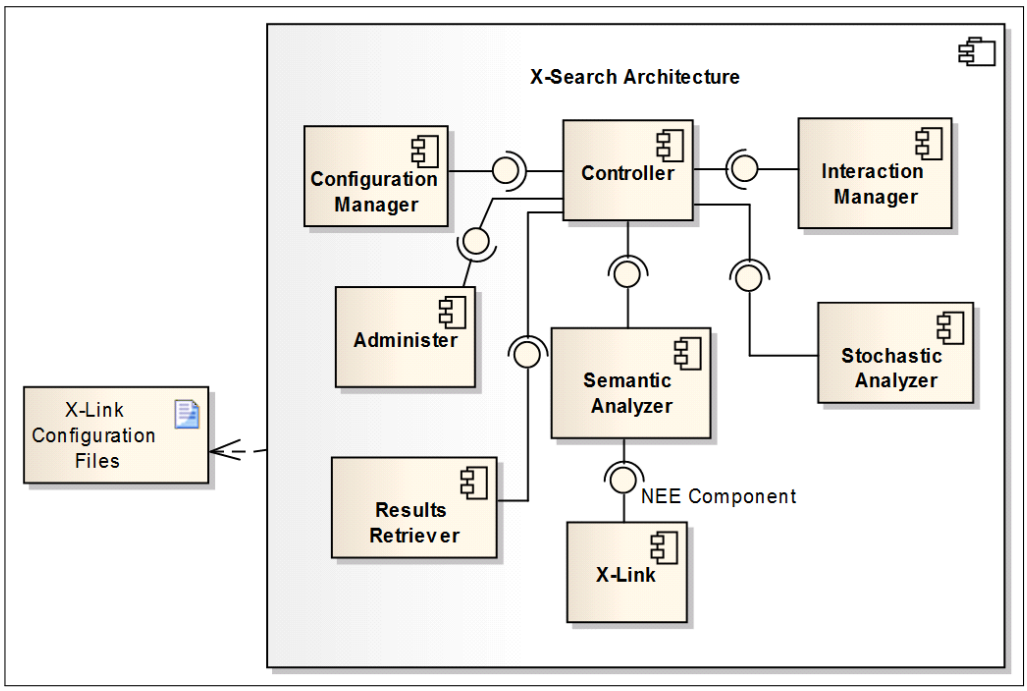


Figure 6.12: The architecture of X-Search.

6.3 Assembling the Pieces: The Search System "X-Search"

`X-Search`[3] is a configurable meta-search engine we have designed and implemented which supports the proposed search paradigm, configurability and interaction model. `X-Search` reads the description of a search source (OpenSearch [8] compliant), queries that source, analyzes the returned results in various ways, and also exploits the availability of semantic repositories.

Figure 6.12 shows the architecture of `X-Search`. The core component is the `Controller` which links and controls all the components. `Results Retriever` is responsible for retrieving the top-L hits from the underlying search system (L is configurable through the UI and has default value 50). `X-Search` semantically analyzes the retrieved hits (either the snippets or the entire contents) at real-time using the `Semantic Analyzer` component. `Semantic Analyzer` exploits the `X-Link` framework both for NEE and for enriching the identified entities with additional (semantic) information. `Stochastic Analyzer` ranks the entities and the related semantic data, while `Interaction Manager` is responsible for the interaction between system and user.

As regards the interaction model, the result of the analysis is visualized and exploited according to the faceted exploration interaction paradigm (as described in 6.1). Specifically, `X-Search` groups the detected entities in facets according to their category. Figure

[3] `http://www.ics.forth.gr/isl/X-Search`

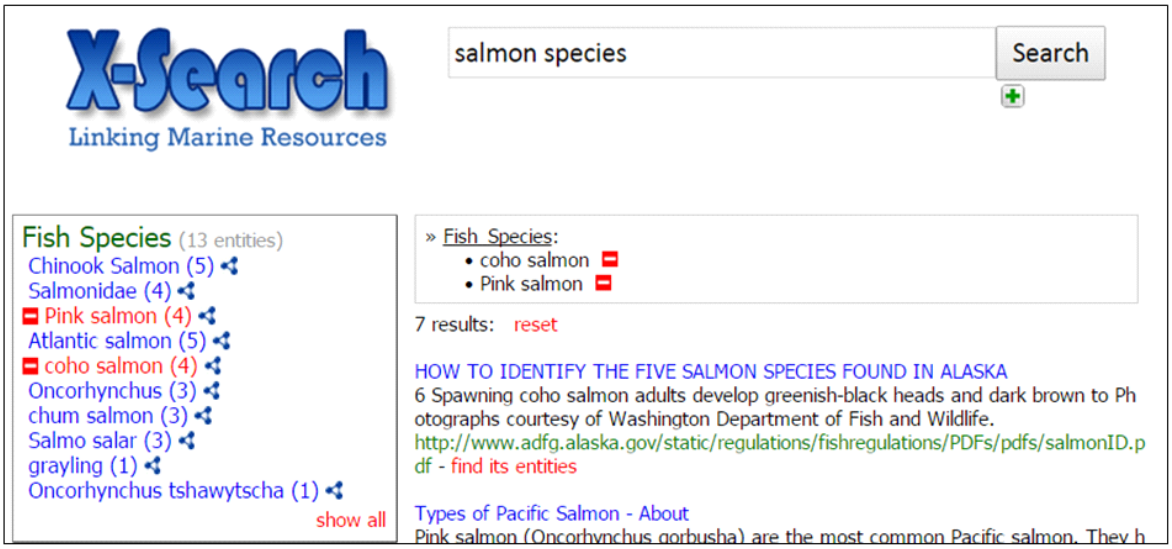


Figure 6.13: Faceted exploration of search results in X-Search.

6.13 depicts an example in which the user has narrowed the search space to 7 results that contain the fish species "pink salmon" and "coho salmon". In addition, `X-Search` provides the necessary linkage between the mined entities and semantic information (as described in 6.2). In particular, the user can request to retrieve semantic resources that match an identified entity (according to the specified Linking Template Queries) by clicking on a small icon next to the entity name. Figure 6.14 depicts an example in which the user has requested to explore the entity "pink salmon". The user can then inspect the properties of a related resource according to the specified Enrichment Template Queries. For example, in Figure 6.15 the user has requested to inspect the properties of a DBpedia resource related to the entity "pink salmon".

By exploiting `X-Link`, `X-Search` is fully configurable in terms of the supported categories of entities, the underlying SKBs and the semantic information that is retrieved for the identified entities. The administrator of `X-Search` can configure all this information through a user-friendly configuration page.

`X-Search` has been deployed and used in the following contexts:

- For Marine Search (described in 6.3.1)
- For Patent Search (described in 6.3.2)

6.3.1 X-Search for Marine Search

In the context of the iMarine[4] project, `X-Search` has been configured to identify Fish Species, Water Areas, Countries, and Regional Fisheries Bodies, while the SKB that is exploited is

[4] EU FP7, 2011-2014, `http://www.i-marine.eu/`

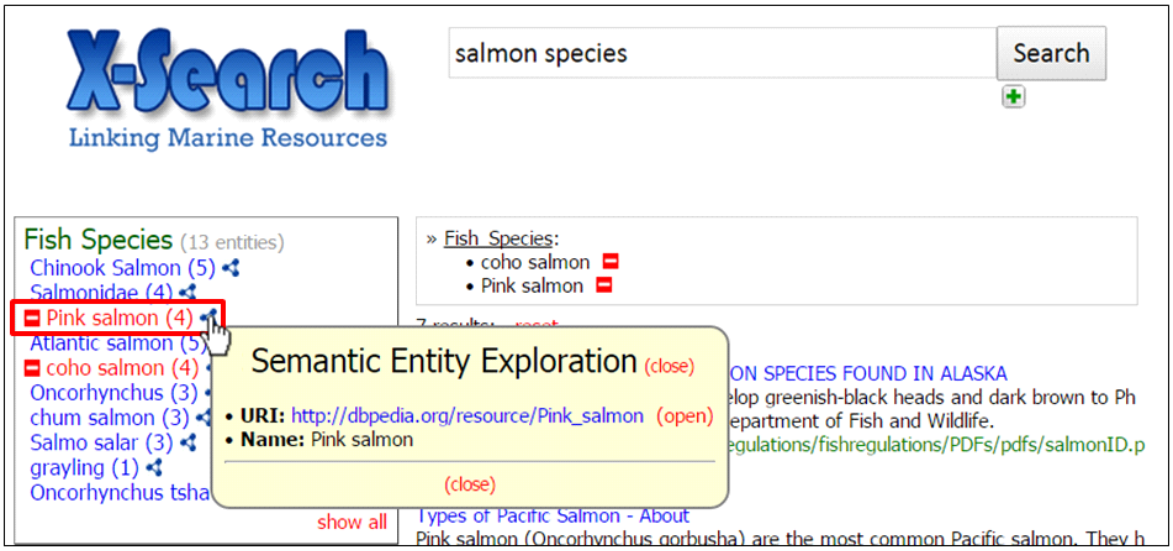


Figure 6.14: Exploring an identified entity in X-Search.

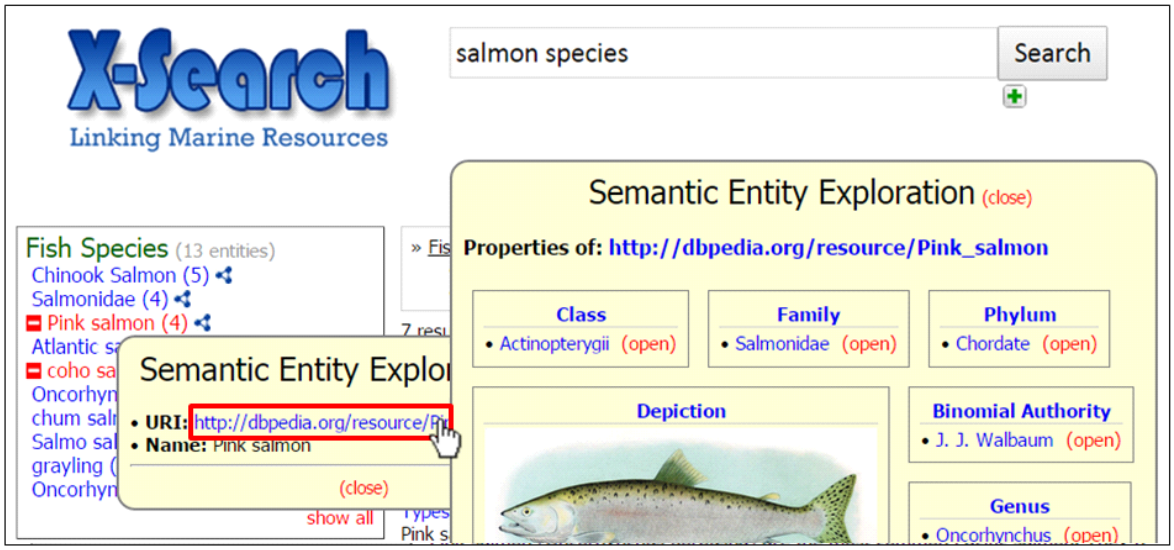


Figure 6.15: Inspecting the properties of a resource in X-Search.

the MarineTLO-based Warehouse [198]. The MarineTLO-based Warehouse integrates information coming from FishBase[5], WoRMS[6], ECOSCOPE[7], FLOD[8] and DBpedia, and currently contains information (more than 5M triples) about marine species (about 54,000), ecosystems, water areas, vessels, etc.

