

Lehrstuhl für Verteilte Informationssysteme  
Fakultät für Informatik und Mathematik  
Universität Passau



Doctoral Thesis

# Semantic Snippets via Query-Biased Ranking of Linked Data Entities

Conducted as cotutelle-de-thèse in cooperation with

Laboratoire LIRIS  
INSA de Lyon  
Doctoral School InfoMaths  
Lyon, France

M.Sc. Mazen Alsarem

March 2016



# *Acknowledgements*

Prima facie, I would like to express my sincere gratitude to my advisors Prof. Sylvie Calabretto, and Prof. Harald Kosch for their continuous support of my PhD study and research, for their patience and tenacity over the last years. Their guidance and immense knowledge helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisors for my Ph.D study.

I would like to express my sincere gratitude to my co-advisor Dr. Pierre-Edouard Portier for his constant support, guidance and motivation. It would never have been possible for me to take this work to completion without his incredible support. I benefited greatly from our many fruitful discussions.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Harald Sack, Prof. Gabriella Pasi, Prof. Mohand Boughanem, Dr. Fabien Gandon, and Prof. Michael Granitzer, for their interest in my work, their input and insightful comments.

I would like to say thanks the Université Franco-Allemande (UFA) for the financial and scientific support. I would also thank all the MDPS members for all the discussions and the fun that we have had.

I also thank Dr. Tobias Mayer and Dr. David Coquil who generously gave up a lot of time to help me.

I thank my fellow labmates: Albin Petit, Diana Nurbakova, Vincent Primault, Azhar Ait Ouassarah and Merza Klaghstan, for the stimulating discussions and for the sleepless nights we were working together before deadlines.

I would like to give special thanks to my close friends Vincent Barrellon, Manel Charfi, Zeina Torbey and Georges Takkouz for all the fun we have had in the last years.

I would also like to thank my family in Syria, and my step family all around the world. They were always supporting me and encouraging me with their love and best wishes.

Finally, a special thanks from the heart to my lovely wife Salam for her support. You help me to regain hope after despair, resume life after obstructions, restart journeys after detours, revive strength after defeat and resurrect dreams after rejection. Thank you.



*Dedicated to my wife **Salam** and my daughter **Julia**.*

*This work is also dedicated to Syrian people in their noble and peaceful quest for freedom, human rights and justice.*



# *Abstract*

In our knowledge-driven society, the acquisition and the transfer of knowledge play a principal role. Web search engines are somehow tools for knowledge acquisition and transfer from the web to the user. The search engine results page (SERP) consists mainly of a list of links and snippets (excerpts from the results). The snippets are used to express, as efficiently as possible, the way a web page may be relevant to the query.

As an extension of the existing web, the semantic web or “web 3.0” is designed to convert the presently available web of unstructured documents into a web of data consumable by both human and machines. The resulting web of data and the current web of documents coexist and interconnect via multiple mechanisms, such as the embedded structured data, or the automatic annotation.

In this thesis, we introduce a new interactive artifact for the SERP: the “Semantic Snippet”. Semantic Snippets rely on the coexistence of the two webs to facilitate the transfer of knowledge to the user thanks to a semantic contextualization of the user’s information need. It makes apparent the relationships between the information need and the most relevant entities present in the web page.

The generation of semantic snippets is mainly based on the automatic annotation of the LOD<sup>1</sup>’s entities in web pages. The annotated entities have different level of importance, usefulness and relevance. Even with state of the art solutions for the automatic annotations of LOD entities within web pages, there is still a lot of noise in the form of erroneous or off-topic annotations. Therefore, we propose a query-biased algorithm (LDRANK) for the ranking of these entities. LDRANK adopts a strategy based on the linear consensual combination of several sources of prior knowledge (any form of contextual knowledge, like the textual descriptions for the nodes of the graph) to modify a PageRank-like algorithm.

For generating semantic snippets, we use LDRANK to find the more relevant entities in the web page. Then, we use a supervised learning algorithm to link each selected entity to excerpts from the web page that highlight the relationship between the entity and the original information need.

In order to evaluate our semantic snippets, we integrate them in ENsEN (Enhanced Search Engine), a software system that enhances the SERP with semantic snippets.

Finally, we use crowdsourcing to evaluate the usefulness and the efficiency of ENsEN.

**Keywords:** Semantic Snippets, Entity Ranking, Web of Data.

---

<sup>1</sup>Linking Open Data





# Résumé

Dans notre société fondée sur la connaissance, l'acquisition et le transfert de connaissances jouent un rôle principal. Les moteurs de recherche sur le Web sont en quelque sorte des outils d'acquisition et de transfert des connaissances du Web à l'utilisateur. La page de résultats d'un moteur de recherche (Search Engine Results Page - SERP) se compose principalement d'une liste de liens et de snippets (extraits à partir des résultats). Les snippets sont utilisés pour exprimer, aussi efficacement que possible, la façon dont une page Web peut être pertinente pour la requête.

Le Web sémantique ou "Web 3.0" est conçu pour transformer le Web de documents non structurés en un Web de données exploitable à la fois par les machines et les humains. Le Web de données obtenu et le Web de documents actuel coexistent et sont interconnectés via de multiples mécanismes, tels que les données structurées intégrées dans les pages Web, ou l'annotation automatique.

Dans cette thèse, nous introduisons un nouvel artefact interactif pour le SERP: le "Snippet Sémantique". Les snippets sémantiques s'appuient sur la coexistence des deux Webs pour faciliter le transfert des connaissances aux utilisateurs grâce à une contextualisation sémantique du besoin d'information de l'utilisateur. Ils font apparaître les relations entre le besoin d'information et les entités les plus pertinentes présentes dans la page Web.

La génération des snippets sémantiques repose principalement sur l'annotation automatique des entités de LOD dans les pages Web. Les entités annotées ont des niveaux d'importance, d'utilité et de pertinence différents. Les solutions de l'état de l'art pour l'annotation automatique des entités LOD dans les pages Web génèrent encore beaucoup de bruit sous la forme d'annotations erronées ou hors sujet. Par conséquent, nous proposons un algorithme biaisé-requête (LDRANK) pour l'ordonnement de ces entités. LDRANK adopte une stratégie basée sur la combinaison consensuelle linéaire de plusieurs sources de connaissances a priori (toute forme de connaissances contextuelles, comme les descriptions textuelles des noeuds du graphe) pour modifier un algorithme de type PageRank. Pour générer des snippets sémantiques, nous utilisons LDRANK pour trouver les entités les plus pertinentes dans la page Web. Ensuite, nous employons un algorithme d'apprentissage supervisé pour lier chaque entité sélectionnée à des extraits de la page Web qui mettent en évidence la relation entre l'entité et le besoin d'information original.

Afin d'évaluer nos snippets sémantiques, nous les intégrons dans ENsEN (Enhanced Search Engine), un système logiciel qui améliore le SERP avec des snippets sémantiques. Enfin, nous utilisons le crowdsourcing pour évaluer l'utilité et l'efficacité de ENsEN.

**Mots-clés:** Semantic Snippets, Ordonnement d'entités, Web de Données.



# *Zusammenfassung*

In unserer heutigen Wissensgesellschaft spielen der Erwerb und die Weitergabe von Wissen eine zentrale Rolle. Internetsuchmaschinen fungieren als Werkzeuge für den Erwerb und die Weitergabe von Wissen aus dem Web an den Nutzer. Die Ergebnisliste einer Suchmaschine (SERP) besteht grundsätzlich aus einer Liste von Links und Textauszügen (Snippets). Diese Snippets sollen auf möglichst effiziente Weise ausdrücken inwiefern eine Webseite für die Suchanfrage relevant ist. Als Erweiterung des bestehenden Internets, überführt das semantische Web - auch genannt "Web 3.0" - das momentan vorhandene Internet der unstrukturierten Dokumente in ein Internet der Daten, das sowohl von Menschen als auch Maschinen verwendet werden kann. Das neu geschaffene Internet der Daten und das derzeitige Internet der Dokumente existieren gleichzeitig und sie sind über eine Vielzahl von Mechanismen miteinander verbunden, wie beispielsweise über eingebettete strukturierte Daten oder eine automatische Annotation. In dieser Arbeit stellen wir ein neues interaktives Artefakt für das SERP vor: Das "Semantische Snippet". Semantische Snippets stützen sich auf die Koexistenz der beiden Arten des Internets um mit Hilfe der Kontextualisierung des Informationsbedürfnisses eines Nutzers die Weitergabe von Wissen zu erleichtern. Sie stellen die Verbindung zwischen dem Informationsbedürfnis und den besonders relevanten Entitäten einer Webseite heraus. Die Erzeugung semantischer Snippets basiert überwiegend auf der automatisierten Annotation von Webseiten mit Entitäten aus der Linking Open Data Cloud (LOD). Die annotierten Entitäten besitzen unterschiedliche Ebenen hinsichtlich Wichtigkeit, Nützlichkeit und Relevanz. Selbst bei state-of-the-art Lösungen zur automatisierten Annotation von LOD-Entitäten in Webseiten, gibt es stets ein großes Maß an Rauschen in Form von fehlerhaften oder themenfremden Annotationen. Wir stellen deshalb einen anfragegetriebenen Algorithmus (LDRANK) für das Ranking dieser Entitäten vor. LDRANK setzt eine Strategie ein, die auf der linearen Konsensuskombination (engl. linear consensual combination) mehrerer a-priori Wissensquellen (jedwede Art von Kontextwissen, wie beispielsweise die textuelle Beschreibung der Knoten des Graphen) basiert um damit den PageRank-Algorithmus zu modifizieren. Zur Generierung semantischer Snippets finden wir zunächst mit Hilfe von LDRANK die relevantesten Entitäten in einer Webseite. Anschließend verwenden wir ein überwachtetes Lernverfahren um jede ausgewählte Entität denjenigen Abschnitten der Webseite zuzuordnen, die die Beziehung zwischen der Entität und dem ursprünglichen Informationsbedarf am Besten herausstellt. Um unsere semantischen Snippets zu evaluieren, integrieren wir sie in ENsEN (Enhanced Search Engine), ein Softwaresystem das SERP um semantische Snippets erweitert. Zum Abschluss bewerten wir die Nützlichkeit und die Effizienz von ENsEN mittels Crowdsourcing.

**Schlagwörter:** Semantische Snippets, Ranking von Entitäten, Web of Data.



# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>ix</b>
<b>Résumé</b>	<b>xi</b>
<b>Zusammenfassung</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xix</b>
<b>List of Tables</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Context	5
1.2 Background	6
1.2.1 The “Web”	6
1.2.2 Web of Data	6
1.2.3 Co-existence of the two web (web of documents and web of data)	8
1.2.4 Web Information Retrieval	10
1.2.5 Search User Interface	12
1.3 Statement of the Problem	14
1.4 Research Questions	15
1.5 Application Scenarios	16
1.5.1 Application Scenario 1: General Vs. Specific user information need	16
1.5.2 Application Scenario 2: Answering directly (in SERP) hard questions	19
1.6 Summary of Contributions	24
1.6.1 Query-biased Ranking for Linked Data Entities	24
1.6.2 “Semantic Snippets” and the Enhanced Search Engine (ENsEN)	25
1.7 Structure of the Thesis	27
<b>2 A Query-biased Ranking for LOD Entities</b>	<b>29</b>
2.1 Introduction and Context	30
2.2 Related Works	31
2.2.1 Comparative study	33

2.2.2	Selected approaches from the literature	38
2.2.3	Conclusion	49
2.3	Our Algorithm (LDRANK)	50
2.3.1	Context	50
2.3.2	Prior Knowledge Derived from the Ranking Provided by the Web Search Engine Result Page	51
2.3.3	Prior Knowledge Derived from an Iterative Latent Semantic Analysis of the Textual Data Describing the Entities	51
2.3.4	Belief Aggregation Strategy	53
2.3.5	LDRANK	54
2.4	Evaluation	54
2.4.1	Introduction	54
2.4.2	Build the evaluation dataset	55
2.4.3	Experiments	58
2.4.4	Results and discussion	59
2.5	Conclusion	59
<b>3</b>	<b>The Semantic Snippets</b>	<b>61</b>
3.1	Introduction	62
3.2	Related Work	63
3.2.1	Enhancing Snippets for the Semantic Web	63
3.2.2	Enhancing Snippets for the Web of documents	64
3.3	Elements composing a Semantic Snippet	66
3.4	Semantic Snippet as an Interactive Artifact	67
3.5	Semantic Snippets' Generation	69
3.6	Learning to rank documents' excerpts	71
3.6.1	Background	71
3.6.2	Related Work	76
3.6.3	Feature Engineering	78
3.6.4	Training Dataset	81
3.6.5	Re-Balancing the Training Dataset	82
3.6.6	First Results	83
3.6.7	Results using other regrouping strategies	83
3.6.8	Feature Selection	84
3.6.9	Selected Features	85
3.6.10	Conclusion	86
<b>4</b>	<b>Enhanced Search Engine (ENsEN)</b>	<b>87</b>
4.1	Introduction	88
4.2	Software Design and Architecture	88
4.2.1	External Services	88
4.2.2	High-level Architecture	90
4.2.3	Internal Components	90
4.2.4	Data Model	99
4.2.5	Workflow	101
4.3	User Interface	103
4.3.1	Visual Design	104