[5] http://www.fishbase.org/
[6] http://www.marinespecies.org/
[7] http://www.ecoscopebc.ird.fr/
[8] http://www.fao.org/figis/flod/

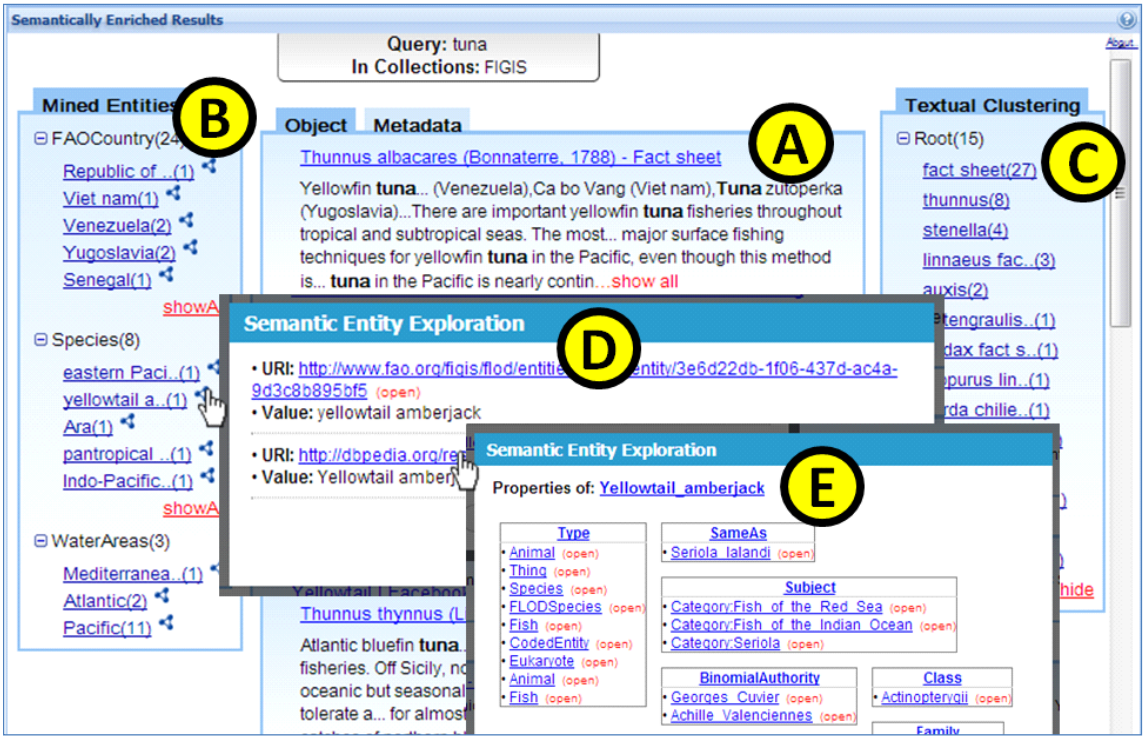


Figure 6.16: X-Search in gCube.

It has been also deployed within an e-infrastructure in which the underlying search system is gCube Search. gCube [45] is a service-oriented application framework that supports the on-demand sharing of resources for computation, content and application services. gCube enables the realization of e-infrastructures that support the notion of Virtual Research Environments (VREs), i.e., collaborative digital environments through which scientists exchange information and produce new knowledge. Figure 6.16 depicts an indicative screen shot of `X-Search` in gCube. We notice that for a particular query, the user can see the top results and the metadata of each result (A), the identified entities grouped in facets (B) as well as the result of textual clustering (C). The user can also inspect semantic resources that match an identified entity (D) and explore their properties (E).

6.3.2 X-Search for Patent Search

In the context of the `PerFedPat`[9] project, `X-Search` has been configured to identify the (medicine-related) categories Diseases, Drugs, Proteins, and Chemical Substances, while the online version of DBpedia is exploited as the underlying SKB. Figure 6.17 depicts an in-

[9]EU FP7, 2011-2014, `http://www.perfedpat.eu/`

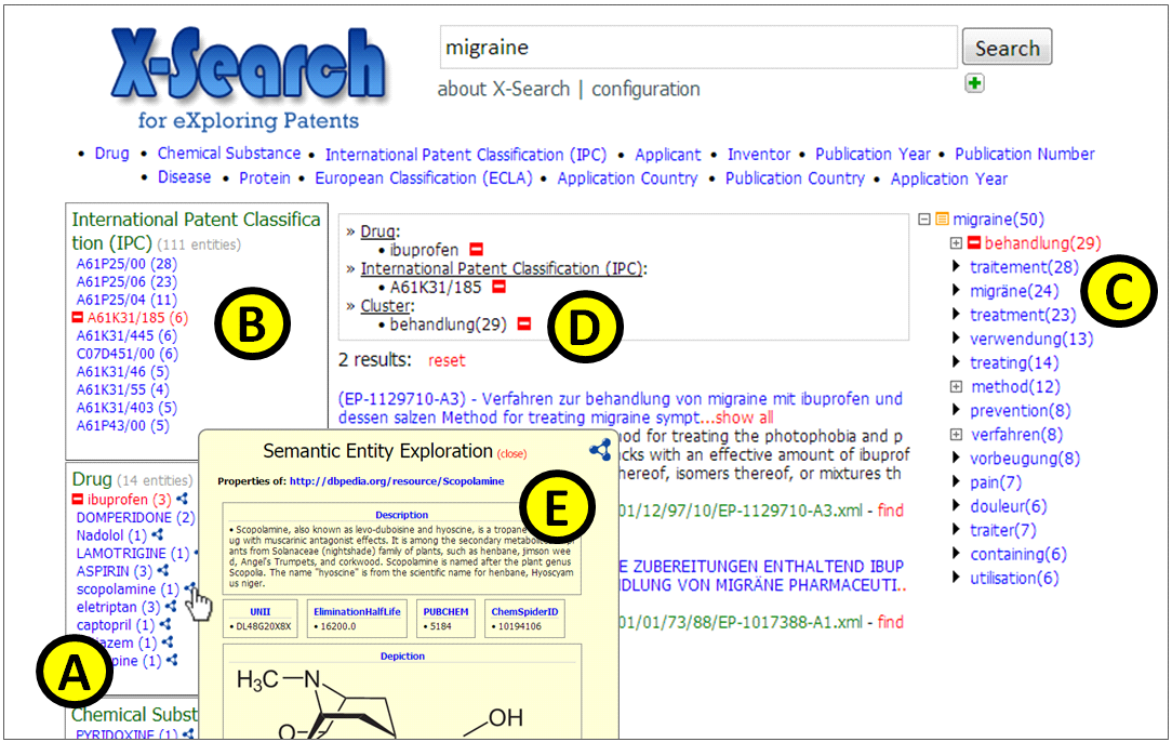


Figure 6.17: X-Search for Patent Search.

dicative screen shot. Note that the user has many options for restricting the search space by selecting one or more identified entities (A), metadata values (B), or clusters (C). For instance, in this example the user has focused on the Drug *ibuprofen*, the International Patent Classification *A61531/185*, and the Cluster *behandlung*, restricting the search space to only 2 results (D). Furthermore, the user is able to retrieve (at real-time) more information about an identified entity (E).

The `PerFedPat` project aims to research into a new generation of advanced patent search systems for the patent related industries and the whole spectrum of patent users. It aims to design a framework for integrating multiple patent data sources, patent search tools and UIs. The `iPerFedPat` system [188], which is the main result of the project, is based on the `ezDL` framework [31] and has a pluggable architecture. It provides core services and operations that allow integrating multiple patent data sources and patent related data streams. It also provides multiple search tools and UIs while hiding complexity from the end-user. Through a REST API, `iPerFedPat` exploits `X-Search` for offering entity-based faceted exploration of the search space as well as entity enrichment capabilities.

Figure 6.18 depicts an indicative screen shot of `iPerFedPat`. We notice that the interface is split in several windows, each corresponding to a different tool. In this example,

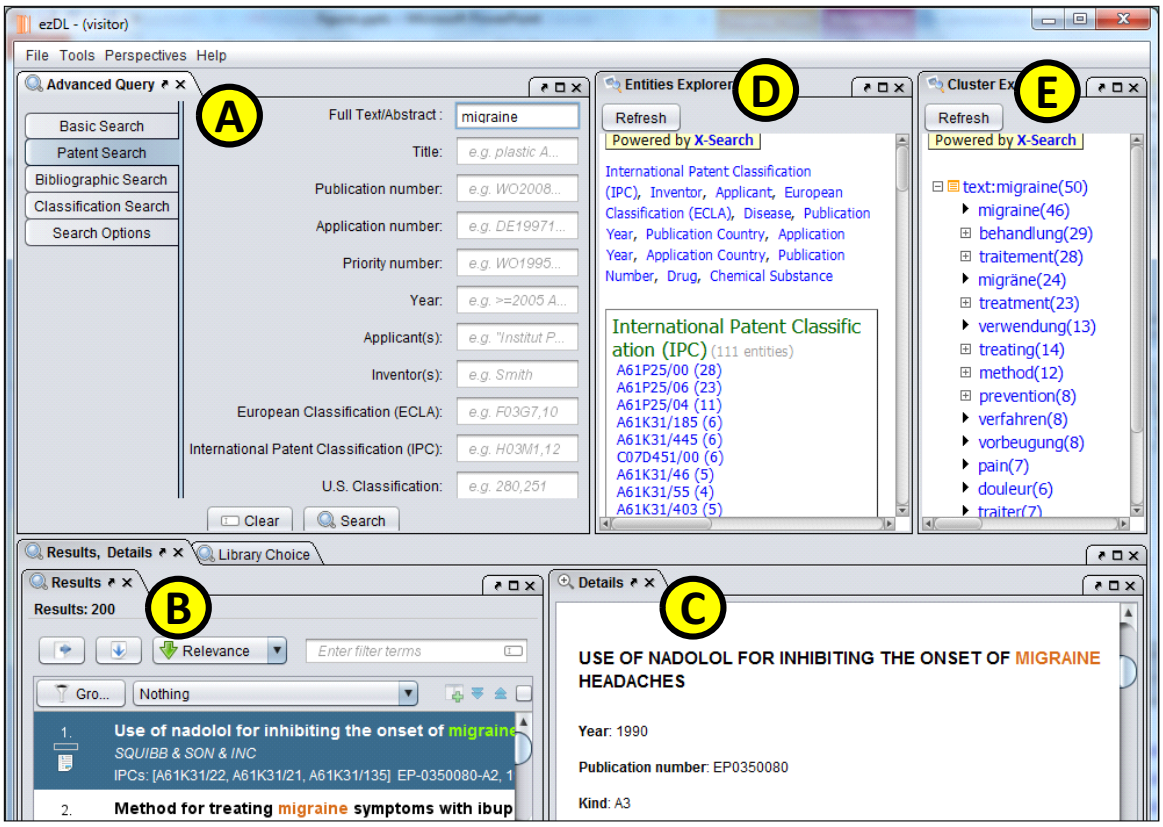


Figure 6.18: The iPerFedPat system.

the user has submitted the query `migraine` using the "Advanced Query" tool (A) and the top results are shown in the "Results" tool (B). Details of a specific result can be displayed using the "Details" tool (C). The user can also inspect the identified entities and the metadata (grouped in categories) (D), as well as a clustering of the search space (E). Thereby, the user can narrow the search space by a selecting one or more entities, metadata values or clusters.

6.4 Theophrastus: Entity-based Automatic Annotation of Web Documents

`Theophrastus`[10] is a system that supports the automatic annotation of Web documents through NEE and provides entity exploration services by exploiting the LOD at real-time. The system was based on requirements coming from the biodiversity domain (in the context of iMarine and Blue Hackathon 2013[11]) and aims at assisting biologists in their re-

[10] `http://www.ics.forth.gr/isl/Theophrastus`
[11] `http://wiki.agroknow.gr/agroknow/index.php/BlueHackathon2013`

search on species and biodiversity. By exploiting X-Link, Theophrastus has been designed to be highly configurable regarding different aspects like EoI, information cards (as described in 6.2), and external (biodiversity-related) search systems. As a result it can be exploited in different contexts and other areas of interest.