4.3.2	ENsEN design and Crowdsourcing . . . . .	104
4.4	Implementation . . . . .	106
4.4.1	Technological and architectural choices . . . . .	106
4.4.2	Hardware Configurations . . . . .	108
4.5	Crowdsourcing Evaluation of ENsEN . . . . .	108
4.5.1	Methodology . . . . .	108
4.5.2	Results Analysis . . . . .	111
4.6	Conclusion . . . . .	116
<b>5</b>	<b>Conclusion</b>	<b>117</b>
5.1	Research summary . . . . .	118
5.2	Future Work and Perspectives . . . . .	120
5.2.1	LDRANK: Exploring more prior knowledge sources . . . . .	120
5.2.2	Semantic Index . . . . .	122
5.2.3	SERP from documents to concepts . . . . .	123
5.2.4	Personalization and Recommendation . . . . .	124
5.2.5	ENsEN from prototype to product . . . . .	125
<b>A</b>	<b>Crowdsourcing Microtask Example for the Evaluation of ENsEN's search interface.</b>	<b>127</b>
A.1	Introduction . . . . .	128
A.2	Microtask . . . . .	129
A.3	Used topics and questions . . . . .	134
	<b>Bibliography</b>	<b>137</b>
	<b>Publications List</b>	<b>147</b>





# List of Figures

1.1	Structured Data Rankers	11
1.2	Google results for “dinosaurs” query	17
1.3	Google results for “dinosaurs communications” query	18
1.4	ENsEN results for “dinosaurs communications” query	19
1.5	Google results for “school produced the most justices for Supreme Court justices” query	20
1.6	ENsEN results for “school produced the most justices for Supreme Court justices” query	21
1.7	Most related concepts of “Supreme Court of the United States”	22
1.8	Entity description of “Harvard Law School”	23
1.9	Most related concepts of “Harvard Law School”	24
1.10	Elements of a semantic snippet.	26
1.11	Elements of a Entity description	26
2.1	A simple two-state Markov chain.	39
2.2	Directed graph representing a web of six pages.	40
2.3	The Markov chain matrix for the example graph.	40
2.4	The Markov chain stochastic matrix for the example graph	41
2.5	The stochastic-primitive matrix for the example graph	41
2.6	Paper Object Relationship Graph	46
2.7	DING: The two-layer model of the web of data	48
2.8	Comparison of the NDCG scores for the four different strategies ( <i>EQUI</i> , <i>HIT</i> , <i>SVD</i> and <i>LDRANK</i> )	58
2.9	Comparison of the execution time for the four different strategies	59
3.1	The semantic snippet layout	66
3.2	The Entity Description panel	68
3.3	Example of the generated datasets	82
3.4	The unbalance of the dataset	82
4.1	ENsEN Context and High-level Architecture (Component Diagram)	90
4.2	ENsEN Full Detailed Component Diagram	91
4.3	Example of a generated RDF graph	94
4.4	ENsEN Data Model Diagram	100
4.5	ENsEN Activity Diagram	102
4.6	ENsEN Homepage, the query page	104
4.7	SERP of ENsEN	105
4.8	ENsEN Deployment Diagram	107

4.9	Participants' expertise for searching on the web, based on the accuracy of their answers to the questionnaire . . . . .	111
4.10	Accuracy of the answers depending on the level of expertise (A,B,C,D,E) of the participants for the web search domain . . . . .	111
4.11	Accuracy of the answers per system . . . . .	112
4.12	Accuracy of the answers by topic and for each of the two systems . . . . .	113
4.13	Overall evaluation of the effectiveness and the ease of use for the factual questions . . . . .	114
4.14	Overall evaluation of the effectiveness and the ease of use for the questions with a list answer . . . . .	114
4.15	Ages of the participants . . . . .	115
4.16	Participants' search habits in terms of the kind of queries they are most likely to perform . . . . .	115
4.17	Preferred elements of the GUI . . . . .	116
5.1	The research dependency . . . . .	120
A.1	Google interface . . . . .	129
A.2	ENsEN interface . . . . .	130
A.3	ENsEN concept description interface . . . . .	130

# List of Tables

1.1	LOD cloud statistics <sup>3,4,5</sup> . . . . .	7
2.1	Comparative study: summary of dimensions and approaches . . . . .	36
2.2	The stochastic matrix of the last example . . . . .	39
2.3	Comparative study: Summary of dimensions and approaches . . . . .	43
3.1	Results for learning to select sentences apt to explain the relationship between an entity and the user’s information need . . . . .	83
3.2	Results for <i>dataset-2</i> . . . . .	84
3.3	Results for <i>dataset-3</i> . . . . .	84
3.4	Feature selection results for (i) the filter approach with the information gain metric, and for (ii) the wrapper approach with two greedy strategies for exploring the space of features’ subsets (values are the F1 score for the positive class (F1-True)) . . . . .	85
3.5	Feature selection results using the forward selection wrapper approach with the logistic regression algorithm and while the number of initial features varies from 0 to 10 according to their information gain score . . .	85
5.1	The summary of ENsEN’s response time per step (log of 3 queries and 5 results per query) . . . . .	126



# Chapter 1

## Introduction



**Preface**

**When seen up close, dangers are controllable:** when you begin to climb the mountain of your dreams, pay attention to the surroundings. There are cliffs, of course. There are almost imperceptible cracks in the mountain rock. There are stones so polished by storms that they have become as slippery as ice. But if you know where you are placing each footstep, you will notice the traps and how to get around them. [*Manual for Climbing Mountains, Paulo Coelho*]





## 1.1 Context

In the knowledge-driven society, the acquisition and the transfer of the knowledge play a principal role. If a user tries to acquire new knowledge, she needs information about the problem, the domain and the concepts around the problem. Understanding a problem requires building an internal representation of the acquired information as a knowledge structure (a set of concepts and relations). Building this structure is based on the information about how these concepts are inter-related and their degree of relevance to the problem.

Due to the high amount of information available (especially on the web), retrieving and presenting the suitable information is a laborious task that search engines (SE) try to handle. The search engine results page (SERP) is somehow a tool for the knowledge acquisition and transfer, it collects and transfers information from the web to the user. The elements of a SERP (mainly snippets) help the user in conceptualizing the answer to satisfy the information need. A snippet (as an excerpt from a web page determined at query-time, found under the page's title of each entry in a SERP) is used to express, as efficiently as possible, the way a web page may be relevant to the query.

A 2010 eye tracking study by [Marcos and González-Caro, 2010] shows that users spend more time looking at the snippet than looking at other parts of the result (e.g. title, URL, ...).

By making explicit the reason a document seems to meet, at least partially, the information need of a user, the snippet alleviates the semantic mismatch between relevance and 'aboutness'. Indeed, for most existing approaches the snippet is made of a short and continuous excerpt that contains the query's keywords. Thus, the snippet offers an explanation of the way the document is about the keywords. Given this, the user will more adequately evaluate the relevance of the document to her information need.

With current web search engines, this process still requires an extra effort from the user, finding the main subject of a web page, the most important concepts mentioned in the document and how they are related to each other and to her information need. The user still needs to contextualize her information need in each document.

In this thesis, we make use of the co-existence of the web of documents and the web of data to enhance the user experience when interacting with the SERP. For a better understanding of the context, some topics and technologies at the frontier between these two webs are clarified in the next section.

## 1.2 Background

As the principal domain of our thesis is enhancing the search interface using both webs (web of documents and web of data), we introduce in this section a general background regarding the web of documents, the web of data, how they co-exist, the web information retrieval and the search user interface.

While this chapter gives a rather broad overview and details some selected topics, specific and extended information on related work is given in the corresponding sections of the subsequent chapters.

### 1.2.1 The “Web”

Nowadays, the World Wide Web (WWW), commonly known as the “Web” is considered as the greatest and most widely used information system. Moreover the web knew several historical progressive stages (or versions): in 1989, the idea of the web (as “web 1.0”) was introduced by Tim Berners-Lee as a global hypertext space in which any network accessible information would be referred to by a single Universal Document Identifier (UDI). It was a readable (read-only) version with limited interactions between sites and users, entirely made up of static web pages with no interactive content, connected by hyperlinks. Then a richer interactive version “the web 2.0” was introduced. It is based on the users collaboration and the information sharing, with very efficient search engines. As the interactivity of the web evolves gradually over time, we cannot determine exactly where web 1.0 ends and web 2.0 begins. Finally, the semantic version “the web 3.0” was proposed as an extension of the existing WWW with a target to convert the presently available web of unstructured documents to a web of information/data consumable by both human and machines.

From technological and structural point-of-view, the web evolved from a graph of documents and hyperlinks, i.e. the “web of documents”, to a more informative and finer grained graph of entities and typed links, i.e. the “web of data”.

### 1.2.2 Web of Data

The last version of the web (*the Semantic Web (SW)*) was described by its inventor Tim Berners-Lee as “an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.” [Berners-Lee et al., 2001]. Likewise, the W3C defines the Semantic Web as a vision of providing a common framework that allows data to be shared and reused across

Year	Size
2006	LD introduced by Berners-Lee, 0 dataset
2007	28 datasets
2008	45 dataset
2009	94 dataset
2010	203 dataset, 26 billion RDF triples
2011	295 dataset, 30 billion triples
2012	300 dataset
2014	1014 dataset, 558 data sources

TABLE 1.1: LOD cloud statistics <sup>3,4,5</sup>

application, enterprise, and community boundaries <sup>1</sup>. The W3C issued a set of standards in order to promote common data formats and exchange protocols on the SW; the most important standard is the *Resource Description Framework (RDF)*, which is a metadata model for web resources description.

In 2006, Tim Berners-Lee wrote his “Linked Data - Design Issues” [Berners-Lee, 2006], where he introduced for the first time the term “*Linked Data (LD)*” as a method of publishing structured data on the web. This method consists of a list of four principles about how to share and connect related data across the web using URIs, HTTP, and RDF: (1) *Use URIs as names for things*, (2) *Use HTTP URIs so that people can look up those names*, (3) *When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)*, and (4) *Include links to other URIs. So that they can discover more things*.

The LD increases the data value by providing standardized mechanisms for describing the data and linking them to other datasets. Linking the datasets together using web technologies, and according to LD principles, creates a huge RDF graph called the “web of data”. The term *web of data* is largely interchangeable with the term *Semantic Web*.

The origins of the web of data lie in the efforts of the W3C Semantic Web research community and particularly in the activities of the Linking Open Data (LOD) project [Heath and Bizer, 2011]. The LOD cloud<sup>2</sup> gathers most of the datasets published on the web using the Linked Data principles. At the time of writing this thesis, it encloses about 1014 datasets describing 8 million entities, with 56.11% of the datasets linked to at least one other dataset. Table 1.1 shows some historical information about the LOD Cloud <sup>3 4 5</sup> [Schmachtenberg et al., 2014] .

<sup>1</sup>W3C Semantic Web Activity”. World Wide Web Consortium (W3C). November 7, 2011.

<sup>2</sup><http://lod-cloud.net/>

<sup>3</sup>State of the LOD Cloud 2011: <http://lod-cloud.net/state/>

<sup>4</sup>State of the LOD Cloud 2014: <http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

<sup>5</sup>Mannheim Linked Data Catalog: <http://linkeddatacatalog.dws.informatik.uni-mannheim.de>

The biggest and most referenced dataset in this cloud is DBpedia. DBpedia is a huge LD dataset extracted from Wikipedia [Bizer et al., 2007], its last version (2014) describes about 4.58 million entities in up to 125 different languages. It is considered as a highly connected Dataset due to its 3 billion RDF triples and the 50 million links towards other RDF datasets<sup>6</sup>.

### 1.2.3 Co-existence of the two web (web of documents and web of data)

The web of documents has a relatively low degree of explicitly structure data. It contains documents, with anchored links between them. Since it was designed for the human consumption, the semantics of its contents are implicit. On the other hand, the primary objects of the web of data are “*things*” (or *entities*) with typed links between them. In contrast to the web of document, it was designed for both machines and humans, the semantics of its contents and links are explicit, and it has a high degree of structure based on the RDF data model.

Nowadays, these two webs coexist. Furthermore, they are highly connected via multiple mechanisms. In the next sections, we mention the most common methods used to connect them.

#### 1.2.3.1 Embedding of structured data in the web of documents

The Linked Data publishing recipes separate the raw-data representations (RDF/XML, Turtle, N3, etc.) and the human-readable representations (HTML), but embedding structured Data directly into HTML allows us to navigate from the web of documents towards the web of data. This mechanism allows the definition and the use of a set of attributes to augment the presentation-oriented (HTML) documents with structured data, allowing the user agents to extract triples from web pages.

This mechanism covers methods like:

**RDFa**, a W3C Recommendation to add a set of attribute-level extensions to HTML, in order to mark words or phrases to be treated as semantic entities.

**Microformats** also extend conventional HTML tags with semantic information; they make use of (X)HTML attributes like “class” or “rel” to embed structured Data.

Finally, **Microdata** is a very recent HTML 5 proposition that extends Microformats and addresses its shortcomings. In Microdata, items are created within an itemscope, every item is assigned some properties (itemprop) and relationships (itemref).

---

<sup>6</sup>DBpedia Version 2014 released: <http://blog.dbpedia.org/?p=77>

### 1.2.3.2 Extracting structured data from web pages and publishing it on the web of data

The primary requirement in this category is to have tools that generate RDF from existing data sources, and then upload the RDF data into a triple store, to make it accessible through the web. This approach is employed by the DBpedia project, among others, the project uses PHP scripts to extract structured data from the infobox in the Wikipedia pages, and then this data is converted to RDF and stored in an OpenLink Virtuoso repository accessible via an SPARQL endpoint. The Open University in the UK also applies this approach in the project “LUCERO”<sup>7</sup> in order to expose and publish its organizational information in LOD [Zablith et al., 2011]. If the information is represented in formats such as XML, CSV, Microsoft Excel, or BibTEX, it can be converted into RDF using an RDFizing tool<sup>8</sup>.