Figures 6.19, 6.20, and 6.21 depict three indicative screenshots of Theophrastus. In all examples, the EoI concern fish species (including species genera and family names). Figure 6.19 shows the annotation of the Wikipedia page of the fish "Yellowfin tuna" after clicking over a bookmarklet. A bookmarklet is a bookmark stored in a Web browser that extends the browser's functionality.[12] In Theophrastus, the bookmarklet sends the URL of the Web page the user is viewing to a server. The server then analyzes the contents of the Web page and presents to the user a new (annotated) Web page. The identified species have been annotated with yellow background and the RDF icon has been placed next to each one of them. By clicking on the icon the user can, at real-time, semantically explore the corresponding species. In this example the user explores the species "bigeye tuna". Specifically, the popup window displays: (A) some basic information regarding "bigeye tuna" like its URI and its binomial (scientific name), (B) a set of information cards, e.g., other species that belong to the same family or genus, its taxonomy, etc., and (C) a set of external systems that the user can visit and query for the corresponding species (e.g., the Biodiversity Heritage Library[13]). The user can browse all this information. For instance, by clicking on the URI of an object the user can inspect the properties of the corresponding species that exist in the specified SKB, while by clicking the "sameGenus" information card the user can instantly browse other species that belong to the same genus. Notice that all this information is derived at real-time and can be configured by the user/administrator.

Figure 6.20 shows an analyzed scientific paper (PDF file). The identified species are displayed in a right sidebar and the user can explore a species by simply clicking on it. In this example, we notice that for the detected species "stingray", there are more than one matching species (the species "Pelagic stingray" and "Atlantic stingray").

Finally, in Figure 6.21 a Web page of scientific news has been annotated. In this example, the user inspects an information card regarding the taxonomy of "Chaunax".

The process supported by Theophrastus is sketched in Figure 6.22. Initially, NEE is performed over the contents of a document using X-Link. The contents are then annotated with the identified entities according to the specified EoI. The user can now explore these entities by exploiting the available semantic information through the specified information cards.

[12] https://en.wikipedia.org/wiki/Bookmarklet
[13] http://www.biodiversitylibrary.org/

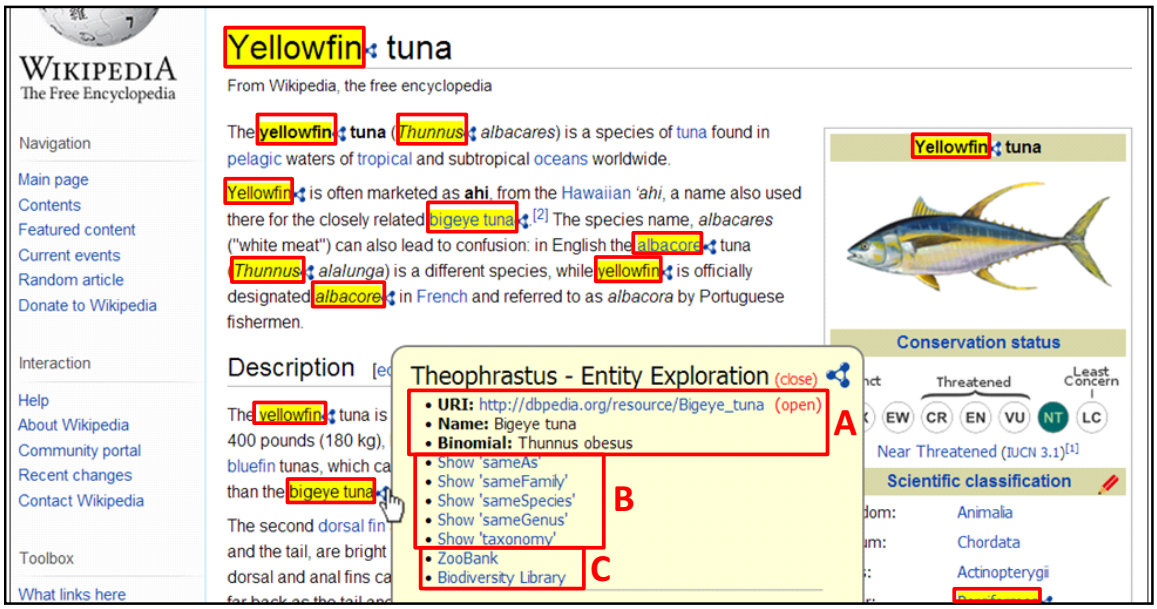


Figure 6.19: Theophrastus over a Wikipedia page.

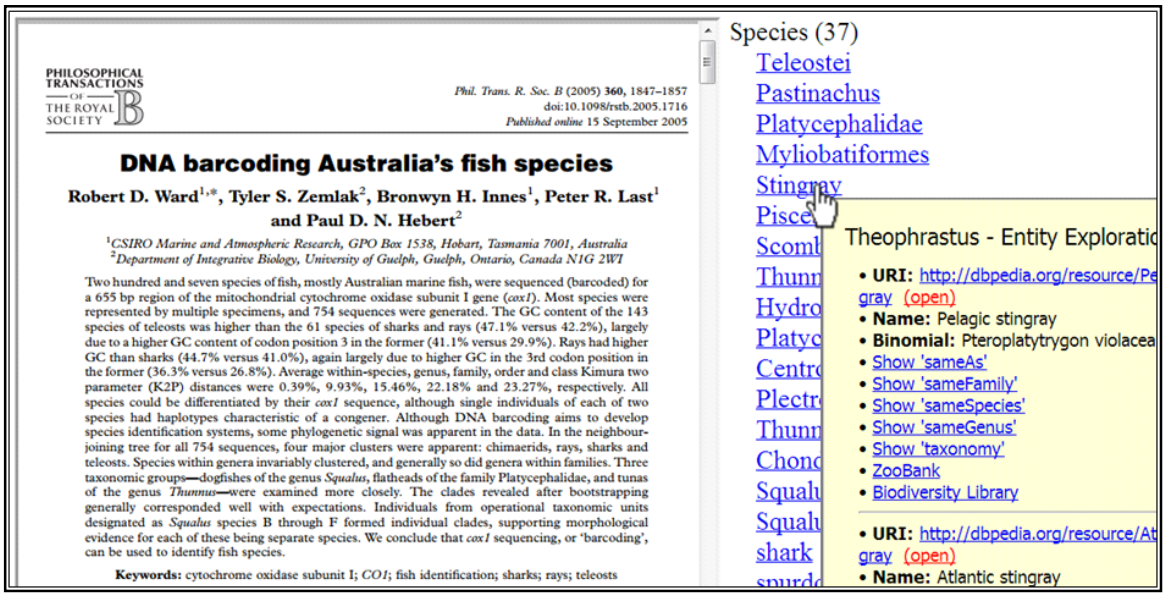


Figure 6.20: Theophrastus over a PDF file.

Figure 6.21: Theophrastus over a news Web page.

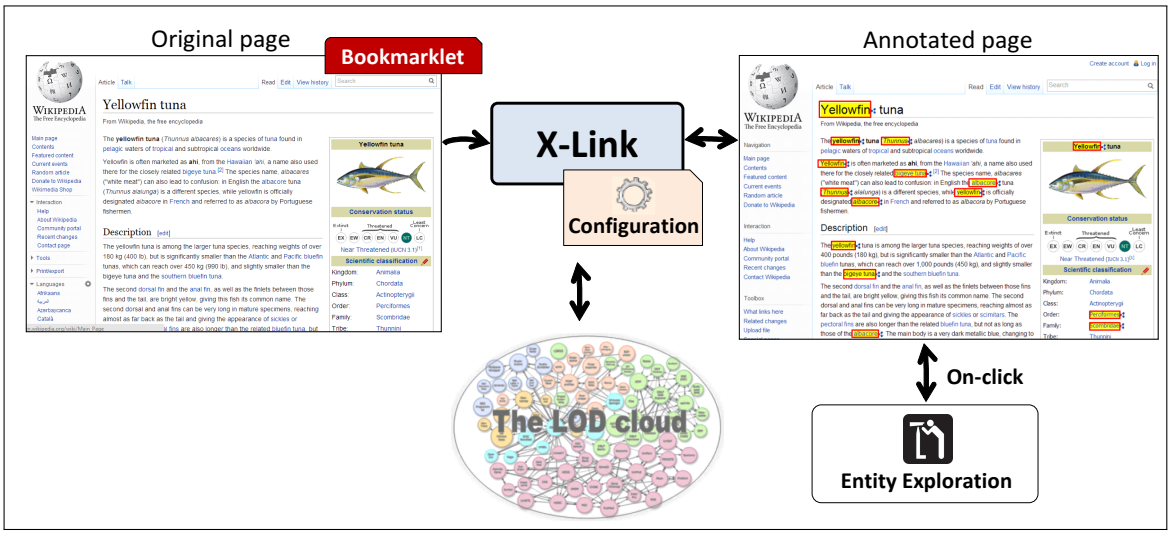


Figure 6.22: The process supported by Theophrastus.

6.5 Weaknesses and Limitations

From our experience, the major issue for providing the "real-time" functionality proposed in this dissertation is the reliability of the online SKBs (like DBpedia). Existing SKBs mainly serve demonstration purposes, are not optimized for efficiency and are not reliable. Actually, *availability* is the main bottleneck towards the success of the Semantic Web as a reliable technology. This is confirmed by the work of Buil-Aranda et al. [44] who tested 427 public endpoints and found that their performance can vary by up to 3-4 orders of magnitude, while only 32.2% of public endpoints can be expected to have monthly uptimes of 99-100%. Nevertheless, in the near future, as such technologies mature and get used in applications, we expect that this problem will be handled.

In the meantime, to increase and reliability and the efficiency of such services, one could *index* the entire or part of the underlying SKBs. Furthermore, as proposed by Umbrich et al. [202], we could keep a local copy of data that hardly changes and offer a hybrid query execution approach for improving the response time and reducing the load on the endpoints, while keeping the results fresh. Of course, in a real application the underlying SKBs may not be publicly available, or a *dedicated Warehouse* can be constructed that will only serve a particular application, like the marineTLO-based warehouse [198] for the marine domain. The SKBs (or the Warehouse) could also be *distributed* in many servers, taking advantage of load balancing techniques [47]. Such approaches can highly improve the performance and the served throughput, however with the cost of loosing the freshness of the results.

Chapter 7
Conclusion

7.1 Synopsis of Contributions

In this thesis, we have focused on integrating search results with Linked Open Data (LOD) for offering advanced exploratory search services and making the LOD directly exploitable by end users. Towards this direction, we have used *named entities* as the "glue" for automatically connecting search results (and documents in general) with LOD.

Since the number of entities identified in a set of retrieved results, as well as the amount of structured information that is available for these entities (their properties and related entities), can be very high, we have proposed a stochastic (Random Walk-based) method for selecting the entities and properties that better characterize the search results and their context, thus providing useful context information. The proposed method promotes the entities identified in the top ranked results as well as the semantic information that is linked with many important (highly-scored) entities. The produced variable-sized semantic graph is exploited in a faceted and session-based interaction scheme that allows users to restrict their focus or information need gradually, while it allows them to instantly inspect information that may exist in different places and that may be laborious and time consuming to locate (avoiding thereby their disengagement from their initial task). In addition, since common IR techniques often do not work well for more complex search tasks, we have examined whether and how such semantic information related to the retrieved results can be exploited for improving the list of results. The aim is to promote low-ranked but relevant hits containing important (for the search context) semantic information.

For evaluating our approach, at first we performed a survey in a professional (marine-related) scenario for examining the usefulness of the derived semantic graph. The results showed that the majority (more than 70%) of the participants a) would like to see a graph representation of the top-5 entities regardless the type of the submitted query, and b) believe that the appearance of a graph of semantic information related to the search results can help them during an exploratory search process. The conducted statistical significance test rejects the randomness of the reported results with a 5% Type-I error. We also reported comparative results which illustrated that the proposed (biased Random Walk-

based) ranking scheme produces more preferred rankings compared to other algorithms (about 22% better ranking compared to a Spreading Activation algorithm and 43% compared to the plain PageRank algorithm).

As regards the proposed model for re-ranking the retrieved results, experimental results over datasets of TREC Clinical Decision Support Track illustrated a notable improvement of the list of results returned by a classical IR system for the majority of the used queries. For instance, re-ranking the top-500 hits using the proposed approach, we can achieve in average about 40% better `bpref` and about 20% better `AveP'` (average precision based on a condensed list) of the top-100 list. This means that the number of relevant hits in the final top-100 list is increased and that existing relevant hits are promoted in higher positions. However, when the initial answer is very good (containing a big number of relevant hits) re-ranking can affect negatively the results by moving relevant hits in lower positions. Nevertheless, the results showed that even in such cases re-ranking can improve recall. Generally, the proposed method seems to perform well on average by increasing mainly recall, but may deteriorate the search results when the initial list is very good. Consequently, such a functionality could be also offered on-demand for cases in which the user is not satisfied by the returned hits. Finally, we saw that additional semantic information about the entities (properties and related entities) can affect negatively the re-ranking process and thus must be carefully considered during the stochastic analysis.