### 1.2.3.3 Wrappers

Large numbers of web applications have started to make their data available on the web through web APIs. These APIs provide diverse query and retrieval interfaces and return results using a number of different formats. Thus, they make them inaccessible to generic data browsers and invisible to the search engines. To overcome these limitations, the Linked Data wrappers can be applied for the purpose of assigning URIs to the entities, when one of these URIs is requested from the web of data, the wrapper redirects (rewrite) the client’s request to a request against the underlying API and the results of the API request are transformed to RDF and sent back to the client.

### 1.2.3.4 Automatic Annotation

Finally, the *Automatic Annotation* that provides a solution for linking unstructured information sources to entities in the web of data. This approach goes through two phases: it starts by performing a Named Entity Recognition (NER) where the NLP technologies are used to analyze, locate and classify fragments of the text into predefined categories such as the names of persons, organizations, etc.

The next phase is to apply the Entity Linking (EL) or the named entity disambiguation (NED) in order to determine the identity of entities mentioned in the text. As a result, the named entities are then linked to the URIs of web of data entities.

---

<sup>7</sup><http://lucero-project.info/>

<sup>8</sup><http://www.w3.org/wiki/ConverterToRdf>

As a key use-case, the DBpedia project has developed DBpedia Spotlight [Mendes et al., 2011] that employs the DBpedia dataset in conjunction with NLP strategies in order to associate entities with words in a text document. In this context, words are more precisely called, *surface forms*. It is highly configurable with whitelists and blacklists of types of entities obtained from the hierarchy of classes of the DBpedia Ontology, with contextual disambiguation, and a confidence score associated with each result. Other systems adopting this approach are AlchemyAPI<sup>9</sup>, which similar to DBpedia Spotlight, it finds entities in various LOD datasets and thus includes a co-reference resolution step, OpenCalais<sup>10</sup>, SemanticAPI from Ontos<sup>11</sup>,...

### 1.2.4 Web Information Retrieval

The web of documents is very important source of information. This massive collection of documents and the need for indexing and retrieving all the contained information encouraged the Information Retrieval (IR) researchers to adapt and invent new approaches able to handle the enormous amount of information on the web. It gave the birth to the “*Web Information Retrieval (WIR)*” domain.

Most conventional approaches in WIR are based on link analysis (PageRank of Google, Hits, samba, etc.), where the ranking of the retrieved documents (web pages) is based on the *incoming* and *outgoing* documents’ links. For instance, the PageRank (PR) [Page et al., 1999] was proposed by Google to rank documents using their popularity.

The explosively growing size of the web of data including more and more comprehensive information, introduced a new problem: how to find, exploit and consume this vast amount of data? Generally speaking, retrieving and ranking relevant information is a fundamental issue in the IR, but when we deal with a huge amount of data, the manner how the results are presented, filtered or ranked became more important. In the web of data context, ranking became more delicate, as the nature of the information items to be ranked and their relations are very different from the case on the web of documents.

Consequently, existing ranking algorithms have been adapted to the web of data and some others new strategies have started to be proposed and implemented.

[Roa-Valverde and Sicilia, 2014] identified several aspects related to the way ranking approaches in the web of data are designed. These aspects are imposed by the new information needs in the web of data and can be summarized as follows:

---

<sup>9</sup>[www.alchemyapi.com](http://www.alchemyapi.com)

<sup>10</sup>[www.opencalais.com](http://www.opencalais.com)

<sup>11</sup>[www.ontos.com](http://www.ontos.com)

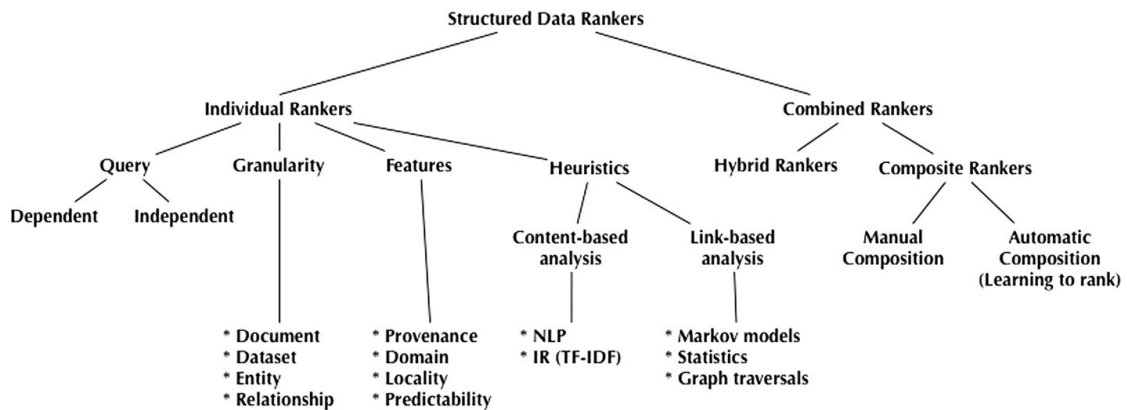


FIGURE 1.1: Structured Data Rankers ([Roa-Valverde and Sicilia, 2014]).

- Dealing with larger and heterogeneous information.
- Integrate both structured and unstructured information.
- Query execution.
- Consolidation of results.

According to these aspects, they classified the structured data rankers in a tree of classes (figure 1.1) with two main configurations: first, applying individual rankers based on a single relevance criterion. Second, combining individual rankers, either as hybrid ranker (i.e. integrate multiple ranking criteria) or as composite ranker by combining various ranking scores produced by different ranking algorithms.

As an example, Swoogle proposed by [Ding et al., 2004] is classified as [ *Individual Ranker* : *Heuristics* : *Link-based analysis* : *Markov models* ]. The authors of this work try to exploit the semantic of relationships during the ranking process, so they offer a modified version of PageRank that considers the different types of predicates between entities.

In the last years, the entity-oriented search became more and more necessary. Many commercial giants started to provide this kind of search service, such as Google Knowledge Graph, Facebook Graph, Bing Snapshot, WolframAlpha and Yandex Islands. All begun in 2010 when Yahoo! researchers discovered by analyzing their search log that there is a significant bias towards *entity-centric queries* [Pound et al., 2010], more than half of queries focus on an entity or an entity type. Besides, 14.3% of the queries contain a context entity or type. Therefore, they introduced the term “Ad-hoc Object Retrieval (AOR)” also commonly called the “*entity search*” as a task to provide a ranking of RDF entities in terms of their relevance to an entity that is explicitly named in the query. A wide range of retrieval approaches is used to achieve this task, such as using a ranking functions based on BM25MF [Campinas et al., 2012] or on a language model [Neumayer

et al., 2012], applying a supervised learning-to-rank approach [Zablith et al., 2011], or even adapting PageRank algorithm to rank entities like in [Hu and Du, 2012].

### 1.2.5 Search User Interface

At the present time, web search engines are the portals of the web. The web exploration usually begins with a query issued in a search engine with some query reformulations if needed. Web search engines take the user information need as a keyword query and try to satisfy this need by interrogating the index, retrieving the documents that are relevant to the query, then showing them in the *Search User Interface (SUI)*.

Users interact with web search engines via the SUI, the role of SUI is to help users in the expression of their information need, to understand search results and to select among the returned results. In the SUI, either the search engine shows full documents, or else the user is presented with a summary of the content of each document called the “*Textual Snippet*” [Hearst, 2011] that helps the user in making the decision about the relevance of this result. The quality of these textual snippets can significantly affect the perceived relevance of the search results, making it a critical success factor of the SUI.

The current standard results display in SUI is a vertical list of titles (along with the URL), and *Textual Snippets* referred to as *SERP (Search Engine Results Page)*.

Usually, a textual snippet is an excerpt drawn from the full text of the document, and that contains the query’s terms (often highlighted). In some cases, some metadata is also shown.

The textual snippets are used by almost every text search engine as a complement of the ranking proposed by the system (system relevance); their principal role is to economize the time and efforts needed to find the relevant document (web page) from the user point-of-view (user relevance).

To enhance the user satisfaction when interacting with a Snippet, the *query-biased snippets* are proposed. They suggest to show the query terms in the context in which they appear in the document, it improves the user’s judgment about the relevance of a result. There are a lot of other factors that also enhance the user satisfaction, like highlighting the query terms, or showing some *deep links* to parts of the document. Additionally, in some cases, user information need can be satisfied directly in the SERP, thus, transform the search engine into an “answer engine” [Nielsen, 2004].

Since the middle of the 20th century, most efforts to improve information retrieval have focused on methods of matching text representations with query representations, and



on methods of enriching the SERP. Recently, however, researchers have undertaken the task of understanding the human, or user, role in IR. The basic assumption behind these efforts is that we cannot design effective IR systems without some knowledge of how users interact with them. Therefore, this line of research that studies users in the process of directly consulting an IR system is called interactive information retrieval (IIR).

In the IIR, Multiple methods were proposed to enhance the search result, including: Query Reformulation, the complex query languages, the “Advanced” Search, Suggestions, Clustering, Categorization, Visualization and the enhanced snippets (or enhanced surrogates).

Having good snippets (or a good SERP) is considered as an important aspect in the IIR and in increasing the user satisfaction, especially to facilitate quick review of retrieval results and access to useful information. A good snippet should allow searchers to make informed decisions about the content of the object being represented, decisions like: Should I read this document? Can I safely ignore it? Is it different from these other retrieved documents?

Hass et al. introduced in [Haas et al., 2011] the notion of *enhanced search results* where they extend the SERP from a list of “ten blue links” with snippets of text, to include multimedia objects, specific key value pairs, and interactive elements. Using structured data or metadata associated with web documents they propose more relevant and more compelling representations of search results. The weakness of their work is its assumption that web pages contain metadata describing its content. The tendency to adopt largely this approach by web search engines to enhance their SERP is due to the immediate promise of this method as it is closer to the traditional expectations of search engine processing and the fact that collecting metadata embedded inside HTML pages requires minimal changes to the existing crawling infrastructure. Adopting this approach by the most popular websites is due to the fact that their business needs the intermediation of the web search engines. Therefore, they provide the meta-data for their web pages to satisfy the SE requirements to enhance their results presentation and improve their popularity. Another method that applies techniques to promote user interaction with search results is the work of [White et al., 2002].

Enhancing the SERP with the intention of improving the user satisfaction can be also achieved by changing the form of search result displays. In [Dumais et al., 2001] authors found that users perform search tasks faster if results are grouped in categories in the interface. In addition to common approaches for providing more relevant textual snippets like in [Varadarajan and Hristidis, 2005].

### 1.3 Statement of the Problem

*“A problem is a chance for you to do your best.” – Duke Ellington*

In an ideal world, when a WIR system (or a web search engine) shows the SERP to the user as an answer to her query, the user identifies easily, directly, rapidly the web pages most likely to fulfill her information need.

However, this is not the case with the current search engines (over the web of documents). In fact, the current SERPs (consisting mainly of textual snippets) have limited expressiveness of the relation between the *user information need* and the *result’s content*. In like manners, they are semantically poor with neither conceptual summary of the result, nor explanation why the user got this result.

Consequently, these SERPs are not ideal in discovering the relevance or the usefulness of the results, especially in the case of the exploratory search, where users do not have a very clear information need. In 2014, a study [Jiang et al., 2014] of the user behavior based on the searching, the clicking and the browsing suggested that:

*“Users cannot perfectly predict whether a result is useful or not purely based on the abstract returned by a search engine.”*

Moreover, as they applied the study over different types of search tasks, they found that:

*“The lower click accuracy in tasks looking for intellectual products indicates that the result abstracts provided in current search engines are probably optimized for factual search only, which is difficult to satisfy users searching for other types of information.”*

We can summarize the statement of this thesis as the following:

*“Considering the fact that the web of data became a huge store of structured and linked data, how can we combine the information from this web with the information from the web of documents to enhance the user experience when interacting with a SERP? How can we make these SERPs more expressive, semantically rich and helpful for the users?”*

In this thesis, our primary assumption is that the retrieval effectiveness and the user satisfaction on the WIR systems can be improved by semantically enhancing the SERP with the assistance of the web of data as a source of structured and linked data.

## 1.4 Research Questions

*“Never try to solve all the problems at once, make them line up for you one-by-one.” – Richard Sloma*

Solving the above problem includes addressing the following questions:

– **Question 1:** *In an IR system, can we introduce a new, interactive, richer and more informative Artifact to the SERP that replace the traditional textual snippets? Can we employ the automatic annotation of the Linked Data entities over the results’ text in the construction of this Artifact?*

The textual snippets proposed by the traditional search engines are excerpts from the documents that often contain the keywords of the user’s query highlighted in their context. They are semantically poor because of the lack of, or not existing semantic representation of the document, the query, and the relation between them. They are also non-interactive artifacts.

Replacing these snippets by new *interactive* artifacts can help the users in organizing the information, structuring their investigation of an information resource, or in making a decision.

The new artifact must be richer and more expressive in reflecting not only the context of the query’s keywords but also where and how the user information need is linked to the most relevant information in the document.

As we saw (Section 1.2), the *Automatic Annotation* is an important method to interconnect the web of documents and the web of data. Applying the automatic annotation over the text of a search result will return a huge set of annotated entities. Can we use these entities in the construction of the new artifact? If yes, this will lead us to the next question.

– **Question 2:** *As the automatic annotation allows the identification of an immense amount of entities in the text, how can we choose the most relevant ones? In other words, how to overcome the problem of the noisiness and off-topic of the annotations? And how to select the relevant entities according to the original query?*

The annotated entities have different levels of relevance (pertinence to the information need) and usefulness (utility in a given context). Even with latest technologies in this domain, we still have annotation errors (i.e., link a word or a surface-form to the wrong entity), and off-topic errors (i.e., correctly annotated but the entity is not related to the context), thus introduce a considerable quantity of noise in the extracted data.

For these reasons, ranking and selecting the most relevant entities (extracted from the web of data) is a crucial aspect to be able to use them in the proposed artifact. Further aspects to examine are: how to identify the relevance from the user point of view?

– **Question 3:** *How to objectively measure the impact of the new proposed artifact on the user experience?*

Measuring if the new interactive artifact has enhanced the user experience or not is crucial. This evaluation must be generic and representative enough to cover all the aspects of presenting and interacting with the new artifact.

## 1.5 Application Scenarios

*“If I’m going to sing like someone else, then I do not need to sing at all.” – Billie Holiday (American jazz singer)*

In the following, we present some application scenarios that illustrate the need for the work developed during this thesis.

### 1.5.1 Application Scenario 1: General Vs. Specific user information need

Sam has a homework to do about “dinosaurs’ communications”, he does not know a lot about dinosaurs, and nothing about their communication, so he decided to do some research on the web. He has the possibility to search about the “dinosaurs”, then about “dinosaurs communications”, or search directly about “dinosaurs communications”.

– In the first scenario, he starts the search session by typing “dinosaurs” in a search engine like Google, he hopes to find a summary about dinosaurs to have a general idea about them. However, as this query is too generic, the search engine gives back many documents and information causing an information overload in the SERP (see Figure 1.2). As a consequence, Sam must read all the snippets in the SERP and even read too many pages in order to build his summary.

We have to notice here, that this type of summary is typically the abstract (short text that describes the entity) of an entity (like dinosaur<sup>12</sup>) in the web of data.

The next step is to reformulate the query by adding the keyword “communications” (see Figure 1.3), then to decide which results are the best from his point-of-view (POV). This decision depends only on the small text associated with each result (the textual snippet),

---

<sup>12</sup> <http://live.dbpedia.org/resource/Dinosaur>

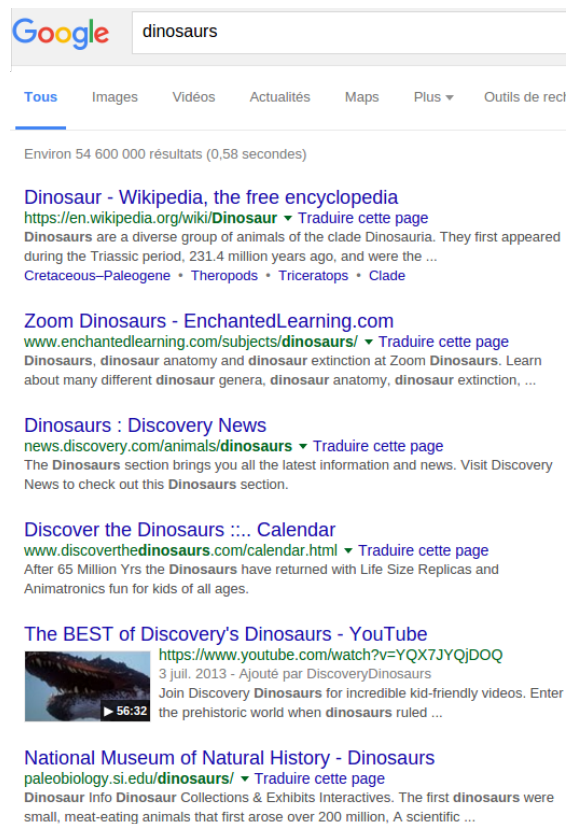


FIGURE 1.2: Google results for “dinosaurs” query

hoping that these results contain what he was searching for, i.e. an explanation about dinosaurs’ communications.

– In the second scenario, Sam starts the search session directly with the query “dinosaurs communications” (see Figure 1.3). The problem, in this case, is that he will suffer from what we call the information limitation, i.e. the query is specific enough to limit the retrieved information only on the query’s topic. So, he will get results about how the dinosaurs communicate, but no general information about the dinosaurs, so for each result he must find some general information about dinosaurs (that may not be included, such as the first result<sup>13</sup>) then about how they communicate.

### 1.5.1.1 A solution

As a solution, our proposed system ENsEN (Enhanced Search Engine) is able to find, the most important concepts in a result (regarding the query), associating each concept with a textual and factual summary built by combining the result’s content with information from other trusted, external sources.

<sup>13</sup><http://www.livescience.com/32271-how-did-dinosaurs-communicate.html>

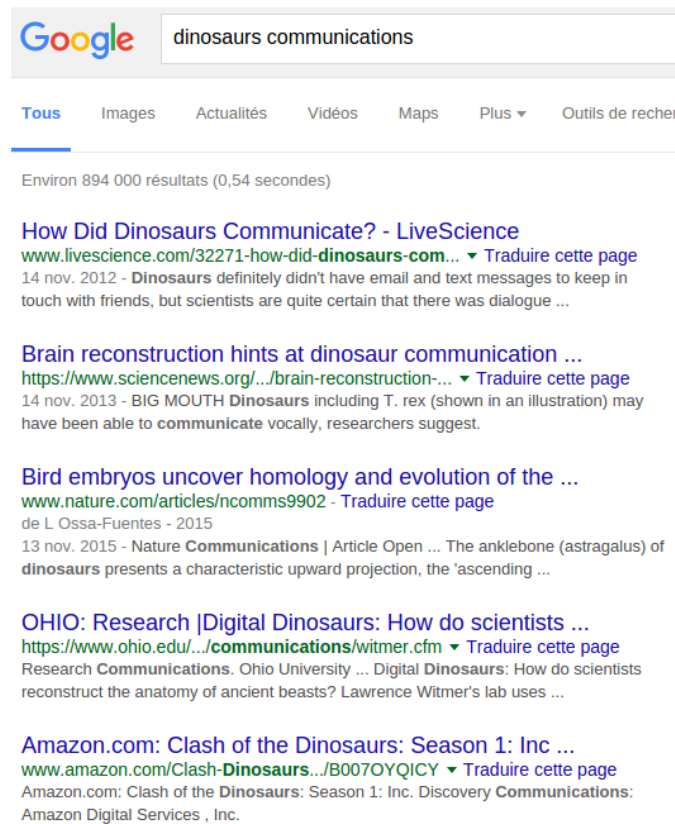


FIGURE 1.3: Google results for “dinosaurs communications” query

Returning to our scenario, Sam types “dinosaurs communications” in this search engine, he will get results talking about dinosaurs communications, see Figure 1.4. However, in the results, “dinosaurs” and “communications” are considered as important concepts. Therefore, ENsEN includes them in the generated snippets with summaries that Sam needed to understand the subject and satisfy his information need. In addition, ENsEN provides a list of the most important concepts related to the last two concepts, with a textual context that explains how these two concepts are related and where (on the result page). In this SERP, Sam will get the definition of dinosaurs (a) and communication (b). He will get also a list of top related concepts to his query (c), such as “Bellows”, “Guttural”, and “Corythosaurus”, to enrich his knowledge of the subject. In addition, the system shows that the excerpt (d) “*The chambered headcrests on some dinosaurs such as **Corythosaurus** and **Parasaurolophus** might have been used to amplify grunts or bellows.*” is one of the best to contextualize the query, however, the term “communication” is not present in this excerpt.

Finally, by knowing the most important concepts (regarding the query) in each result (e), it will be easier for Sam to identify the most relevant result, economizing a lot of time and efforts.

The screenshot shows the ENsEN search interface for the query "dinosaurs communications". The search bar at the top contains the query and a "Search" button. The left sidebar features a "Semantic filter" with a "Dinosaur" category selected (marked with a yellow circle 'c'). Below it is a "Textual Summary" section with a "show" button. The main content area displays three search results:

- Did Dinosaurs Communicate? - USGS:** A result snippet mentioning "The chambered headcrests on some dinosaurs such as Corythosaurus and Parasaurolophus might have been used to amplify grunts or bellows." Below the snippet are several green tags: Dinosaur, Bellows, Guttural, Courtship, Triceratops, Parasaurolophus, Corythosa..., and Communic... (marked with a yellow circle 'b').
- Brain reconstruction hints at dinosaur communication | Science News:** A result snippet dated "14 Nov 2013" mentioning "BIG MOUTH Dinosaurs including T. rex (shown in an illustration) may have been able to communicate vocally, researchers suggest." Below the snippet are several green tags: Dinosaur, Sofia, Bird, Human brain, Crocodile, Duke Univ..., Reading (p...), Shutterstock, Big Brother..., Brainpower, Science N..., and University... (marked with a yellow circle 'e').
- Did dinosaurs communicate with one another? - YouTube:** A result snippet dated "11 juin 2015" mentioning "Ajouté par Petrosains Listening Petrosains Jurassic Adventure Petrosains Sdn Bhd." Below the snippet are several green tags: Dinosaur, French lan..., Google, Jurassic, YouTube, Science, Signaller, Insanity, Lien, Spider-Man, Thor, Avis Rent..., Abselling, Adventure..., Earth, Kuala Lum..., Navigation, and Transcrip... (marked with a yellow circle 'e').

On the right side, there is a "Dinosaur" category section (marked with a yellow circle 'a') containing a "Did dinosaurs communicate? - Fact Monster" result (marked with a yellow circle 'd'). This result includes a snippet and a "Related Concepts" section with "Bellows" and "Guttural" (marked with a yellow circle 'd').

FIGURE 1.4: ENsEN results for “dinosaurs communications” query

### 1.5.2 Application Scenario 2: Answering directly (in SERP) hard questions

Sam’s professor asked him to search (on the web) an answer to the question “what school produced the most justices for Supreme Court justices?”.

Sam does not even know what is the “Supreme Court justices”. Therefore, he started his search session by typing “school produced the most justices for Supreme Court justices” in Google. Google returned the relevant results in SERP with some snippets (see Figure 1.5).

the school produced the most justices for Supreme Court justices|

Tous Actualités Images Shopping Vidéos Plus ▼ Outils de rec

Environ 17 800 000 résultats (0,65 secondes)

**List of law schools attended by United States Supreme ...**  
[https://en.wikipedia.org/.../List\\_of\\_law\\_schools\\_atten...](https://en.wikipedia.org/.../List_of_law_schools_atten...) ▼ Traduire cette page  
List of law schools attended by United States Supreme Court Justices ... advent of modern law schools in the United States, justices, like most attorneys of the ... In total, of the 112 Justices appointed to the Court, 46 have had law degrees, an ...

**The Most Popular Law Schools of Supreme Court Justices ...**  
<time.com/.../the-most-popular-law-schools-of-suprem...> ▼ Traduire cette page  
8 mai 2014 - In all, 48 Supreme Court Justices of the United States successfully graduated from law school. The ones that produced the most justices are ...

**Law Schools That Produced The Most Supreme Court Justices**  
<www.huffingtonpost.com/.../law-schools-supreme-co...> ▼ Traduire cette page  
Law Schools That Produced The Most Supreme Court Justices. 05/08/2014 03:23 pm ET | Updated May 08, 2014. Tyler Kingkade Senior Editor/Reporter, The ...

**Frequently Asked Questions on Justices - Supreme Court of ...**  
[www.supremecourt.gov/faq\\_justices.aspx](www.supremecourt.gov/faq_justices.aspx) ▼ Traduire cette page  
Have any Supreme Court Justices had the same name? Have any Supreme Court ... What law schools did the present Justices graduate from? Why do Justices ...

**Top Law Schools For Supreme Court Justices - NerdWallet**  
<www.nerdwallet.com/.../top-law-schools-supreme-cou...> ▼ Traduire cette page  
26 sept. 2012 - So that begs the question: Which law schools have produced the most Supreme Court Justices? We compiled information from the Federal ...

**US|Three Supreme Court Justices Return to Yale - The New ...**  
<www.nytimes.com/.../three-supreme-court-justices-ret...> ▼ Traduire cette page  
25 oct. 2014 - The justices returned to Yale Law School on Saturday for reunion weekend, and they ... Justice Thomas, 66, has apparently mellowed the most. ... He had, he wrote, "learned the hard way that a law degree from Yale meant one ... Justices Alito and Sotomayor suggested that the Supreme Court may be too ...

FIGURE 1.5: Google results for “school produced the most justices for Supreme Court justices” query

Sam reads these snippets hoping to find the answer, but, unfortunately, no answer there, so he made a decision about the most relevant result (from his POV) and clicked on it. So, he had two expensive solutions: either he reads the whole page to find the answer, or he uses the browser search-in-the-page functionality with keywords like “Supreme Court justices”, “most”, “school”. It is possible that Sam does not find the answer on this page. In this case, he must go back to the SERP and visit another page, or he can re-formulate the query to get different results.



FIGURE 1.6: ENsEN results for “school produced the most justices for Supreme Court justices” query

### 1.5.2.1 A solution

Our system (ENsEN) will return the SERP presented in Figure 1.6, in which, Sam is able to find the most important concepts in each result with respect to the query, each concept is associated with a textual context also relevant to the user’s query. In this SERP, ENsEN shows that “Supreme Court of the United States” is the most important concept, giving Sam the definition of this term, and its most related concepts (Figure 1.7).

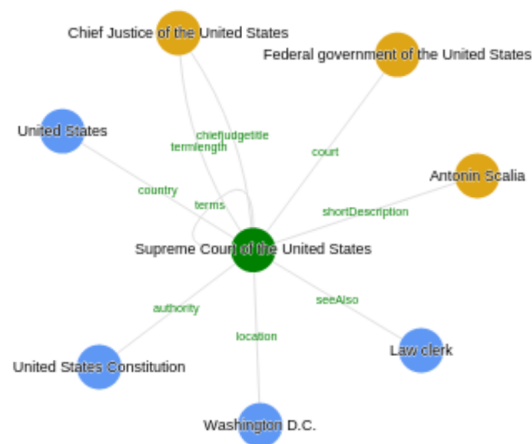


FIGURE 1.7: Most related concepts of “Supreme Court of the United States”


ENsEN shows also that the next two most important concepts are “United States” and “Harvard Law School”, so *Harvard* may be the answer to Sam’s question.