As regards the efficiency of the entire process, we have analyzed experimentally the costs of all steps and we have seen that for up to 100 detected entities (which is the case for snippet-mining), we can offer the proposed functionality at real-time (in less than 4 seconds) even if we access an online SKB like DBpedia. Finally, we showed how we can bound the maximum response time for being scalable even in cases of very big number of identified entities.

We have paid particular attention on the configurability of the proposed approach. Towards this direction, we have proposed a method that exploits the LOD for *configuring* (even dynamically) a Named Entity Extraction (NEE) system. For tackling the configuration requirements we have defined a generic configuration model, while for being able to exchange a supported configuration we have proposed an RDF/S vocabulary, called *Open NEE Configuration Model*. By publishing the configurations supported by one or more NEE services using the proposed model, an application can dynamically (and even at request-time) discover and use the NEE services that best satisfy its annotation needs. In addition, a NEE service that is able to read such configurations can dynamically use a given configuration for annotating a set of documents. To enable relating the output of a NEE process with an applied configuration, we have proposed an extension of the *Open Annotation Data Model*. By accessing the annotation results in this format, an application can offer advanced query services over the annotated data but also integrate semantic information coming from the LOD.

We should stress that, due to the lack of standards related to entity extraction, it would be beneficial for the community if every NEE system supported the configuration model that we propose for making them LOD-aware, and also if every NEE system published the configurations supported by its services using the Open NEE Configuration Model.

Furthermore, we have presented the design and functionality of `X-Link`, a configurable LOD-based NEE framework that supports the proposed configuration model. `X-Link` allows the user/administrator to easily define the categories of entities that are interesting for the application at hand, as well as to update a category and specify how to link and enrich the identified entities, by exploiting one or more Semantic Knowledge Bases (SKBs). This enhanced configurability allows `X-Link` to be used for building and dynamically configuring domain-specific applications (e.g., for identifying *drugs* in a medical search system, for annotating and exploring *fish species* in a marine-related Web page, etc.).

We evaluated `X-Link` in terms of *usability* (ease of configuring it) and *feasibility*. As regards usability, we performed a task-based user study. The results showed that by adopting the proposed approach, one can configure a NEE system within a few minutes. In addition, the majority (80%) of the participants managed to successfully configure the system according to the specified scenario and also found it an easy task. Regarding feasibility, the results of a case study over online DBpedia demonstrated that even if we query a publicly available SKB we can support the exploitation of LOD at real-time.

At the end, we described approaches on how the user can interact with the result of the semantic analysis process for supporting exploratory and entity-based searching, and we presented `X-Search` and `Theophrastus`, two systems we have designed and implemented which support the proposed semantic (and real-time) analysis process, configurability, and interaction model. We also discussed the main weakness of the proposed approach which is related to the reliability of online SKBs, and we saw approaches on how we can cope with this limitation. For example, querying a dedicated semantic Warehouse which applies a load balancing technique can highly improve system's reliability and performance.

We should point out that the proposed approach is *general* (applicable over existing search systems), *configurable* (applicable to many domains), *exploitable in many different contexts* (faceted search, re-ranking, etc.), while the process is *fully automated* (no user effort is required). We believe that the result of this research can provide a general, flexible and adaptive method for enriching search results with structured knowledge in the context of a session-based exploratory search interaction.

7.2 Directions for Future Work and Research

There are several aspects that are worth further work and research.

Stochastic Ranking of Entities, Properties and Search Results

Exploiting the Query Terms

The query terms provided by the user are not exploited in our ranking model although they can provide insights regarding the user task and intention [38, 190]. For example, the query "yellowfin tuna" is probably looking for information about that particular entity, while the query "tuna species" requests a list of entities of a particular type. Thus, an interesting direction for future research is to investigate approaches on how to understand the type of the submitted query and on how to exploit such information both for ranking the entities and the related semantic data and for deciding how to visualize them. In the same context, and since exploratory search often involves long sessions, one can exploit not only the current query but the entire session, i.e., all queries submitted by the user within a time period.

Exploiting Textual Data for Identifying Associations Among the Extracted Entities

It is also interesting to study how textual data related to the identified entities (i.e., text around the entity from the corresponding documents) can be exploited in our model [21, 190]. Such data can provide useful information and reveal hidden associations among the identified entities (e.g., a fish is predator of another fish, a drug is appropriate for curing a specific disease, etc.). This can be especially useful in case the underlying knowledge bases do not contain rich information and thus we do not know whether and how the identified entities are associated to each other. Therefore, it is very interesting to study how such information can be integrated in the proposed Random Walk-based model.

Considering User Actions

Another interesting direction is to study how the user actions in the interactive context of a Faceted Search system can be exploited in our model both for ranking the entities and the related semantic data and for re-ranking the search hits. For example, if the user clicks on an entity, we can consider that this entity is important and thus bias accordingly the transition probabilities. In the same context, an effective interaction model could allow the user to specify which of the displayed semantic data are useful and which not. Such explicit user feedback can help us train our model and improve its effectiveness.

Stochastic Re-Ranking of Search Results

Generality of Re-Ranking Effectiveness

As regards the proposed method for improving the retrieved results, the evaluation results showed that, at least in a medical search context, the proposed re-ranking approach can notably improve the ranking of search results returned by a classic IR system. However, to demonstrate the generality of our model, experiments in also other domains are

needed. It is also very interesting to compare the proposed approach with other methods that exploit entities from Knowledge Bases for improving retrieval effectiveness [72, 213].

Re-Ranking Effect in Perfect and Awful Lists of Results

Another interesting aspect related to the proposed re-ranking method is to study the extreme cases. Specifically, how does our system behave in case the initial list of results is almost perfect (i.e., containing a big number of relevant hits in most of the top positions), and in what extent can our re-ranking method affect such a list? Likewise, how does our system behave in case the initial list is awful (i.e., containing a big number of irrelevant hits in most of the top positions)? In a perfect list, we expect that our method may slightly affect negatively the ranking since the random walker may misplace some low-ranked but irrelevant documents in higher positions. In an awful list, since our method takes into account the scoring of the documents and favors the entities identified in the top-scored documents, we cannot expect improvement of the new list of results. Recall that in an awful list the top retrieved results are not relevant. However, entities identified in these top results are considered important by our algorithm. Our method works well for common cases in which there are some relevant hits in the top scored/ranked documents. Entities in these hits will promote low ranked (and probably relevant) documents (containing these entities) in higher positions.

Understanding the Quality of the Retrieved Results

Another interesting question is whether it is possible to understand the quality of the retrieved results, i.e., to understand if the original list is good in order to decide whether to apply or not the proposed re-ranking process. Log analysis can provide such information (of course in case the submitted query exists in the log). Moreover, implicit signs can reveal that the user is maybe not satisfied with the retrieved results, e.g., when the user is looking many results pages, when she/he runs many queries, etc.

Tuning for Diversity/Recall/Precision

In our work we have not studied the diversity aspect [18]. It is very interesting to investigate how the extracted entities and the related semantic data can be exploited for producing a search result that covers as many relevant topics as possible. We make this remark because one could establish an analogy between our re-ranking model and the mechanisms for pseudo-relevance feedback. An assumption of the latter is that relevance prototypes are "well-behaved", i.e., that either all relevant documents are similar to a single prototype, or that there are different prototypes but they have significant vocabulary overlap [144]. However there are several relevance prototypes. Consequently, a mechanism for pseudo-relevance feedback could reduce the diversity of the search results. It is therefore interesting to investigate how the proposed model of re-ranking affects diversity.

It is interesting also to study how we can tune our algorithm for increasing recall or

precision. The use of facets can be used for recall oriented information needs (e.g., by clicking on an entity we request to inspect all hits containing information about the selected entity), while the proposed re-ranking method can be used to improve both recall and precision. Thus, an interesting aspect is to study which parameters of our ranking model affect recall and, correspondingly, which parameters affect precision.

Personalized Exploratory Search

User Profiles

It is also very interesting to study *personalization* methods. A straightforward approach is to allow users set up and maintain their own configuration(s), e.g., through user profiles [101]. Then, each user could select the configuration to be used for analyzing the list of retrieved results.

User Preferences

Furthermore, in our modeling we consider all categories of entities and all SKBs of the same weight/importance. However, we could promote entities of one or more specific categories (RDF classes), or entities having some specific properties/characteristics, or those coming from specific SKBs. For example, in a marine-related scenario, one could define that the category Fish is more important that the category Country, that fishes of genus "thunnus" are preferable, and so on. This requires extending our model to allow describing such preferences. The preferences can be also expressed at real-time, e.g., the actions of Preference-enriched Faceted Search (PFS) [201] could be exploited for ranking both the entities and the search hits.

Collaborative Exploratory Search

Implicity Collaboration

Another possibility for future work is to study how log analysis data (e.g., clicks on entities for a specific query) can be exploited in our model for inferring similar information needs among users (i.e., for implicit collaboration). Such information can be exploited both for ranking the entities and for re-ranking the search hits.

Re-usable Semantic Graphs

It is interesting also to study how the derived semantic graphs can be re-used across users, e.g., for supporting collaborative search and exploration [155]. This can enable users in a community combine their efforts in several types of IR activities by sharing information resources (semantic graphs in our case) related to search tasks.

Interacting with Top-K Semantic Graphs

Another interesting direction for future research is to study approaches on how exactly the derived top-K semantic graphs should affect the interaction between user and system, as well as on how such graphs can be effectively visualized (especially for big values of K) [107]. After that, it could be interesting to carry out a task-based evaluation on how such semantic (and interconnected) information related to a set of retrieved results affects the user experience and the effectiveness of several types of search tasks.

Entity-based Stochastic Analysis for Query Expansion/Recommendation

It is interesting also to investigate exploiting the result of the proposed (entity-based) stochastic analysis process in other contexts, e.g., for query expansion [72] or recommendation [38], aiming to create and suggest a query that can retrieve more relevant hits.

Indexing and Caching for Efficient and Reliable Stochastic Analysis Process

We saw that the major issue for providing the proposed functionality is the efficiency and reliability of online SKBs. In addition, we have seen that performing entity mining can be time consuming in case we want to analyze a very big number of results and also in case the results have big size. However, offering low response times and being reliable is crucial in IR and especially in general Web Searching. Thereby, it is interesting to study approaches on:

- How to index the underlying corpus for fast entity-based IR (of course in case the corpus is available) [57, 168].
- How to index part or the entire SKBs in order to efficiently find URIs that match an entity name as well as to efficiently retrieve the RDF triples of a specific entity URI. How to update the index, especially in case the LOD changes frequently.
- How to cache semantic graphs, e.g., for the more frequent queries (i.e., those used for autocompletion) [90].
- How to improve the performance and the served throughput, e.g., through a distributed computational model [131].

Named Entity Extraction and Configurability

Systems Interoperability

Regarding the configuration model, one direction is to extend it for modeling also non-functional aspects of a NEE service, like the average annotation time, the average linking time, etc. Our vision is to offer a model that can be used to wholly describe the functionality, the API (i.e., how to use it), and the configurability of a NEE service. This would allow

a client application to dynamically discover and use NEE services by exploiting only standard Web protocols, without needing to set up and maintain a corresponding service.

Another related interesting direction is to study approaches on how to automatically select (at request-time) the NEE services to use from a list of available descriptions (expressed using the proposed Open NEE Configuration Model) according, for example, to the submitted query or the context of the retrieved results.

Facilitating the Construction of SPARQL Queries and Template Queries

We also saw that for specifying the EoI and the related semantic data, knowledge of SPARQL is required. Thus, another possible direction for future work is to study approaches that facilitate the construction of SPARQL queries (for specifying the EoI) and of SPARQL template queries (for specifying how to link and enrich the identified entities) [22, 184]. This will allow the configuration of the NEE process even by common users that have no computer science background.