Actually, in the context of the concept “Harvard Law School” (Figure 1.8), Sam will find the needed answer:

*“The ones that produced the most justices are Harvard(15), Yale(6), and Columbia(2).”*

Not only the answer but also in which result this answer is, i.e. *“The Most Popular Law Schools of Supreme Court Justices — TIME”*, with the possibility to go directly to the corresponding paragraph in the result.

**Harvard Law School** W



**Harvard Law School** (also known as Harvard Law or HLS) is one of the professional graduate schools of Harvard University, located in Cambridge, Massachusetts. Founded in 1817, it is the oldest continually-operating law school in the United States and is home to the largest academic law library in the...

**Context**

[List of law schools attended by United States Supreme Court Justices](#)  
 Curtis , who received his law degree from **Harvard Law School** in 1832, and was appointed to the **Court** [...]

**The Most Popular Law Schools of Supreme Court Justices | TIME**  
 1832, when Benjamin Robbins graduated from **Harvard University**, that the first justice-to-be obtained a[...]

The ones that produced the most justices are **Harvard** (15), **Yale** (6), and **Columbia** (2). [...]

**Frequently Asked Questions on Justices - Supreme Court of the ...**  
 Kennedy - **Harvard** (LL.B) Justice **Clarence Thomas** - **Yale** (J.D.) [...]  
 Breyer - **Harvard** (LL.B) Justice **Samuel A.** [...]

FIGURE 1.8: Entity description of “Harvard Law School”

In addition, ENsEN’s SERP will give Sam enough elements to explore more information about the subject, such as Harvard’s most related concepts (Figure 1.9), how they are related, in which context the query concepts were mentioned.

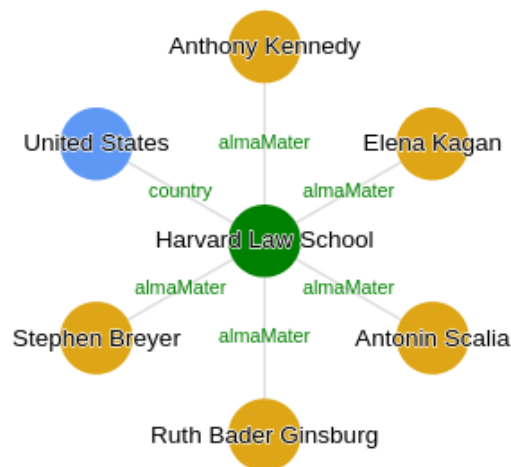


FIGURE 1.9: Most related concepts of “Harvard Law School”

To summarize, ENsEN will give Sam the answer for his question directly in the SERP. So, no need to visit or search again in the results, and it will give him the needed means to explore more about his information need.

## 1.6 Summary of Contributions

*“Each of us is a unique strand in the intricate web of life and here to make a contribution.” – Deepak Chopra*

In this thesis, we propose a novel interactive artifact for the SERP called the “Semantic Snippet”. The semantic snippets facilitate the knowledge transfer and acquisition. It is a semantic entity-based artifact, and it tries to assist the user in her conceptualization process during a search session.

Our proposed approach to solving the problems mentioned above can be decomposed into the following contributions:

### 1.6.1 Query-biased Ranking for Linked Data Entities

The result of applying the automatic annotation on a text is a vast list of annotated entities with a lot of noisy and off-topic ones. Ranking these entities and selecting the most relevant ones with respect to the user information need (second research question) is a crucial issue. Therefore, we propose a query-biased algorithm (LDRANK) for ranking

the entities of an RDF graph. This algorithm is well adapted to our context, where we suppose that the entities were discovered in a web page found by a web search engine as an answer to a user's query.

**LDRANK (Linked Data Rank)** is an algorithm to rank the entities of poorly connected graphs for which textual data can be associated with the nodes. It uses the explicit structure of the graph through a PageRank-like algorithm and the implicit relationships that can be inferred from the text associated with the entities through an original variation of the Singular Value Decomposition (SVD).

### 1.6.2 “Semantic Snippets” and the Enhanced Search Engine (ENsEN)

In this contribution, we introduce our new semantic and interactive artifact of the SERP called “*semantic snippets*”. This new artifact replaces the traditional textual snippets in the search interface.

The semantic snippets focus on LD entities integration to enhance the knowledge transfer process. It tries to employ the Linked Data from the highly connected Open Data (LOD) graph to contextualize the user information need and even contextualize the results; this contextualization improve (as proved in the experiments) the usefulness of the results for the user.



FIGURE 1.10: Elements of a semantic snippet.

The semantic snippet that we propose consists of five elements illustrated in Figure 1.10:

- (1) *the result's title*;
- (2) *the original textual snippet* proposed by the search engine (excerpt from the result) but annotated with entities from the web of data;
- (3) *the new textual snippet*, it is also an excerpt from the result but it was selected using machine learning methods with semantic and textual features;
- (4) *the primary concepts*, best entities annotated in the result that represent the best the relation between the “*user information need*” and the “*result's content*”. The analysis and the selection of the primary concepts is presented in the first contribution (Chapter 2);
- (5) *the entity description* (Figure 1.11), for each primary concept, the semantic snippet shows an info panel called the *entity description*. This panel aims to explain the entity and contextualize it in the text and among the other entities. It consists of (a) a short textual description (abstract), (b) a picture, (c) a textual context section (best excerpts where it was mentioned), (d) related concepts section (which entities it is related to) and finally (e) a graphical representation of the RDF subgraph centered around this entity.

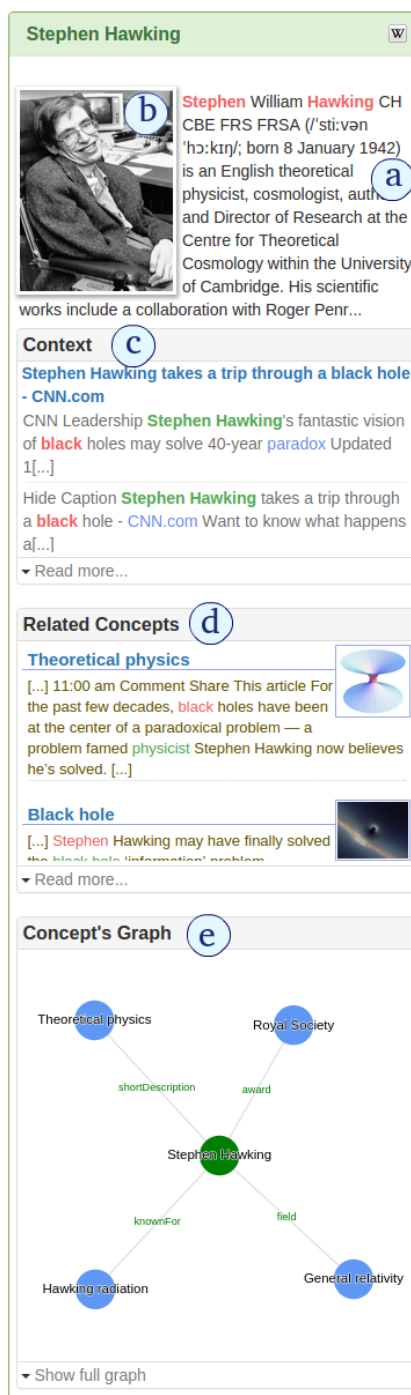


FIGURE 1.11: Elements of a Entity description

In order to better convince the reader of the usefulness and efficiency of the proposed artifacts, we integrate them at the core of *ENsEN*<sup>14</sup> (*Enhanced Search Engine*): a software system that enhances a SERP with semantic snippets (presented in chapter 4). Given the query, we obtain the SERP (we used Google in the experiments). For each result of the SERP, we build the proposed data structure that we use as input to

<sup>14</sup>ENsEN: a live demonstration is available on-line <http://liris.cnrs.fr/drim/projects/ensen/>

execute LDRANK (and LDSVD) to obtain a ranking of the entities. The top-ranked entities will be used in the *Primary Concepts* component. From an SPARQL endpoint, we do a 1-hop extension of the top-ranked entities in order to increase the number of triples among which we will then search for the more important ones. To do this, we build a 3-way tensor from the extended graph that we analyze in order to associate to each top-ranked entity a set of triples that will appear within its description (*Entity Description*).

Finally, we used a machine learning approach (see Section 3.6) to select an excerpt from the web page to be the new textual snippet and short excerpts to be part of the description of each top-ranked entity.

In this approach, we introduce how to use the machine learning to rank and select the documents' excerpts. We are interested in documents, which are the results of a query issued to an information retrieval system. Each of these documents come with a list of annotated entities founded by DBpedia spotlight and a corresponding RDF graph extracted from Linked Data.

This approach is a supervised learning one, where we learn to select documents' excerpts that expose the relevance of an entity (from the most important entities in the document that has been selected using LDRANK) to the user's information need. We propose a classifier that is able to predicate if a sentence (excerpt) expose the relation (Information need, Entity, Document).

Different classifiers were trained using the training set, and their prediction performance was evaluated using the cross-validation. A feature selection study also has been applied to enhanced the prediction and the performance.

## 1.7 Structure of the Thesis

This thesis is organized in five chapters. In the current chapter (Chapter 1), we introduce the context of our thesis, some background information and the research questions that we try to answer.

The Chapter 2 describes our first contribution, the entity ranking algorithm (LDRANK). The proposed interactive artifact (the semantic snippet) is presented in Chapter 3, where we also discuss the machine learning approach to select the textual snippets (Section 3.6). The developed system (ENsEN) is presented in Chapter 4.

As we handle different domains in this thesis, there is no *Related Work* chapter. We chose to discuss the corresponding related works with each contribution.

Each chapter starts with an introduction, which motivates the corresponding research questions by referring to related work, and concludes with a summary of main findings and contributions.

Finally, we conclude this thesis (Chapter 5) by summarizing our main findings and contributions and answering the research questions raised in this chapter. Further, we outline future works made possible by the findings of this thesis in order to allow different professionals to continue research in this direction.



## Chapter 2

# A Query-biased Ranking for LOD Entities

## 2.1 Introduction and Context

In this contribution, we introduce LDRANK, a query-biased algorithm for ranking the entities of a sub-graph of the Linked Open Data (LOD) cloud<sup>1</sup>. LDRANK makes use of any available textual data that can be associated with the nodes of the graph. We apply LDRANK to graphs derived from the automatic detection of LOD entities within the top web pages returned by a web search engine given a user's information need expressed through a query made of a set of keywords.

As to this matter, numerous solutions exist for the automatic detection of LOD entities in plain text, such as DBpedia Spotlight [Mendes et al., 2011] (configurable via white and black lists to filter by types of entities as defined in the class hierarchy of the DBpedia ontology, including a contextual disambiguation phase, and providing a confidence score for each association of a surface form with a potential entity) ; AlchemyAPI<sup>2</sup> (similar to DBpedia Spotlight, it makes use of multiple LOD datasets and therefore it possesses a co-reference resolution step), OpenCalais<sup>2</sup>, SemanticAPI de Ontos<sup>2</sup>, etc.

In our context, we use DBpedia Spotlight, and the textual data associated with the nodes of the graph comes from the DBpedia's abstract of the entity, and also from excerpts of the web page centered on the surface forms where the entity was detected.

To point towards the specificity of the LDRANK algorithm, we follow with a brief description of the relationship between conventional approaches for the ranking of web pages and state of the art proposals for the ranking of the web of data's entities.

In the context of the *web of documents*, a hyperlink indicates a relationship between information carried by two web pages. Although these indications are in general coarse-grained, they revealed themselves as essential for the most-effective algorithms used to rank web pages PageRank [Page et al., 1999], HITS [Kleinberg, 1999], SALSA [Lempel and Moran, 2001].

In the *web of data* context, the links (materialized by triples: subject / predicate / object) represent named relationships between data entities. These links make a web of data from interrelated datasets: the Linked Data cloud. The vast majority of ranking strategies for the web of data (see the next section) relies on adaptations of the PageRank [Page et al., 1999] algorithm.

With LDRANK, we adopt a strategy based on the linear consensual combination of several sources of prior knowledge to modify a PageRank-like algorithm. Here, *prior knowledge* means any form of contextual knowledge apart from the structure of the

---

<sup>1</sup><http://lod-cloud.net/>

<sup>2</sup>[www.alchemyapi.com](http://www.alchemyapi.com) ; [www.opencalais.com](http://www.opencalais.com) ; [www.ontos.com](http://www.ontos.com)

RDF graph. For example, textual descriptions for the nodes of the graph would qualify as a source of prior knowledge. Each such source is used to compute a probability distribution over the entities. Such a distribution gives a prior importance to each entity.

In the next section 2.2, we present the related works about *Semantic Search* and *Entity Ranking*. Then, we study these related works using a comparison schema (consisting of a set of dimensions and classes) designed after studying the latest works and surveys in the domain. This schema helps us to position LDRANK against the most recent approaches. Next we present in details our approach, and its evaluation.

## 2.2 Related Works

Search technologies are today very much interested in the *Semantic search*, where one of the main tasks is to rank linked data entities or other semantic web resources. Semantic search aims to enhance and enrich the traditional information retrieval methods by going beyond keywords matching, and simple link analysis.