Cleaning the Retrieved Semantic Data

Finally, another common problem of existing SKBs that can highly affect the effectiveness of our methods is their quality in terms of completeness, connectivity, etc. Thus, an interesting direction for future research is to study automated methods for cleaning the retrieved semantic data as well as for assessing their quality and connectivity [156, 200].

Bibliography

[1] AlchemyAPI. http://www.alchemyapi.com/.

[2] DrugBank. http://www.drugbank.ca/.

[3] EntityCube. http://entitycube.research.microsoft.com/.

[4] FAO Fisheries Linked Open Data. http://www.fao.org/figis/flod/.

[5] Google Knowledge Graph. https://en.wikipedia.org/wiki/Knowledge_Graph.

[6] Lupedia Enrichment Service, Ontotext. http://lupedia.ontotext.com/.

[7] OpenCalais, Thomson Reuters. http://www.opencalais.com/.

[8] OpenSearch. http://www.opensearch.org/.

[9] SemSearch Challenge. https://km.aifb.kit.edu/ws/semsearch11/.

[10] Simple Knowledge Organization System (SKOS). http://www.w3.org/2004/02/skos/.

[11] SPARQL 1.1 Federated Query, W3C Recommendation, 21 March 2013. http://www.w3.org/TR/sparql11-federated-query/.

[12] SPARQL 1.1 Protocol (W3C Recommendation). http://www.w3.org/TR/sparql11-protocol/.

[13] Sparql 1.1 query language (w3c recommendation). http://www.w3.org/TR/sparql11-query/.

[14] W3C Provenance Data Model (PROV). http://www.w3.org/TR/prov-overview/.

[15] Wikimeta. http://www.wikimeta.com/.

[16] Yahoo! Content Analysis API. http://developer.yahoo.com/contentanalysis/.

[17] A. Agrawal, S. Sudarshan, A. Sahoo, A. Sandalwala, and P. Jaiswal. Entity Ranking and Relationship Queries Using an Extended Graph Model. In *18th International Conference on Management of Data*, 2012.

[18] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

[19] James Allan, Bruce Croft, Alistair Moffat, and Mark Sanderson. Frontiers, challenges, and opportunities for information retrieval: Report from SWIRL 2012 the second strategic workshop on information retrieval in Lorne. In *ACM SIGIR Forum*, volume 46, pages 2–32. ACM, 2012.

[20] Carlo Allocca, Mathieu d"Aquin, and Enrico Motta. Impact of using relationships between ontologies to enhance the ontology search results. In *The Semantic Web: Research and Applications*, pages 453–468. Springer, 2012.

[21] Mazen Alsarem, Pierre-Edouard Portier, Sylvie Calabretto, and Harald Kosch. Ranking Entities in the Age of Two Webs, an Application to Semantic Snippets. In *The Semantic Web. Latest Advances and New Domains*. 2015.

[22] Oszkar Ambrus, Knud Moller, and Siegfried Handschuh. Konduit VQB: a Visual Query Builder for SPARQL on the Social Semantic Desktop. In *Workshop on Visual Interfaces to the Social and Semantic Web*, 2010.

[23] Teresa K Attwood, Douglas B Kell, Philip McDermott, James Marsh, SR Pettifer, and David Thorne. Utopia documents: linking scholarly literature with research data. *Bioinformatics*, 26(18):i568–i574, 2010.

[24] Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 619–626. ACM, 2001.

[25] Atanaz Babashzadeh, Jimmy Huang, and Mariam Daoud. Exploiting semantics for improving clinical information retrieval. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 801–804. ACM, 2013.

[26] Ricardo Baeza-Yates and Emilio Davis. Web page ranking using link attributes. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 328–329. ACM, 2004.

[27] Krisztian Balog, Arjen P de Vries, Pavel Serdyukov, Paul Thomas, and Thijs Westerveld. Overview of the TREC 2009 Entity Track. Technical report, 2009.

[28] B. Bamba and S. Mukherjea. Utilizing resource importance for ranking semantic web query results. *Semantic Web and Databases*, 2005.

[29] Manolis Baritakis. Using Linked Data for Named Entity Extraction and Disambiguation. Master's thesis, University Of Crete, Computer Science Department, Greece, 2016.

[30] H. Bast, A. Chitea, F. Suchanek, and I. Weber. ESTER: efficient search on text, entities, and relations. In *30th international ACM SIGIR conference on research and development in information retrieval*, 2007.

[31] Thomas Beckers, Sebastian Dungs, Norbert Fuhr, Matthias Jordan, Sascha Kriewel, and Vu Tuan Tran. ezdl: An interactive search and evaluation system. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 9–16, 2012.

[32] Dave Beckett and Brian McBride. RDF/XML syntax specification (revised). *W3C recommendation*, 10, 2004.

[33] Krishna Bharat and Monika R Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 104–111. ACM, 1998.

[34] Matthew W Bilotti, Boris Katz, and Jimmy Lin. What works better for question answering: Stemming or morphological query expansion. In *Information Retrieval for Question Answering, SIGIR Workshop*, 2004.

[35] Barry Bishop, Atanas Kiryakov, Damyan Ognyanov, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. Factforge: A fast track to the web of data. *Semantic Web*, 2(2):157–166, 2011.

[36] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3), 2009.

[37] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the web (LDOW2008). In *Proceedings of the 17th international conference on World Wide Web*, pages 1265–1266. ACM, 2008.

[38] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity Recommendations in Web Search. In *The Semantic Web–ISWC 2013*, pages 33–48. Springer, 2013.

[39] D. Bonino, A. Ciaramella, and F. Corno. Review of the state-of-the-art in patent information and forthcoming evolutions in intelligent patent informatics. *World Patent Information*, 32(1), 2010.

[40] Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3-4):349–373, 2004.

[41] Allan Borodin, Gareth O Roberts, Jeffrey S Rosenthal, and Panayiotis Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Transactions on Internet Technology (TOIT)*, 5(1):231–297, 2005.

[42] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1), 1998.

[43] Chris Buckley and Ellen M Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM, 2004.

[44] Carlos Buil-Aranda, Aidan Hogan, Jurgen Umbrich, and Pierre-Yves Vandenbussche. Sparql web-querying infrastructure: Ready for action? In *ISWC 2013*. Springer, 2013.

[45] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. gCube: a service-oriented application framework on the grid. *ERCIM News*, 72:48–49, 2008.

[46] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 243–250. ACM, 2008.

[47] Valeria Cardellini, Michele Colajanni, and Philip S Yu. Dynamic Load Balancing on Web-Server Systems. *Internet Computing, IEEE*, 3(3):28–39, 1999.

[48] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June (Paul) Hsu, and Kuansan Wang. ERD'14: Entity Recognition and Disambiguation Challenge. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 1292–1292. ACM, 2014.

[49] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.

[50] Arthur Carvalho and Kate Larson. A consensual linear opinion pool. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2518–2524. AAAI Press, 2013.

[51] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic resource compilation by analyzing hyperlink

structure and associated text. *Computer Networks and ISDN Systems*, 30(1):65–74, 1998.

[52] Youjin Chang, Iadh Ounis, and Minkoo Kim. Query reformulation using automatically generated query concepts from a document space. *Information processing & management*, 42(2):453–468, 2006.

[53] Eric Charton, Michel Gagnon, and Benoit Ozell. Automatic Semantic Web Annotation of Named Entities. In *Advances in Artificial Intelligence*, pages 74–85. Springer, 2011.

[54] Rashmi Chauhan, Rayan Goudar, Ritu Sharma, and Anamika Chauhan. Domain ontology based semantic search for efficient information retrieval through automatic query expansion. In *International Conference on Intelligent Systems and Signal Processing*. IEEE, 2013.

[55] Gong Cheng, Weiyi Ge, and Yuzhong Qu. Falcons: searching and browsing entities on the semantic web. In *Proceedings of the 17th international conference on World Wide Web*, pages 1101–1102. ACM, 2008.

[56] T. Cheng and K.C.C. Chang. Entity search engine: Towards agile best-effort information integration over the Web. In *3rd Biennial Conference on Innovative Data Systems Research*, 2007.

[57] Tao Cheng and K.C.C. Chang. Beyond pages: supporting efficient, scalable entity search with dual-inversion index. In *13th International Conference on Extending Database Technology*, 2010.

[58] Tao Cheng, Xifeng Yan, and K.C.C. Chang. EntityRank: searching entities directly and holistically. In *33rd international conference on Very large data bases (VLDB)*, 2007.

[59] Boris Chidlovskii, Natalie S Glance, and M Antonietta Grasso. Collaborative reranking of search results. In *Proc. AAAI-2000 Workshop on AI for Web Search*, 2000.

[60] Sungbin Choi, Jinwook Choi, Sooyoung Yoo, Heechun Kim, and Youngho Lee. Semantic concept-enriched dependence model for medical information retrieval. *Journal of biomedical informatics*, 47:18–27, 2014.

[61] Paolo Ciccarese, Marco Ocana, Leyla Jael Garcia Castro, Sudeshna Das, and Tim Clark. An open annotation ontology for science on web 3.0. *Journal of Biomedical Semantics*, 2, 2011.

[62] Paolo Ciccarese, Marco Ocana, and Tim Clark. Open semantic annotation of scientific publications using DOMEO. *Journal of biomedical semantics*, 3:1–14, 2012.

[63] M. Ciglan, K. Nrvg, and L. Hluchý. The SemSets model for ad-hoc semantic list search. In *21st international conference on World Wide Web*, 2012.

[64] Paul R Cohen and Rick Kjeldsen. Information retrieval by constrained spreading activation in semantic networks. *Information processing & management*, 23(4):255–268, 1987.

[65] F. Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6), 1997.

[66] F. Crestani and P.L. Lee. Searching the Web by constrained spreading activation. *Information Processing & Management*, 36(4), 2000.

[67] Silviu Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL*, volume 7, pages 708–716. Citeseer, 2007.

[68] Jingyu Cui, Fang Wen, and Xiaoou Tang. Real time google and live image search re-ranking. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 729–732. ACM, 2008.

[69] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[70] Celia da Costa Pereira, Mauro Dragoni, and Gabriella Pasi. Multidimensional relevance: Prioritized aggregation in a personalized Information Retrieval setting. *Information processing & management*, 48(2):340–357, 2012.

[71] L. Dali, B. Fortuna, T. Duc, and D. Mladenic. Query-Independent Learning to Rank for RDF Entity Search. In *9th Extended Semantic Web Conference*, 2012.

[72] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 365–374. ACM, 2014.

[73] Mathieu d'Aquin and Enrico Motta. Watson, more than a semantic web search engine. *Semantic Web*, 2(1):55–63, 2011.

[74] Jan De Beer and Marie-Francine Moens. Rpref: a generalization of Bpref towards graded relevance judgments. In *Proceedings of the 29th annual international ACM*

SIGIR conference on Research and development in information retrieval, pages 637–638. ACM, 2006.

[75] R. Delbru, N. Toupikov, M. Catasta, G. Tummarello, and S. Decker. Hierarchical link analysis for ranking Web data. In *7th Extended Semantic Web Conference*, 2010.

[76] Gianluca Demartini, Tereza Iofciu, and Arjen P De Vries. Overview of the INEX 2009 Entity Ranking Track. In *Focused Retrieval and Evaluation*, pages 254–264. Springer, 2009.

[77] Persi Diaconis and Ronald L Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 262–268, 1977.

[78] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and Ranking Knowledge on the Semantic Web. In *4th International Semantic Web Conference*, 2005.

[79] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the Semantic Web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659. ACM, 2004.

[80] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.

[81] Marco Trevisan Enrico Franconi, Paolo Guagliardo. Quelo: a NL-based Intelligent Query Interface. In *2nd Workshop on Controlled Natural Languages (CNL 2010)*, 2010.

[82] P. Fafalios, M. Baritakis, and Y. Tzitzikas. Exploiting linked data for open and configurable named entity extraction. *International Journal on Artificial Intelligence Tools*, 24(02), 2015.

[83] P. Fafalios, I. Kitsos, Y. Marketakis, C. Baldassarre, M. Salampasis, and Y. Tzitzikas. Web Searching with Entity Mining at Query Time. In *5th Information Retrieval Facility Conference*, 2012.