Semantic search becomes a very dynamic research area thanks to the emergence of the web of data, and the metadata embedded in the web of documents. The analysis of these embedded data allows discovering more relationships between the embedded entities. These relations were not explicit in the text, or not in a machine readable format. However, this type of search raises new challenges, such as: (1) *How to modelize the query*: in the semantic search, the query can be more than a list of keywords, it can be expressed as a set of entities or even as a structured query. (2) *What is the definition of a “document”*: in a semantic search system, the document can take multiple forms, such as semantic resource, physical file, entity and its aggregated information, the result of a SPARQL query. (3) *The ranking approach*.

In this work, we are interested in answering the last question, viz. how to rank the semantic resources? In particular (and according to our context), how to rank the annotated entities according to the user information need of which we have a partial knowledge through the user query.

Entity ranking is the process of finding and sorting entities according to their relevance to the user’s information need. It is a data oriented search approach where results can come from combining information from multiple sources instead of a single one. This research domain is at the frontier of two communities: information retrieval and semantic web.

Lastly, two extensive surveys, [Roa-Valverde and Sicilia, 2014] and [Jindal et al., 2014], focused on the semantic search domain. In the first survey [Roa-Valverde and Sicilia, 2014], the authors formalize the problem of ranking linked data, they provide a complete overview of ranking methodologies on the web of data. They select the most relevant algorithms according to their impact in this field and describe them in details. They describe the challenges of ranking on the web of data, the rank operator that reflect the strategy of relevance, and how the different approaches (or implementations of the rank operator) can be classified. They identify two main configurations for choosing a *Ranker* (ranking algorithm), *Individual rankers* based on a single relevance criterion, and *Combined rankers* where multiple ranking criteria are integrated into a *hybrid ranker*, or multiple ranking scores are combined into a *composite ranker*.

In order to classify the individual rankers, the authors identified four dimensions or aspects:

- (1) **Queries:** Ranking approaches consider user queries in the computing of the scores, or not;
- (2) **Features:** A Feature is an aspect that exists in the data that makes possible to establish a comparison between items belonging to a data source;
- (3) **Granularity:** It refers to the granularity of the data source to be ranked (i.e., entity, identifier, relationships, dataset, document);
- (4) **Heuristics:** A heuristic makes reference to a particular mechanism that guide the score computation process.

The second survey [Jindal et al., 2014] classifies the ranking approaches for the Semantic search on the web into three categories considering their stage of ranking:

- (1) *Entity ranking:* In terms of entity-oriented search, where the objective is to retrieve entities that match the query.
- (2) *Relationship ranking:* That determines a relative importance of relationships found with respect to a user's query context.
- (3) *Semantic document ranking:* In terms of document-oriented search, ranking the documents according to the presence of the most relevant entities along with the most relevant relationships among those entities.

We notice that the second survey [Jindal et al., 2014] studies only one of the dimensions proposed by the first survey [Roa-Valverde and Sicilia, 2014] which is the *Ranking granularity*. However, it provides a deep discussion of this dimension, in addition to studying more works in the corresponding domain.

### 2.2.1 Comparative study

Given these two surveys, complemented by other works, [Butt et al., 2015], [Hildebrand et al., 2007], [Koumenides and Shadbolt, 2014], and [Yumusak et al., 2014], we propose the following comparison schema, presented in Table 2.1, where we define the primary dimensions of a semantic ranking approach, to position the main works in this domain. We do not present or study the domain in an exhaustive way; we rather provide a detailed outline and reflective examples from the literature.

#### 2.2.1.1 Studied dimensions

The dimensions that we propose in order to study and compare the works are the following:

1. **Query dependency**

Query dependency makes reference to the way in which the user query is considered in the ranking. An algorithm is “query-independent” (also known as “static”, “absolute” or “global”) if it ranks the items regardless of the user query, it relies on the internal structure of the dataset. These approaches are implemented on the complete dataset irrespective of the query.

On the opposite, a “query-dependent” algorithm (also called “dynamic” or “focused”) ranks the items regarding the user input. Often, in the query-dependent approaches, the ranking model is applied only to the retrieved set of items; thus it gives a **relative** order for these items.

It is possible to find algorithms that combine both static and dynamic strategies as in [Anyanwu et al., 2005].

2. **Granularity** One of the most important aspects when ranking information is the nature of the items to rank, i.e. the granularity of what is being ranked, it determines the amount of information represented by each item, like a document or an individual entity.

We use the following scale of granularities:

- (1) *Document*: It refers to the “semantic web document (SWD)” which is an atomic data transfer packet on the semantic web. It is both a web page addressable by a URL and an RDF graph containing semantic web data. Its content is either an ontology (schema or model) or a conventional RDF data. The document-centric approaches list the URIs or the labels of the matched documents and/or parts of document in order of their relevance.

(2) *Entity*: An entity is a self-contained unit of information that has relationships with other entities [Delbru et al., 2010], such as a person, a place or an event. The entity-centric approaches consolidate available data about the entity from multiple documents and list them as a profile of the entity.

(2) *Relationship*: An RDF predicate defines a typed relationship between two entities. The relationship-centric approaches find most relevant relationships between input entities. In this kind of approaches, using structured queries is a very common practice.

### 3. Scope of the dataset

The Semantic search techniques can be classified into those that explore a schema defined by an ontology, and those that explore data generated according to this schema.

The “ontology-search techniques” find the classes and properties within *specific ontology* or across ontologies (*non-specific ontology*). However the “linked-data-search techniques” focus on the retrieval of entities (or relationships among entities) in knowledge bases (KBs). These KBs can be restricted to one domain (*Domain-based KB*), or generic one (*Cross-domain KB*). Some approaches search only in the *LOD* cloud (which is a cross-domain KB).

In addition to the KBs, the “linked-data-search techniques” also search for entities in the meta-data embedded in a web document (*Embedded RDF*) or in an RDF graph which is the result of an automatic annotation process (*AA-RDF*). In both cases, the entities can belong to a domain-based KB, a cross-domain KB or even to the *LOD* cloud.

### 4. Domain

Some semantic search techniques are *specific* to a particular domain. Other techniques stay generic: they can apply to any dataset *independently* of its domain.

### 5. Algorithm’s type

The ranking algorithms follow in general three schools, (1) link-based analysis, (2) learning to rank analysis, or (3) combination of measures such as graph theory measures (e.g. centrality, Density, etc.), entropy (e.g. information gain), semantic similarity, etc.

Link analysis was proposed as a method to rank hypertext documents considering the hyperlinks to incorporate structural features during the ranking. Semantic search techniques adapted these algorithms to semantic structure.

Most of the approaches in this category are PageRank-based, they consider mainly the popularity of an item in the dataset proposing to modify the original algorithm

using different sources of knowledge. Some ranking strategies, analyze only the result data (that matches the query introduced by the user), we class them under the “*Dynamic Modified PageRank*” category. Authority-based approaches also exist. The authority is a measure of trustworthiness and it is used mainly in *HITS* [Kleinberg, 1999], and its variations.

A recent trend for semantic ranking is the adaptation of *learning-to-rank* approaches from the machine learning domain. They integrate the semantic features in the learning process.

Some techniques mix the last approaches with other measures, such as the informativeness, the coverage and the user feedback. They propose to combine them in different ways, and often with different weights (*Weighted Combination of measures*).

*The informativeness* represents the amount of information carried by each item in the dataset, that helps to identify and rank it. For example, we can think of the entropy [Shannon, 2001].

*The coverage* depends on the query and measures how much of a query’s terms or structure is covered by an item. The coverage-based techniques apply some adapted versions of the Vector Space Model and BM25, that calculate the similarity between a query and the matched item.

The user feedback is also considered as a ranking factor, using the view count or the query log.

In the context of ranking relationships, the classical information retrieval techniques are not applicable as (in contrast to the case of documents or entities) there is no way to determine how good a query matches a relationship. To overcome this problem, information theory techniques were proposed relying on how predictable a result might be for users, forming the “*Discovery search*” approaches.

Other techniques were also proposed, such as the ones based on the *language models (LM)*.

In the next Table (2.1), we show a total of 17 ranking approaches, and we compare them by focusing on the dimensions mentioned above.

Dimension	Class	OntologyRank	PopRank	SemRank	ReConRank	AKTiveRank	NAGA	Hart et al.	TripleRank	RareRank	DBpediaRanker	DING	Tanon et al.	Sigma	Ranking Complex Relations	Dali and Fortuna	DWRank	RankProperties
Query Dependency	Query independent	x	x	x			x	x	x	x	x	x	x					
	Query dependent			x	x	x								x	x	x	x	x
Granularity	Document	x																
	Entity (Resource)		x		x		x	x	x	x	x	x	x	x		x	x	
	Relationship			x			x		x			x	x	x				x
Scope of the Dataset	Specific Ontology					x				x					x		x	
	Non-specific Ontology	x																
	Domain-based KB		x							x								
	Cross-domain KB	x		x	x		x	x	x		x	x	x	x		x		x
	LOD								x		x		x					x
	Embedded RDF				x						x		x					x
Domain	AA-RDF																	
	Specific		x							x								
Algorithm's type	Independent	x		x	x		x	x				x				x	x	x
	Modified PageRank	x	x				x	x		x		x						
	Dynamic Modified PR				x													
	Discovery Search			x														
	Weighted Combination of measures			x		x					x	x	x	x	x			
	Hits based													x				
	Learn to Rank			x				x	x							x	x	x
Language model (LM)						x												

TABLE 2.1: Comparative study: summary of dimensions and approaches



### 2.2.1.2 Analysis

Table 2.1 shows that the studied approaches include different ranking criteria, like link analysis, learning-to-rank... Most of the Semantic search approaches are query independent and targeting entities (and sometimes their relationships). In general, the studied works rank items of cross-domain Linked Data datasets, and when an approach rank ontology's items, the used ontology is often a specific one.

From an algorithmic point of view, most of the approaches are based on a modification of PageRank. These modifications propose (generally) to replace the random component of PageRank represented by the uniformed teleportation matrix (all the nodes have the same probability to be attained through a teleportation) by a biased teleportation matrix representing some prior knowledge about the importance of the nodes.

In the case of *OntologyRank*, they propose to use the different types of links between the semantic documents as prior knowledge in order to modify the PageRank. In the same way, *PopRank* proposes to consider the link types by integrating different weights (according to the type), which they named the “popularity propagation factors”. As *RareRank* handles domain-specific dataset, they propose a non-random surfer model, or what they call a *rational searcher* guided by link analysis (e.g the citation), and content analysis (e.g. indirect links between documents with similar or closely related topics). *DING* also assign weights to links and applies PageRank to the weighted dataset graph.

The previous approaches are query-independent, and the application of the ranking algorithm is off-line on the whole dataset. However, other approaches, such as *ReConRank*, propose to apply the modified PageRank only on the result set, so they send the query, retrieve the resulted items, and then they apply the algorithm. We call this kind of algorithms “Dynamic modified PageRank”.

Similar approaches were proposed but based on the other famous link-analysis algorithm Hits, like *TripleRank*, where the authors represent the triples in a tensor, and then they apply the PARAFAC decomposition, which can be seen as a multi-modal counterpart to web authority ranking with HITS.

Another category is the learning-to-rank approaches applied to the web of data (e.g. [Dali et al., 2012]). These approaches rely on the availability of relevance judgments for learning.

Some other approaches tried to merge multiple measures and methods in order to compute the item's score. They often use a weighted combination of these measures. The combined measures can be structural like in *AKTIVERank* (class match, density, semantic similarity, betweenness), others approaches (e.g. *SemRank*) used measures from

the information theory that rely on the predictability of a result (based on the information gain and the likelihood that a user could have guessed that such a result exists). A set of similarity measures (resource similarity, resource label-abstract similarity) are combined in *DBpediaRanker* to find the relevance with respect to the user query.

Studying the last six dimensions allows us also to distinguish and choose the approaches, the most related to our context. As aforementioned, our context is the query-based ranking of entities in an RDF graph, generated by an automatic annotation process over a web page returned by a search engine given a user's query. These entities belong to the LOD graph, which is a cross-domain knowledge base. Besides, these annotated entities are domain-independent. LDRANK proposes a dynamic modification of PageRank since it considers the query in the ranking, see Table 2.3 for the LDRANK positioning.

Consequently, we identified five approaches as similar, depending on their shared dimension with LDRANK.

## 2.2.2 Selected approaches from the literature

The identified related approaches (and LDRANK) are PageRank-based; thus, we present a short background about link analysis and PageRank, and then we discuss in details the selected approaches. Finally, we compare ourselves to them.

### 2.2.2.1 Background

In this background, we recall the main topics related to the link analysis and PageRank.

#### Link analysis

Link analysis refers to the techniques that handle the ranking problems by exploiting the link structures. These links may be between actors in social networks, citation links in scholarly publications or hyperlinks among web pages.

These techniques become very successful for locating authoritative and quality documents thanks to their reasonable assumptions, and superior performances. Since 1998, two ranking techniques, PageRank [Page et al., 1999] and HITS [Kleinberg, 1999] became the center of interest in IR. The PageRank algorithm is the technology proposed by Google that dominates the world of search engines. The HITS algorithm has been used to identify authority and hub web pages on the web.

### Original PageRank

Before presenting the original PageRank model, we must mention some mathematical (and graph theory) definitions:

**The Markov Chain** (see Chapter four in [Langville and Meyer, 2011]): It is a random process that undergoes transitions from one state to another on a state space (Figure 2.1). It must possess a property that the probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it. A Markov chain is characterized by a stochastic matrix, see Table 2.2.

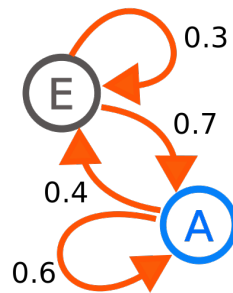


FIGURE 2.1: A simple two-state Markov chain.

	A	E	$\Sigma$
A	0.6	0.4	1
E	0.7	0.3	1

TABLE 2.2: The stochastic matrix of the last example

**A nonnegative matrix:** a nonnegative matrix is a matrix in which all the elements are equal to or greater than zero.

**A stochastic matrix:** it is a square matrix with each row consisting of non-negative real numbers whose sum is unity. A key property of a stochastic matrix is that it has a principal left eigenvector corresponding to its largest eigenvalue 1.

**An irreducible adjacency matrix:** An adjacency matrix of a directed graph is irreducible if and only if such directed graph is strongly connected, i.e. every node is reachable from every other node.

**A primitive matrix:** A square stochastic matrix  $A$  is primitive if it is non-negative and its  $n^{th}$  power is positive for some natural number  $n$ . In other words, for some  $n \geq 1$  the matrix  $A^n$  has no entries equal to 0.

**An aperiodic chain:** An aperiodic chain is an irreducible Markov chain with a primitive transition matrix.

**The power iteration method:** the power iteration is an eigenvalue iterative algorithm, where given a matrix  $A$ , the algorithm will produce a number  $\lambda$  (the eigenvalue) and a nonzero vector  $v$  (the eigenvector), such that  $Av = \lambda v$ .

The original algorithm as described in [Page et al., 1999], is based on the random walk and theory of Markov Chains. A random surfer visits web pages by following the hyperlinks among them, and the process can be modeled as a Markov Chain with one state for each web page, where the hyperlink structure of the web is a directed graph, the nodes represent web pages, and the directed arcs represent hyperlinks. For example<sup>3</sup>, consider the small document collection consisting of six web pages linked as in Figure 2.2.

A Markov chain is characterized by a stochastic matrix  $P$  whose element  $p_{ij}$  is the probability of moving from state  $i$  (page  $i$ ) to state  $j$  (page  $j$ ) in one time-step, see Figure 2.3. This matrix is not stochastic, some rows of the matrix, such as row 2, contain all zeros. This occurs whenever a node contains no outlinks; many such nodes exist on the web. Such nodes are called dangling nodes.

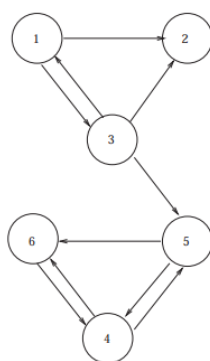


FIGURE 2.2: Directed graph representing a web of six pages.

$$\mathbf{P} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{matrix} .$$

FIGURE 2.3: The Markov chain matrix for the example graph.

In order to make this matrix a stochastic, one solution is to replace all zero rows,  $0^T$ , with  $\frac{1}{n}e^T$ , where  $e^T$  is the row vector of all ones and  $n$  is the order of the matrix (see Figure 2.4).

In the Brin and Page model, they force the transition probability matrix  $P$ , which is built from the hyperlink structure of the web, to be stochastic and primitive using the

<sup>3</sup>example from [Langville and Meyer, 2011]

$$\bar{\mathbf{P}} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

FIGURE 2.4: The Markov chain **stochastic** matrix for the example graph.

damping factor  $\alpha$  and the idea of teleport operation. They use then this primitive transition probability matrix to make the Markov chain an aperiodic chain, on which they apply the power method that converges to a stationary vector called the PageRank vector.

So, the stochastic  $\mathbf{P}$  matrix is transformed as follows:

$$\bar{P} = \alpha P + (1 - \alpha)E,$$

where  $0 \leq \alpha \leq 1$ , and  $E = \frac{1}{n}ee^T$ ,  $e$  is  $n$ -vector of all ones.

We call  $\mathbf{E}$  the *teleport operator* that represent a *random jump*. When performing the random jump action, the surfer is jumping to a page chosen randomly according to the random jump distribution vector (the teleport operator). When we apply the last adjustment on the example, this gives the next result (Figure 2.5).

$$\begin{pmatrix} 1/60 & 7/15 & 7/15 & 1/60 & 1/60 & 1/60 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 19/60 & 19/60 & 1/60 & 1/60 & 19/60 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 7/15 & 7/15 \\ 1/60 & 1/60 & 1/60 & 7/15 & 1/60 & 7/15 \\ 1/60 & 1/60 & 1/60 & 11/12 & 1/60 & 1/60 \end{pmatrix}$$

FIGURE 2.5: The **stochastic-primitive** matrix for the example graph.

From web graph point of view, a complete set of outgoing links from each web page to all others is added, therefore from each node there is a probability, called teleport, to reach all other nodes in the web graph.

In classic PageRank, the teleport operator is uniform but authors of the algorithm suggest taking other distribution vectors as a tool for personalization.

Finally, the computation of the PageRank can be done using the power iteration method that converges toward the PageRank vector.

### PageRank Modifications

The teleportation in the original PageRank is uniform, i.e., from a web page, the walker has the same probability to go to any other page. This uniform treatment is considered as unrealistic, especially in the case of the web of data, where the links are typed.

Therefore, many approaches propose to modify the original algorithm and the random walker model. The researchers in this domain propose other models like the “topic-sensitive PageRank”, “Rational Surfer” and the “intelligent surfer”.

Most of these modifications modify the *teleport operator*, by adding weights according to a prior knowledge about the graph.

In the next section, we present the selected works (highlighted in Table 2.3) and how they modified the PageRank to be suitable to rank entities of the web of data.

Dimension	Class	LDRANK	OntologyRank	PopRank	SemRank	ReConRank	AKTiveRank	NAGA	Hart et al.	TripleRank	RareRank	DBpediaRanker	DING	Tanon et al.	Sig.ma	Ranking complex relations	Dali and Fortuna	DWRank	RankProperties
Query Dependency	Query independent	x	x	x	x			x	x	x	x	x	x	x			x	x	x
	Query dependent	x				x	x								x	x			
Granularity	Document		x				x												
	Entity (Resource)	x		x		x		x	x	x	x	x	x	x	x		x	x	
Scope of the Dataset	Relationship				x			x		x			x	x	x	x			x
	Specific Ontology						x				x					x		x	
	Non-specific Ontology		x																
	Domain-based KB			x							x								
	Cross-domain KB	x	x		x	x		x	x	x		x	x	x	x		x		x
	LOD	x										x							x
	Embedded RDF					x							x						
	AA-RDF	x																	
Domain	Specific			x							x								
	Independent	x	x		x	x		x	x				x			x	x		x
Algorithm's type	Modified PageRank	x	x	x				x	x		x		x						
	Dynamic Modified PR					x													
	Discovery Search				x														
	Weighted Combination of measures				x		x					x	x	x	x	x			
	Hits based								x	x					x				
	Learn to Rank			x													x	x	x
	Language model (LM)							x											

TABLE 2.3: Comparative study: Summary of dimensions and approaches

### 2.2.2.2 Selected approaches

In the Table 2.3, we added our algorithm LDRANK in order to compare it and its context, to the works of the state-of-the-art. This comparison helped us to select the related works the most similar to ours. In this section, we will discuss the selected approaches in details, focusing on the aspects we are interested in, such as the *algorithm type* and the *ranking factor*.

#### (1) OntologyRank

Swoogle [Ding et al., 2004] is a semantic web search engine and a metadata search provider. Swoogle uses the OntologyRank as ranking algorithm, where the authors use cross-references between semantic web documents (SWDs) to define a graph and compute a score for each ontology in a manner analogous to PageRank used by the Google web search engine.

In this work, they consider an SWD as a web document in which the information is partitioned into smaller fragments. An SWD contains information that is modeled as an ontology or part of it. Authors distinguish between two kinds of documents, which they refer to as Semantic Web Ontologies (SWOs) and Semantic Web DataBases (SWDBs). A document is an SWO when a significant portion of its statements add new properties or constraints that define new terms or extend terms defined in other SWDs. On the other hand, an SWDB introduces individuals and makes assertions without adding or extending any term. OntologyRank calculates the relevance of SWDs taking into account four kinds of relationships among documents:

1. **imports:** SWD2 imports SWD1.
2. **uses-term:** SWD2 uses some of SWD1's terms without importing it.
3. **extends:** SWD2 extends the definitions of terms defined by SWD1.
4. **asserts:** SWD2 makes assertions about the individuals defined by SWD1.

They found that the PR model is not appropriate for the semantic web, the links have semantics and no longer equiprobable. Therefore, they suggest that the surfer in the semantic web should not be purely random, but rather rational. As an example: if an SWD imports some SWO, then clearly that link must be followed, the present SWD would not make any sense without the SWO it imports. Thus, OntologyRank propose to modify PageRank, such that instead of uniforming the probability of following any link in the graph (as in the original PR), OntologyRank assigns unequal probability of following links according to their types. Therefore, they assign different weights to



the four categories of inter SWD relations. This kind of analysis is independent of the user's information need, which make this algorithm (as the original PR) a static ranking algorithm.

We can characterize this method as simple, because it does not exploit the provenance, the content, neither the context. This simplicity makes this algorithm very efficient, but it may introduce some limitations depending on the usage (e.g. the limited amount of relationships that are examined during the ranking process).

## (2) **PopRank** [Nie et al., 2005]

PopRank is a query-independent, link analysis model to rank objects within a particular domain. It is also a PageRank-based algorithm that takes into account the semantic of the relationships among different objects, by assigning a popularity propagation factor to each type of object relationship. The various popularity propagation factors for these heterogeneous relationships were assessed with respect to their impact on the global popularity ranking.

The main use case of PopRank is ranking objects within a collection of authors, conferences, and journals. The algorithm considers both the web popularity of an object, and the object relationships to calculate its popularity score.

From the authors' point-of-view, the objects are contained over the web, the information about an object is usually rendered as a block of web pages. Thus, the web popularity can be computed by considering the PageRank scores of web pages containing the object and the importance of the web page blocks. If a page is popular, its object information is more likely to be read. The web popularity is used to denote the probability that a web user reads the information about an object. Therefore, the authors propose the "Random Object Finder" model, where the surfer starts his random walk on the web, and then he follows the object relationship links. Once he finds the first object on the web, he never hits back but eventually gets annoyed and will restart his random walk on the web again to find another seed object.

As OntologyRank, PopRank extends PageRank by adding different weights to each link depending on the type of relationship, like a citation between two papers, a paper is written by an author, publishing a paper in a conference, and so on. They call this weight the "popularity propagation factor (PPF)".

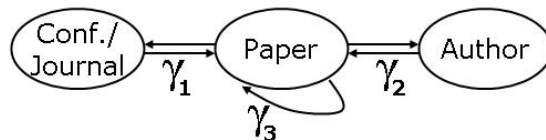


FIGURE 2.6: Paper Object Relationship Graph [Nie et al., 2005].

In Figure 2.6, we show an example of a small graph containing some links pointing to a paper object, where we need three propagation factors **PPF** ( $\gamma_1$ ,  $\gamma_2$ , and  $\gamma_3$ ) for the three different types of relationships: cited-by, authored-by, and published-by, respectively.

The ranking proposed by PopRank is also independent of the user's information needs, i.e. it is a static ranking algorithm.

### (3) ReConRank [Hogan et al., 2006]

On the contrary to the last two algorithms, ReConRank is a query-dependent ranking algorithm. ReConRank only analyzes result data that matches the query introduced by the user. Authors argue that the static strategy followed by PageRank is non-feasible for three reasons:

- The size of the dataset might be excessively large.
- The updating of the RDF data would suppose the ranking to be recomputed.
- The scores are not correlated with user queries, which means that returned results might not be the most relevant from the user point of view.

In this work, the authors introduce three algorithms: ResourceRank, ContextRank and ultimately ReConRank. They are the same algorithm but applied to different graphs; ResourceRank ranks the resources of the resulted graph, ContextRank is applied on the graph's context (provenances), and finally ReConRank calculates the ranking score relying on a unique unified graph of resources and provenances.

*ResourceRank* is a PageRank-like algorithm that aims at ranking RDF resources. It considers the resources as nodes in the graph. As PageRank, ResourceRank follows an iterative computation over a connectivity matrix that is derived from the graph structure. The difference from PageRank is that in the first iteration, PageRank initializes all nodes with an equal score. However, ResourceRank assigns to each node the ratio of all links that it receives as in-links. In addition, ResourceRank includes weightings during the computation, which can be manipulated in order to tune the ranking function.

*ContextRank* tries to improve the ranking quality by introducing provenance information. It defines a context as the provenance or source of data. It can be considered as

an extension to ResourceRank in order to include the provenance (the context) during the ranking computation.

*ReConRank* computes the ranking of result data (that match a user query) which can be considered as a topical subgraph. This subgraph contains two types of resources, first the resources matching the query, i.e. nodes directly linked to the particular root node (the query), and second the context, i.e. the neighbor resources that can be reached after a specific number of desired hops in the graph. Thus, the ranking value of an entity (or resource) is not only dependent on its own importance, but also the importance of its context (or provenances).

#### (4) **RareRank** [Wei et al., 2011]

In RareRank, the authors are interested in one specific domain: scientific publications. Therefore they propose to replace the random surfer model by another model more informed, they call it the “Rational Research Model”. They argue that for this specific domain, there is less randomness than for web search.

This algorithm is used in the **Semantic Web Search Engine (SWSE)**<sup>4</sup> in three steps: the data is crawled from the web, and then it is transformed into a directed labeled graph, and finally the RareRank algorithm is applied to compute ranking scores for resources.

Contrary to the last algorithms, RareRank uses not only the link information (e.g., a citation between two publications) but also the content information. The content information analysis refers to the existence of related information between two objects (resources) not connected by a physical link. This content information is modeled through the use of an ontology, which allows the navigation from a document to another even when there were no explicit link between them.