[84] P. Fafalios, I. Kitsos, and Y. Tzitzikas. Scalable, flexible and generic instant overview search. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 333–336. ACM, 2012.

[85] P. Fafalios, P. Papadakos, and Y. Tzitzikas. Enriching Textual Search Results at Query Time using Entity Mining, Linked Data and Link Analysis. *International Journal of Semantic Computing*, 08(04):515–544, 2014.

[86] P. Fafalios, M. Salampasis, and Y. Tzitzikas. Exploratory Patent Search with Faceted Search and Configurable Entity Mining. In *1st International Workshop on Integrating IR technologies for Professional Search (ECIR'13 Workshop)*, 2013.

[87] P. Fafalios and Y. Tzitzikas. Exploiting available memory and disk for scalable instant overview search. *Web Information System Engineering–WISE 2011*, pages 101–115, 2011.

[88] P. Fafalios and Y. Tzitzikas. X-ENS: Semantic Enrichment of Web Search Results at Real-Time. In *36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2013.

[89] P. Fafalios and Y. Tzitzikas. Post-Analysis of Keyword-based Search Results using Entity Mining, Linked Data and Link Analysis at Query Time. In *2014 IEEE Eighth International Conference on Semantic Computing (ICSC 2014)*, Newport Beach, California, USA, June 16-18 2014. IEEE.

[90] P. Fafalios and Y. Tzitzikas. Type-Ahead Exploratory Search through Typo and Word Order Tolerant Autocompletion. *Journal of Web Engineering*, 14(1-2):80–116, 2015.

[91] Pavlos Fafalios and Panagiotis Papadakos. Theophrastus: On Demand and Real-Time Automatic Annotation and Exploration of (Web) Documents using Open Linked Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2014.

[92] Ronald Fagin, Ravi Kumar, and D Sivakumar. Comparing Top-K Lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003.

[93] Laura Faulkner. Beyond the Five-User Assumption: Benefits of Increased Sample Sizes in Usability Testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, 2003.

[94] Miriam Fernandez, Ivan Cantador, Vanesa Lopez, David Vallet, Pablo Castells, and Enrico Motta. Semantically enhanced Information Retrieval: an ontology-based approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):434–452, 2011.

[95] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with Wikipedia pages. *arXiv preprint arXiv:1006.3498*, 2010.

[96] Sebastien Ferre and Alice Hermann. Semantic Search: Reconciling expressive querying and exploratory search. In *The Semantic Web–ISWC 2011*, pages 177–192. Springer, 2011.

[97] Larry Fitzpatrick and Mei Dent. Automatic feedback using past queries: social searching? In *ACM SIGIR Forum*, volume 31, pages 306–313. ACM, 1997.

[98] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[99] Michel Gagnon, Amal Zouaq, and Ludovic Jean-Louis. Can We Use Linked Data Semantic Annotators for the Extraction of Domain-Relevant Expressions? In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 1239–1246. International World Wide Web Conferences Steering Committee, 2013.

[100] Eugene Garfield et al. Citation analysis as a tool in journal evaluation. American Association for the Advancement of Science, 1972.

[101] Susan Gauch, Jason Chaffee, and Alexander Pretschner. Ontology-based personalized search and browsing. *Web Intelligence and Agent Systems: An international Journal*, 1(3, 4):219–234, 2003.

[102] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Inferring web communities from link topology. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space*, pages 225–234. ACM, 1998.

[103] BC Grau, E Kharlamov, S Marciuska, D Zheleznyakov, M Arenas, and E Jimenez-Ruiz. SemFacet: Semantic Faceted Search over Yago. *WWW Demo*, 2014.

[104] Parth Gupta, Kalika Bali, Rafael E Banchs, Monojit Choudhury, and Paolo Rosso. Query expansion for mixed-script information retrieval. In *37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014.

[105] Sairam Gurajada, Jaap Kamps, Arunav Mishra, Ralf Schenkel, Martin Theobald, and Qiuyue Wang. Overview of the INEX 2013 Linked Data Track. In *In CLEF 2013 Evaluation Labs and Workshop, Online Working Notes*. Citeseer, 2013.

[106] Kevin Haas, Peter Mika, Paul Tarjan, and Roi Blanco. Enhanced results for web search. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 725–734. ACM, 2011.

[107] Steffen Hadlak, Heidrun Schumann, and Hans-Jorg Schulz. A survey of multi-faceted graph visualization. In *Eurographics Conference on Visualization (EuroVis)*, pages 1–20, 2015.

[108] Harry Halpin, Daniel M Herzig, Peter Mika, Roi Blanco, Jeffrey Pound, Henry S Thompson, and Duc Thanh Tran. Evaluating ad-hoc object retrieval. In *Proc. of the Intl. Workshop on Evaluation of Semantic Technologies*, 2010.

[109] Xianpei Han and Jun Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 215–224. ACM, 2009.

[110] A. Harth, S. Kinsella, and S. Decker. Using Naming Authority to Rank Data and Ontologies for Web Search. In *8th International Semantic Web Conference*, 2009.

[111] Joseph Hassell, Boanerges Aleman-Meza, and I Budak Arpinar. *Ontology-driven automatic entity disambiguation in unstructured text*. Springer, 2006.

[112] Taher H Haveliwala. Topic-sensitive PageRank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM, 2002.

[113] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[114] Ivan Herman, Guy Melancon, and M Scott Marshall. Graph visualization and navigation in information visualization: A survey. *Visualization and Computer Graphics, IEEE Transactions on*, 6(1):24–43, 2000.

[115] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232. ACM, 2011.

[116] A. Hogan, A. Harth, and S. Decker. ReConRank: A Scalable Ranking Method for Semantic Web Data with Context. In *2nd International Workshop on Scalable Semantic Web Knowledge Base Systems*, 2006.

[117] Aidan Hogan, Andreas Harth, Jurgen Umbrich, Sheila Kinsella, Axel Polleres, and Stefan Decker. Searching and browsing linked data with SWSE: the semantic Web search engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):365–401, 2011.

[118] Vidit Jain and Manik Varma. Learning to re-rank: query-dependent image re-ranking using click data. In *Proceedings of the 20th international conference on World wide web*, pages 277–286. ACM, 2011.

[119] Kalervo Jarvelin and Jaana Kekalainen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[120] Glen Jeh and Jennifer Widom. Scaling personalized Web Search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM, 2003.

[121] Vikas Jindal, Seema Bawa, and Shalini Batra. A review of ranking approaches for semantic search on Web. *Information Processing & Management*, 50(2):416–425, 2014.

[122] H. Joho, L.A. Azzopardi, and W. Vanderbauwhede. A survey of patent users: an analysis of tasks, behavior, search functionality and system requirements. In *Procs of the 3rd symposium on Information interaction in context*. ACM, 2010.

[123] Clement Jonquet, Nigam Shah, Cherie Youn, Chris Callendar, Margaret-Anne Storey, and M Musen. NCBO annotator: semantic annotation of biomedical data. In *International Semantic Web Conference*, 2009.

[124] Guyonvarch Joris and Sebastien Ferre. Scalewelis: a Scalable Query-based Faceted Search System on Top of SPARQL Endpoints. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*, 2013.

[125] Jason J Jung. Cross-lingual query expansion in multilingual folksonomies: A case study on Flickr. *Knowledge-Based Systems*, 42:60–67, 2013.

[126] M. Kaki. Findex: search result categories help users when document ranking fails. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2005.

[127] M. Kaki and A. Aula. Findex: improving search result use through automatic filtering categories. *Interacting with Computers*, 17(2):187–206, 2005.

[128] Nattiya Kanhabua and Kjetil Nrvg. Determining time of queries for re-ranking search results. In *Research and Advanced Technology for Digital Libraries*, pages 261–272. Springer, 2010.

[129] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and ranking knowledge. In *IEEE 24th International Conference on Data Engineering*, 2008.

[130] M. G. Kendall. Rank correlation methods. Griffin, London, 1970.

[131] Ioannis Kitsos, Kostas Magoutis, and Yannis Tzitzikas. Scalable entity-based summarization of web search results using MapReduce. *Distributed and Parallel Databases*, 32(3):405–446, 2014.

[132] J. Kleb and A. Abecker. Entity reference resolution via spreading activation on rdf-graphs. In *7th Extended Semantic Web Conference*, 2010.

[133] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 1999.

[134] Alex Kohn, Francois Bry, Alexander Manta, and Dirk Ifenthaler. *Professional Search: Requirements, Prototype and Preliminary Experience Report*, pages 195–202. 2008.

[135] Reiner Kraft and Jason Zien. Mining anchor text for query refinement. In *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004.

[136] B. Kules, R. Capra, M. Banta, and T. Sierra. What do exploratory searchers look at in a faceted search interface? In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 313–322. ACM, 2009.

[137] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective Annotation of Wikipedia Entities in Web Text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009.

[138] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001.

[139] Kyung Soon Lee, W Bruce Croft, and James Allan. A cluster-based resampling method for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 235–242. ACM, 2008.

[140] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Soren Auer, et al. DBpedia–A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 2014.

[141] Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*, 33(1):387–401, 2000.

[142] X. Li, C. Li, and C. Yu. Entity-relationship queries over Wikipedia. In *2nd international workshop on Search and mining user-generated contents*, 2010.

[143] X. Li, C. Li, and C. Yu. EntityEngine: answering entity-relationship queries using shallow semantics. In *19th ACM international conference on Information and knowledge management*, 2010.

[144] Christopher D Manning, Prabhakar Raghavan, Hinrich Schutze, et al. *Introduction to Information Retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[145] G. Marchionini. Exploratory Search: from finding to understanding. *Communications of the ACM*, 2006.

[146] Alberto O. Mendelzon and Davood Rafiei. What do the neighbours think? Computing Web page reputations. *IEEE Data Eng. Bull.*, 23(3):9–16, 2000.

[147] Pablo N Mendes, Max Jakob, Andres Garcia-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011.

[148] Donald Metzler and W Bruce Croft. A Markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.

[149] Iris Miliaraki, Roi Blanco, and Mounia Lalmas. From Selena Gomez to Marlon Brando: Understanding Explorative Entity Search. In *Proceedings of the 24th International Conference on World Wide Web*, pages 765–775. International World Wide Web Conferences Steering Committee, 2015.

[150] David Milne and Ian H Witten. An open-source toolkit for mining Wikipedia. *Artificial Intelligence*, 194:222–239, 2013.

[151] David N Milne, Ian H Witten, and David M Nichols. A knowledge-based search engine powered by wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 445–454. ACM, 2007.

[152] Taiki Miyanishi, Kazuhiro Seki, and Kuniaki Uehara. Improving pseudo-relevance feedback via tweet selection. In *22nd ACM international conference on information & knowledge management*. ACM, 2013.

[153] Diego Molla, Menno Van Zaanen, and Daniel Smith. Named Entity Recognition for Question Answering. *Proceedings of ALTW*, pages 51–58, 2006.

[154] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: A Unified Approach. *Transactions of the Association for Computational Linguistics*, 2, 2014.

[155] Meredith Ringel Morris. Collaborative search revisited. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 1181–1192. ACM, 2013.

[156] Michalis Mountantonakis, Nikos Minadakis, Yannis Marketakis, Pavlos Fafalios, and Yannis Tzitzikas. Quantifying the connectivity of a semantic warehouse and understanding its evolution over time. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2014.

[157] Andrew Y Ng, Alice X Zheng, and Michael I Jordan. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 903–910. Citeseer, 2001.

[158] Andrew Y Ng, Alice X Zheng, and Michael I Jordan. Stable algorithms for link analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–266. ACM, 2001.

[159] Zaiqing Nie, Yuanzhi Zhang, Ji-Rong Wen, and Wei-Ying Ma. Object-level ranking: bringing order to web objects. In *Proceedings of the 14th international conference on World Wide Web*, pages 567–574. ACM, 2005.

[160] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. Sindice. com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1):37–52, 2008.

[161] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the Web. 1999.

[162] Jeff Z Pan, Edward Thomas, and Derek Sleeman. Ontosearch2: Searching and querying web ontologies. *Proc. of WWW/Internet*, 2006:211–218, 2006.

[163] P. Papadakos, S. Kopidaki, and Y. Armenatzoglou, N.and Tzitzikas. Exploratory Web Searching with Dynamic Taxonomies and Results Clustering. In *Proceedings of the 13th European Conference on Digital Libraries*, September 2009.

[164] Panagiotis Papadakos, Nikos Armenatzoglou, Stella Kopidaki, and Yannis Tzitzikas. On exploiting static and dynamically mined metadata for exploratory web searching. *Knowledge and Information Systems*, 30:493–525, 2012.

[165] Laurence AF Park and Kotagiri Ramamohanarao. Query expansion using a collection dependent probabilistic latent semantic thesaurus. In *Advances in Knowledge Discovery and Data Mining*. Springer, 2007.

[166] Gabriel Pinski and Francis Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing & Management*, 12(5):297–312, 1976.

[167] Robin L Plackett. Karl Pearson and the chi-squared test. *International Statistical Review/Revue Internationale de Statistique*, pages 59–72, 1983.

[168] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov. KIM–a semantic platform for information extraction and retrieval. *Natural language engineering*, 10(3-4):375–392, 2004.

[169] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th international conference on World wide web*, pages 771–780. ACM, 2010.

[170] W. Pratt and L. Fagan. The usefulness of dynamically categorizing search results. *Journal of the American Medical Informatics Association*, 7(6):605–617, 2000.

[171] Davood Rafiei and Alberto O Mendelzon. What is this page known for? Computing Web page reputations. *Computer Networks*, 33(1):823–835, 2000.

[172] Anand Rajaraman, Jeffrey D Ullman, Jeffrey David Ullman, and Jeffrey David Ullman. *Mining of massive datasets*, volume 1. Cambridge University Press Cambridge, 2012.

[173] Dietrich Rebholz-Schuhmann, Miguel Arregui, Sylvain Gaudan, Harald Kirsch, and Antonio Jimeno. Text processing through Web services: calling Whatizit. *Bioinformatics*, 24(2), 2008.

[174] Matthew Richardson and Pedro Domingos. The Intelligent surfer: Probabilistic Combination of Link and Content Information in PageRank. In *NIPS*, pages 1441–1448, 2001.

[175] Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. Statistical machine translation for query expansion in answer retrieval. In *Annual Meeting Association For Computational Linguistics*, 2007.

[176] Antonio M Rinaldi. An ontology-driven approach for semantic information retrieval on the Web. *ACM Transactions on Internet Technology (TOIT)*, 9(3):10, 2009.

[177] Giuseppe Rizzo and Raphael Troncy. NERD: Evaluating Named Entity Recognition Tools in the Web of Data. In *ISWC 2011, Workshop on Web Scale Knowledge Extraction (WEKEX'11)*, Bonn, GERMANY, 10 2011.

[178] Giuseppe Rizzo and Raphael Troncy. NERD: A Framework for Unifying Named Entity Recognition and Disambiguation Extraction Tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76. Association for Computational Linguistics, 2012.

[179] Giuseppe Rizzo, Raphael Troncy, Sebastian Hellmann, and Martin Bruemmer. NERD meets NIF: Lifting NLP Extraction Results to the Linked Data Cloud. *LDOW*, 937, 2012.

[180] Antonio J Roa-Valverde and Miguel-Angel Sicilia. A survey of approaches for ranking on the web of data. *Information Retrieval*, pages 1–31.

[181] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[182] Joseph John Rocchio. Relevance feedback in information retrieval. 1971.

[183] C. Rocha, D. Schwabe, and M.P. Aragao. A hybrid approach for searching in the Semantic Web. In *13th international conference on World Wide Web*, 2004.

[184] Alistair Russell, Paul R. Smart, Dave Braines, and Nigel R. Shadbolt. NITELIGHT: A Graphical Tool for Semantic Query Construction. In *Semantic Web User Interaction Workshop (SWUI 2008)*, April 2008.

[185] G.M. Sacco and Y. Tzitzikas. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, volume 25. Springer, 2009.

[186] Tetsuya Sakai. Ranking the NTCIR systems based on multigrade relevance. In *Information Retrieval Technology*, pages 251–262. Springer, 2004.

[187] Tetsuya Sakai. Alternatives to bpref. In *30th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007.

[188] Michail Salampasis and Allan Hanbury. A Generalized Framework for Integrated Professional Search Systems. In *Multidisciplinary Information Retrieval*, pages 99–110. Springer, 2013.

[189] Robert Sanderson, Paolo Ciccarese, and Herbert Van de Sompel. Open Annotation Data Model. *W3C Community Draft*, 2013.

[190] Michael Schuhmacher, Laura Dietz, and Simone Paolo Ponzetto. Ranking Entities for Web Queries Through Text and Knowledge. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1461–1470. ACM, 2015.

[191] Matthew S Simpson, Ellen M Voorhees, and William Hersh. Overview of the TREC 2014 Clinical Decision Support Track. Technical report, DTIC Document, 2014.

[192] Kostas Stefanidis, Vasilis Efthymiou, Melanie Herschel, Vassilis Christophides, et al. Entity Resolution in the Web of Data. In *WWW (Companion Volume)*, pages 203–204, 2014.

[193] Renxu Sun, Chai-Huat Ong, and Tat-Seng Chua. Mining dependency relations for query expansion in passage retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006.

[194] Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th annual international ACM*

SIGIR conference on Research and development in information retrieval, pages 162–169. ACM, 2006.

[195] Arthur R Taylor, Colleen Cool, Nicholas J Belkin, and William J Amadio. Relationships between categories of relevance criteria and stage in task completion. *Information Processing & Management*, 43(4):1071–1084, 2007.

[196] John A Tomlin. A new paradigm for ranking pages on the world wide web. In *Proceedings of the 12th international conference on World Wide Web*, pages 350–355. ACM, 2003.

[197] Thanh Tran, Peter Mika, Haofen Wang, and Marko Grobelnik. SemSearch'11: the 4th semantic search workshop. In *Proceedings of the 20th international conference companion on World wide web*, pages 315–316. ACM, 2011.

[198] Y. Tzitzikas, C. Alloca, C. Bekiari, Y. Marketakis, P. Fafalios, M. Doerr, N. Minadakis, T. Patkos, and L. Candela. Integrating Heterogeneous and Distributed Information about Marine Species through a Top Level Ontology. In *Proceedings of the 7th Metadata and Semantic Research Conference (MTSR'13)*, November 2013.

[199] Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. Faceted Exploration of RDF/S Datasets: A Survey. *Journal of Intelligent Information Systems (JIIS)*, 2016.

[200] Yannis Tzitzikas, Nikos Minadakis, Yannis Marketakis, Pavlos Fafalios, Carlo Allocca, and Michalis Mountantonakis. Quantifying the connectivity of a semantic warehouse. In *EDBT/ICDT Workshops*, pages 249–256, 2014.

[201] Yannis Tzitzikas and Panagiotis Papadakos. Interactive exploration of multidimensional and hierarchical information spaces with real-time preference elicitation. *Fundamenta Informaticae*, 122(4):357–399, 2013.

[202] Jurgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira. Hybrid SPARQL Queries: Fresh vs. Fast Results. In *The Semantic Web–ISWC 2012*, pages 608–624. 2012.

[203] Rachanee Ungrangsi, Chutiporn Anutariya, and Vilas Wuwongse. SQORE-based ontology retrieval system. In *Database and Expert Systems Applications*, pages 720–729. Springer, 2007.

[204] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Roder, Daniel Gerber, Sandro Athaide Coelho, Soren Auer, and Andreas Both. AGDISTIS - Graph-Based Disambiguation of Named Entities using Linked Data. In *The Semantic Web–ISWC 2014*. Springer, 2014.

[205] Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, Euripides GM Petrakis, and Evangelos E Milios. Semantic similarity methods in wordNet and their application to information retrieval on the web. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 10–16. ACM, 2005.

[206] Robert A Virzi. Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough? *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 34(4):457–468, 1992.

[207] Ellen Voorhees, Donna K Harman, et al. *TREC: Experiment and evaluation in information retrieval*, volume 63. MIT press Cambridge, 2005.

[208] Xiaogang Wang, Shi Qiu, Ke Liu, and Xiaoou Tang. Web Image Re-Ranking Using Query-Specific Semantic Signatures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(4):810–823, 2014.

[209] Ryen W White and Resa A Roth. Exploratory search: Beyond the query-response paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–98, 2009.

[210] Stewart Whiting, Iraklis Klampanos, and Joemon Jose. Temporal pseudo-relevance feedback in microblog retrieval. In *Advances in Information Retrieval*. Springer, 2012.

[211] M.L. Wilson et al. A longitudinal study of exploratory and keyword search. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '08, pages 52–56. ACM, 2008.

[212] Wenpu Xing and Ali Ghorbani. Weighted pagerank algorithm. In *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*, pages 305–314. IEEE, 2004.

[213] Chenyan Xiong and Jamie Callan. Query expansion with Freebase. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, pages 111–120. ACM, 2015.

[214] Yang Xu, Gareth JF Jones, and Bin Wang. Query dependent pseudo-relevance feedback based on wikipedia. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 59–66. ACM, 2009.

[215] Emine Yilmaz and Javed A Aslam. Estimating average precision with incomplete and imperfect judgments. In *15th ACM international conference on Information and knowledge management*. ACM, 2006.

[216] Emine Yilmaz, Evangelos Kanoulas, and Javed A Aslam. A simple and efficient sampling method for estimating AP and NDCG. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 603–610. ACM, 2008.

[217] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables. *Proceedings of the VLDB Endowment*, 4(12):1450–1453, 2011.

[218] Shipeng Yu, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma. Improving pseudo-relevance feedback in Web information retrieval using web page segmentation. In *12th international conference on World Wide Web*. ACM, 2003.

[219] S. Zampetakis, Y. Tzitzikas, A. Leonidis, and D. Kotzinos. Star-like auto-configurable layouts of variable radius for visualizing and exploring RDF/S ontologies. *Journal of Visual Languages & Computing*, 23(3), 2012.

[220] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

[221] X. Zhang, G. Cheng, and Y. Qu. Ontology summarization based on rdf sentence graph. In *16th international conference on World Wide Web*, 2007.

[222] Ziming Zhuang and Silviu Cucerzan. Re-ranking search results using query logs. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 860–861. ACM, 2006.

[223] Hilal Zitouni, Sare Sevil, Derya Ozkan, and Pinar Duygulu. Re-ranking of web image search results using a graph algorithm. In *19th International Conference on Pattern Recognition, 2008*. IEEE, 2008.

Appendix A
Configuration Forms of X-Link Evaluation Prototype

Here we give screenshots of the Web application we developed (described in 4.6.1) for evaluating the usability of the `X-Link` framework.

Figure A.1 depicts the form through which the user can *add* a new category of entities. We notice that the user can give a name for the category, the SPARQL endpoint of a SKB, and a SPARQL query. Then, she/he can run the query and inspect the entity names. In this way, the user can directly make any corrections at the displayed entity names, or run another SPARQL query and inspect the new result. At the end, the user can add the new category to `X-Link` by clicking the "Add" button. Figure A.2 shows the corresponding form for *updating* an existing category. Figure A.3 shows the form through which the user can specify how to *link* the identified entities with resources from a SKB. Notice that the user gets informed about how to create the SPARQL template query, while she/he can also load an example template query. Finally, Figure A.4 depicts the form that allows the user to specify how to *enrich* the identified entities with semantic information coming from a SKB.

Add a Category by: Sparql Query

Category Name

Endpoint

sparql query

Click here to load an example query

Run the SPARQL query and load the entity names

Entities

Add Clear

Figure A.1: The form for adding a new category of entities in X-Link.

Select category for update: Fish_Species

Select

Update a Category by: Sparql Query

Category Name: Fish_Species

Endpoint:

sparql query:

Click here to load an example query

Run the SPARQL query and load the entity names

Entities:

Update Clear

Figure A.2: The form for updating a category in X-Link.

Select category for specifying linking query

Fish_Species

Select

Endpoint

Please give the SPARQL template query

(The sparql template query must contain the character sequence [ENTITY] which corresponds to the name of an identified entity.)

Click here to load an example query.

Example of filter expression to compare strings (ignoring case):

```
FILTER(regex(str(?label),"[ENTITY]",'i'))
```

Example of filter expression to get strings only in English:

```
FILTER(lang(?label)='en')
```

Set Query

Figure A.3: The form for specifying how to link the identified entities in X-Link.

Select category for specifying enriching query

Fish_Species

Select

Endpoint

Please give the SPARQL template query
(The sparql template query must contain the character sequence [URI] which corresponds to the URI of an identified entity.)
Click here to load an example query.

Set Query

Figure A.4: The form for specifying how to enrich the identified entities in X-Link.

Appendix B
Evaluation Metrics for Incomplete Relevance Judgements

Here we describe and give the formulas of the metrics used in the evaluation of the proposed method for re-ranking the list of retrieved results (the method and its evaluation have been analyzed in 5.4 and 5.5.3, respectively).

We used the following metrics which have been specially designed for evaluation environments with incomplete relevance data:

- `Bpref`: as proposed in [43]
- `AveP'`: Average Precision based on a condensed list (after removing all unjudged docs) as proposed in [187]
- `nDCG'`: Normalized Discounted Cumulative Gain based on a condensed list as proposed in [187]
- `Q'`: Q-Measure on a condensed list as proposed in [187]
- `rpref_relative2`: as proposed in [187]
- `P@10'`: Precision at rank 10 based on a condensed list

Below, we provide the formulas of the above metrics as defined by Tetsuya Sakai [187].

In our experiments, we evaluated the top-100 lists. At first, let R denote the number of judged relevant documents in a top-100 list and let N denote the number of judged non-relevant documents. Since the following metrics rely on judged documents only, we can safety remove all unjudged documents from the list before computing the metrics. The result is a condensed list (with ≤ 100 documents) containing judged documents only. Let $r'(\leq r)$ denote the new rank of a document in the condensed list. Moreover, let $numrel(r')$ be the number of relevant documents in top-r' of the ranked list, and $isrel(r')$ be a function that returns 1 if the document at rank r' is relevant and 0 if it is not relevant.

Bpref

`Bpref` was introduced by Buckley and Voorhees [43] and is an IR metric based on binary

relevance. `Bpref` is defined as:

$$bpref = \frac{1}{R}\sum_{r'} isrel(r')(1 - \frac{min(R, r' - numrel(r'))}{min(R, N)}) \tag{B.1}$$

Notice that $r' - numrel(r')$ is the number of judged non-relevant documents ranked above the document at rank r'.

This formula is highly correlated with average precision (AveP) when full relevance assessments are available and is more robust when the relevance assessments are reduced. It penalizes a system that ranks a judged non-relevant document above a judged relevant one.

AveP'

Average Precision based on a condensed list is defined as:

$$AveP' = \frac{1}{R}\sum_{r'} isrel(r')\frac{numrel(r')}{r'} \tag{B.2}$$

nDCG'

Normalized Discounted Cumulative Gain (`nDCG`) uses a graded relevance scale and actually measures the usefulness, or gain, of a document based on its position in the list of results [119].

Let $\mathcal{L}$ denote a relevance level and let $gain(\mathcal{L})$ denote the gain value for retrieving a $\mathcal{L}$-relevant document. In our case, we have S-relevant (highly relevant) with gain = 2 and A-relevant (relevant) with gain = 1 in addition to judged non-relevant documents (for which gain = 0). The cumulative gain at a rank r is defined as:

$$cg(r) = \sum_{1 \le i \le r} gain(i) \tag{B.3}$$

Now, by listing up all S-relevant documents followed by all A-relevant documents we can produce an ideal ranked output. The cumulative gain of this ideal ranked output is the sum of all available gain values, specifically:

$$cg_I(R) = \sum_{\mathcal{L}} R(\mathcal{L})gain(\mathcal{L}) \tag{B.4}$$

where $R(\mathcal{L})$ denotes the number of $\mathcal{L}$-relevant documents.

However, highly relevant documents appearing lower in the list of search results should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. Thus, the *Discounted Cumulative Gain* at a rank r is defined as:

$$DCG(r) = \sum_{1 \leq i \leq r} \frac{2^{gain(i)} - 1}{log_2(i+1)} \tag{B.5}$$

Likewise, $IDCG(r)$ at a rank r is the discounted cumulative gain of an *ideal* ranked list. Now, the *normalized* Discounted Cumulative Gain at the rank $r = R + N$ on the *condensed* list is defined as:

$$nDCG'(r) = \frac{DCG(r)}{IDCG(r)} \tag{B.6}$$

Q'

`Q-Measure`, proposed by Tetsuya Sakai [186], is another evaluation metric that can handle graded relevance scale. Q-Measure on a *condensed* list (`Q'`) is defined as:

$$Q' = \frac{1}{R} \sum_{r'} isrel(r') \frac{cg(r') + numrel(r')}{cg_I(r') + r'} \tag{B.7}$$

where $cg(r')$ is the cumulative gain at rank r' and $cg_I(r')$ is the cumulative gain of an ideal ranked list (as defined in the above analysis of `nDCG'`). `Q'` is highly correlated with AveP and its discriminative power is known to be at least as high as that of AveP.

rpref˙relative2

`Rpref` is an alternative to `bpref`, designed to handle grade relevance [74]. `rpref_relative2` is an extension of `Rpref` introduced by Tetsuya Sakai [187] and is defined as:

$$rpref\dot{}relative2 = \frac{1}{cg_I(R)} \sum_{r',\, gain(r')>0} gain(r')(1 - \frac{penalty(r')}{r'}) \tag{B.8}$$

where:

$$penalty(r') = \sum_{i'<r',\, gain(i')<gain(r')} \frac{gain(r') - gain(i')}{gain(r')} \tag{B.9}$$

`rpref_relative2` uses relative normalization to *emphasize* misplacement penalties based on highly ranked relevant documents (it also handles some flaws of `Rpref`).

P@10'

Precision at rank 10 (P@10) corresponds to the number of relevant results on the first search results page (which usually contains 10 results). Precision at rank 10 on a con-

densed list (P@10') is simply defined as:

$$P@10' = \frac{numrel(10)}{10} \tag{B.10}$$

This metric favors condensed lists containing many judged relevant documents before the judged non-relevant documents early in the ranked list. However, it fails to take into account the positions of the relevant documents among the top-10 list. Another shortcoming is that if the total number of relevant results is less than 10, even a perfect system will have a score less than 1.

Appendix C
Publications, Systems and Models

Publications

The research activity related to this thesis has so far produced the following publications (ordered by publication date):

(1) P. Fafalios, I. Kitsos, Y. Marketakis, C. Baldassarre, M. Salampasis and Y. Tzitzikas, *Web Searching with Entity Mining at Query Time*, Proceedings of the 5th Information Retrieval Facility Conference, IRFC 2012, Vienna, Austria, July 2012

(2) P. Fafalios, M. Salampasis and Y. Tzitzikas, *Exploratory Patent Search with Faceted Search and Configurable Entity Mining*, Proceedings of the 1st International Workshop on Integrating IR technologies for Professional Search, in conjunction with the 35th European Conference on Information Retrieval (ECIR'13), Moscow, Russia, March 2013.

(3) P. Fafalios and Y. Tzitzikas, *X-ENS: Semantic Enrichment of Web Search Results at Real-Time*, Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, Demo Paper, SIGIR 2013, Dublin, Ireland, 28 July - 1 August 2013

(4) P. Fafalios, M. Baritakis and Y. Tzitzikas *Configuring Named Entity Extraction through Real-Time Exploitation of Linked Data*, 4th International Conference on Web Intelligence, Mining and Semantics (WIMS'14), Thessaloniki, Greece, June 2014.

(5) P. Fafalios and P. Papadakos, *Theophrastus: On Demand and Real-Time Automatic Annotation and Exploration of (Web) Documents using Open Linked Data*, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 29 (2014), Elsevier

(6) P. Fafalios and Y. Tzitzikas, *Post-Analysis of Keyword-based Search Results using Entity Mining, Linked Data and Link Analysis at Query Time*, IEEE 8th International Conference on Semantic Computing (ICSC'14), Newport Beach, California, USA, June 2014

(7) P. Fafalios and Y. Tzitzikas, *Exploratory Professional Search through Semantic Post-Analysis of Search Results*, Professional Search in the Modern World, Lecture Notes in Computer Science (LNCS), Springer, 2014

(8) P. Fafalios, P. Papadakos and Y. Tzitzikas, *Enriching Textual Search Results at Query Time using Entity Mining, Linked Data and Link Analysis*, International Journal of Semantic Computing, Vol. 08, No. 04 (2014), World Scientific

(9) P. Fafalios and Y. Tzitzikas, *Type-ahead Exploratory Search through Typo and Word Order Tolerant Autocompletion*, Journal of Web Engineering, Vol. 14, No. 1 & 2 (2015), Rinton Press

(10) P. Fafalios, M. Baritakis and Y. Tzitzikas, *Exploiting Linked Data for Open and Configurable Named Entity Extraction*, International Journal on Artificial Intelligence Tools, World Scientific, 2015

(11) P. Fafalios and Y. Tzitzikas, *Entity-based Stochastic Analysis of Search Results for Query Expansion and Results Re-Ranking*, Text Retrieval Conference (TREC'15), Gaithersburg, Maryland, USA, November 2015

In more details (and with regard to the contributions of this thesis):

- In (1) we started exploiting the LOD to enrich keyword-based Web searching with entity mining that is performed at query-time.
- In (2) we apply and test this approach in the context of a Patent Search system.
- In (3) we demonstrate the `X-ENS` system which exploits the LOD for retrieving (on-demand) more information for the identified entities, and which allows the user/administator to configure the underlying search system, the entities of interest, and the information to retrieve for the detected entities.
- In (4) we elaborate on the configurability of a NEE system and we present the `X-Link` framework.
- In (5) we exploit the configuration model proposed in (4) in the context of an entity annotation and exploration system tailored for marine biology, called `Theophrastus`.
- In (6) we introduce an exploratory search process in which the search results are connected with LOD at query time with no human effort, and we detail and evaluate a Link Analysis (Random Walk-based) approach for selecting the semantic information that better characterizes the search results.
- In (7) we present an exploratory strategy for Professional Search which combines the metadata that are already available, the result of textual clustering and the result of entity mining that can be performed at query time. We also propose interaction models that exploit the result of the above processes, we present application examples, and finally we introduce the `X-Search` system.
- In (8) we provide more technical details and experimental results regarding the search paradigm and the Random Walk-based ranking scheme introduced in (6).
- In (9) we show how such exploratory search services can be offered during query typing, through a richer autocompletion service called `IOS` (Instant Overview Search).
- In (10) we introduce the Open NEE Configuration Model and the extension of the

Open Annotation Data Model, and we propose and evaluate methods for ranking the semantic resources that match an entity name.

- In (11) we study whether and how we can exploit entities from the emerging Web of Data for enhancing IR, specifically for automatic query expansion and for re-ranking a list of retrieved results.

Systems and Models

In the context of this thesis, the following systems and models were developed:

- `X-Link` (`http://www.ics.forth.gr/isl/X-Link`)
- `X-Search` (`http://www.ics.forth.gr/isl/X-Search`)
- `Theophrastus` (`http://www.ics.forth.gr/isl/Theophrastus`)
- `IOS` (`http://www.ics.forth.gr/isl/ios`)
- Open NEE Configuration Model (`http://www.ics.forth.gr/isl/oncm`)
- Open Annotation Extension Model (`http://www.ics.forth.gr/isl/oae`)

Appendix D
Acronyms

EoI Entities of Interest

FLOD FAO Fisheries Linked Open Data

GKG Google Knowledge Graph

IR Information Retrieval

KBM Knowledge Base Mirror

LOD Linked Open Data

ML Machine Learning

NED Named Entity Disambiguation

NEE Named Entity Extraction

NER Named Entity Recognition

NLP Natural Language Processing

SEGIE Semantically-Enriched Graph of Identified Entities

SGODE Semantic Graph Of Documents and Entities

SKB Semantic Knowledge Base

STG State Transition Graph

UI User Interface