The computation of RareRank is based on the principle of PageRank. But in the case of RareRank, the authors use two transition graphs. The first is the *ontology schema* graph that designates the relations between ontological classes and the weights assigned to these relations. Weights in the ontology schema graph are customizable parameters. They essentially reflect the users’ search preferences and the semantics of a domain. They can be estimated automatically using sophisticated approaches, such as monitoring community of users’ search activities by collecting user click-through data.

Moreover, the second is the *knowledge base* graph that consists of the instances (or the entities) and their relationships instantiated from the schema ontology. The weight of a relation between two instances (A,B) is determined by the weight of the relation between

---

<sup>4</sup><http://swse.deri.org/>

the classes of the instances in the schema graph, and how many instances of the same type as B are linked to by A.

As for PageRank, the ranking vector is computed by applying the power iteration method to the transition probability matrix.

### (5) DING [Delbru et al., 2010]

In DING (or Dataset rankING), the authors modelize the web of data with two layers (see Figure 2.7); the *Dataset layer* and the *Entity layer*, where entities and their relationships are grouped in datasets, and the union of datasets and their relationships form the web of data. The entities are inter-linked by the RDF links (intra-dataset links) while the datasets are connected by the inter-dataset links. The inter-dataset links between two datasets are aggregated to form a weighted linkset on the dataset layer.

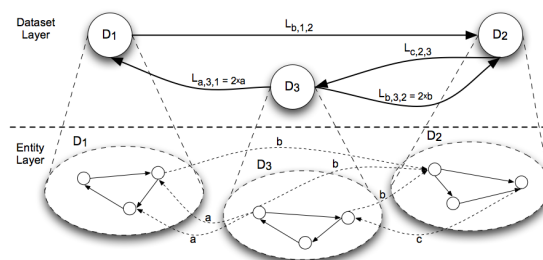


FIGURE 2.7: The two-layer model of the web of data. Dashed edges on the entity layer represent inter-dataset links [Delbru et al., 2010].

The weights of the linksets can be seen as an estimation of the probability to go from a dataset to another following a certain kind of link. In order to compute these weights, the authors propose a new measure similar to the TF-IDF in information retrieval, which is the *Link Frequency / Inverse Dataset Frequency (LF-IDF)*. This measure takes into account both the number of links contained in a linkset and the general importance of the label involved in the link. This measure promotes the links with a high frequency in a certain dataset and low dataset frequency in the dataset collection. For example, *dbpprop:reference* links defined by DBpedia will have a higher weight than links such as *rdfs:seeAlso*.

DING offers an adaption of PageRank to the web of data that exploits the locality of entities in their dataset and the inter-dataset relationships, where the final score is calculated in three steps:

1. **Dataset ranking:** the authors use the random surfer model applied to the weighted dataset graph (i.e. a modified PageRank) in order to calculate the dataset rank score.

2. **Entity ranking:** DING also calculates the ranking of the entities within a dataset. Thus, they propose two different ranking methods according to the existence of Spam within the dataset or not. If the dataset has a high chance of Spam, a robustness against Spam method is needed. Therefore, the authors apply the *Weighted EntityRank* method, which is based on PageRank and exploits the internal entities and intra-links of a dataset in order to compute the importance of an entity node within a dataset. When one can assume that the dataset is Spam-free, another more efficient method is used, *The Weighted LinkCount*, which is a weighted in-degree counting links algorithm.
3. **Global entity ranking:** the authors argue that the EntityRank and LinkCount represent good solutions for local entity ranking. However, an approach that combines them with the Dataset ranking in order to take into account the particularity of each dataset will give better results. In this case, the Dataset rank represents the probability of selecting the dataset, and the local entity rank (EntityRank or LinkCount) represents the probability of selecting an entity within this dataset. However, this approach introduces a problem, that the global ranking favors the smaller datasets, e.g. a small dataset that receives even a single link is likely to possess its top entities' score way above many of the top ones from bigger datasets. Therefore, they propose to normalize the local ranks based on the dataset size.

### 2.2.3 Conclusion

By studying these last approaches, we noticed that although some works (like ReConRank) rank RDF items embedded in the web pages, none of them handle the case of ranking entities in an RDF graph produced by an automatic annotation process. We also notices that most of the entity ranking algorithms that deal with ranking in a context similar to ours are based on a modification of PageRank. These methods adapt PageRank for the web of data and share basically the same idea, i.e. search for a source of prior knowledge about the importance of the graph's nodes, then propose a specific modification based on this prior knowledge.

The studied approaches do not explore how to benefit from available textual descriptions of the contexts where the entities appear. In addition, they are not able to integrate complementary sources of prior knowledge (e.g. the ranking produced by a web search engine, and the textual evidences provided by the content of the web pages, etc.)

Thus, our proposed algorithm LDRANK introduces a new prior knowledge based on an iterated latent semantic analysis of the textual data that can be used to describe the nodes. But, what really distinguishes LDRANK from existing approaches is that it

offers a generic solution in which we can combine several sources of prior knowledge to transform the stochastic transition matrix into an ergodic Markov chain whose steady state is the final ranking, i.e. LDRANK proposes a generic solution to adapt PageRank to the web of data.

## 2.3 Our Algorithm (LDRANK)

### 2.3.1 Context

LDRANK computes a ranking of the entities of an RDF graph. It expects descriptive textual data to be associated with each entity. LDRANK also depends on the knowledge of an information need expressed as a query made of a set of keywords.

We focus on the use case of ranking LOD entities automatically found in a web page returned by a web search engine given a user's query. Today, many solutions are available to detect LOD entities in plain text (e.g., DBpedia Spotlight, AlchemyAPI, etc.). We built our evaluation dataset using DBpedia Spotlight (cf. section 2.4.2.1).

Apart from the explicit structure of the graph, LDRANK uses two sources of prior knowledge about the importance of the entities: (i) the ranking provided by a web search engine for the web pages within which the entities have been detected, and (2) implicit relationships discovered through an iterated latent semantic analysis of the descriptive texts associated with the entities.

LDRANK is well adapted to graph that are sparse (i.e., with few explicit links between the entities) and noisy (i.e., with a high proportion of nodes irrelevant to the information need). The evaluation dataset introduced in section 2.4.2.1 is made of such sparse and noisy graphs.

The core of LDRANK comprises three key components. First, LDRANK uses the explicit structure of the graph through a PageRank-like algorithm. Second, it uses the implicit relationships that can be inferred from the text associated with the entities through an original variation of the Singular Value Decomposition (SVD) algorithm. Finally, LDRANK also takes into account the ranking of the web pages where the entities were found thanks to a scoring function first introduced by Fafalios and Tzitzikas [Fafalios and Tzitzikas, 2014].

Both the SVD-based textual analysis and the use of the ranking obtained from a web search engine, result in probability vectors that express some prior knowledge (or belief) about the importance of the entities (see sections 2.3.2 and 2.3.3). These probability

vectors are then combined through a consensual linear opinion aggregation strategy first introduced by Carvalho and Larson [Carvalho and Larson, 2013] (see section 2.3.4). This process results in a unique probability distribution over the entities.

This later probability distribution is used to influence the convergence of a PageRank-like algorithm towards a stable probability distribution, i.e. the eigen vector of eigen value unity for the Markov process modeling the graph structure that links the entities together. This distribution is the final ranking for the entities (see section 2.3.5).

### 2.3.2 Prior Knowledge Derived from the Ranking Provided by the Web Search Engine Result Page

**Algorithm H** (*Hit Score*). This algorithm computes a probability vector (*hitdistrib*) that represents prior knowledge about the importance of the entities. This knowledge is derived from the ranking of the web pages in which they were detected. This ranking is returned by a web search engine to answer the user’s information need expressed as a query made of a set of keywords. This strategy was first introduced by Fafalios and Tzitzikas [Fafalios and Tzitzikas, 2014].

**H1.**  $A \leftarrow$  the list of the top web pages as ranked by a web search engine to answer a query made of keywords.

**H2.**  $E \leftarrow$  the set of detected entities in these web pages (e.g., the result of applying DBpedia Spotlight).

**H3.**  $docs(e) \equiv$  the documents of  $A$  that contain the detected entity  $e$ .

**H4.**  $rank(a) \equiv$  the rank of document  $a$  in  $A$ .

**H5.**  $hitscore(e) \equiv \sum_{a \in docs(e)} (size(A) + 1) - rank(a)$

**H6.**  $hitdistrib[e] \leftarrow hitscore(e) / \sum_{e' \in E} hitscore(e')$

**H7.** [End.]    **■**

### 2.3.3 Prior Knowledge Derived from an Iterative Latent Semantic Analysis of the Textual Data Describing the Entities

**Algorithm S** (*Iterative SVD*). This algorithm computes a probability vector (*svddistrib*) that represents prior knowledge about the importance of the entities based on an analysis of the textual data associated to each entity.

- S1.** [*Initial matrix.*]  $R \leftarrow$  the sparse entity-stem matrix (i.e., entities in rows, stems in columns) in Compressed Column Storage (CCS) format<sup>5</sup>.
- S2.** [*Initial important entities.*]  $info\_need \leftarrow$  a set of entities, viz. the union of (i) the entities detected in the text of the query and (ii) the entity with the best hitscore (for the case when no entities were detected in the query). We assume that these entities are likely to be close to the information need of the user.
- S3.** [*First SVD.*]  $(U, S, V^T) \leftarrow svdLAS2A(R, nb\_dim)$  Compute the singular value decomposition (SVD) of  $R$  at rank  $k = nb\_dim$ . Since  $R$  is very sparse, we use the *las2* algorithm developed by Michael W. Berry [Berry, 1992] to compute the decomposition:  $R_k = U_k S_k V_k^T$  with  $U_k$  and  $V_k$  orthogonal,  $S_k$  diagonal, such that  $\|R - R_k\|_F$  is minimized (i.e., from the perspective of the Frobenius norm,  $R_k$  is the best rank- $k$  approximation of  $R$ ).
- S4.** [*Entities' coordinates in the reduced space.*]  $SUT \leftarrow SU^T$  In the new  $k$ -dimensional space, this operation scales the coordinates of the entities (i.e., the rows of  $U$ ) by their corresponding factor in  $S$ . Thus, we obtain the coordinates of the entities in the reduced space (i.e., the columns of  $SUT$ ).
- S5.**  $prev\_norms \leftarrow$  euclidean norms of the entities in the reduced space.
- S6.** [*Updated matrix.*]  $R' \leftarrow R$  where the rows corresponding to the entities of  $info\_need$  have been multiplied by the *stress* parameter (since  $R$  is in CCS format, it is more convenient to do this operation on the transpose of  $R$ ).
- S7.** [*Second SVD.*]  $(U', S', V'^T) \leftarrow svdLAS2A(R', nb\_dim)$
- S8.** [*Updated entities' coordinates in the reduced space.*]  $SUT' \leftarrow S'U'^T$
- S9.**  $norms \leftarrow$  updated euclidean norm of the entities in the reduced space.
- S10.** [*Drift of the entities away from the origin of the reduced space.*]  
 $svdscore(e) \equiv norms[e] - prev\_norms[e]$ .
- S11.**  $svddistrib[e] \leftarrow svdscore(e) / \sum_{e'} svdscore(e')$
- S12.** [*End.*] **■**

<sup>5</sup>[http://netlib.org/linalg/html\\_templates/node92.html](http://netlib.org/linalg/html_templates/node92.html)



We shall now introduce the essential property of the SVD on which relies Algorithm S. For a strong dimensional reduction (i.e., for small values of  $k$ ), the transformation  $S_k U^T$  tends to place near the origin of the  $k$ -dimensional reduced space, the entities that, in the row space of  $R$ , were orthogonal to many other entities. Indeed, the SVD can be seen as an optimization algorithm: to minimize the error due to the impossibility for an entity to be orthogonal to more than  $k$  non co-linear entities, this entity should be placed as close to the origin as possible. A similar argument can be used to show that entities co-linear to many other entities in the row space of  $R$  will also tend to be near the origin of the  $k$ -dimensional space. Thus, the entities far from the origin in the reduced space tend to have “interesting” (i.e., discriminative) relationships with other entities also far from the origin. Furthermore, the entities that are in the direction of greatest variation of the data in the original space are those well aligned with the axis corresponding to the biggest singular value in the reduced space (i.e., the axis that suffered from the greatest extension). Therefore, these entities tend to remain far from the origin of the reduced space. Thus, the principle of algorithm S is to artificially increase the importance of the entities semantically close to the query in order to force them in the direction of the greatest variation of the data, and to observe among the other entities those which, after this transformation, move the most away from the origin of the reduced space. With the argument stated above, these entities may therefore be qualified as potentially related to the information need in an interesting/discriminative way.

We obtained the best experimental results with a reduction to the one dimensional line (i.e., with  $nb\_dim = 1$  in steps S3 and S7 of Algorithm S), and with a stress factor (step S6 of Algorithm S) of 1000.0.

#### 2.3.4 Belief Aggregation Strategy

We consider *hitdistrib* (from Algorithm H), *svddistrib* (from Algorithm S), and the equiprobable distribution (*equidistrib*) as three experts’ beliefs (or prior knowledge) about the importance of the entities. To aggregate these beliefs, we apply the algorithm of Carvalho and Larson [Carvalho and Larson, 2013] that computes iteratively an optimal linear combination of multiple probability vectors. At each step of this iterative algorithm, the expert  $i$  re-evaluates the distribution that represents her opinion as a linear combination of the distributions of all the experts. In this linear combination, the weight associated by expert  $i$  to the distribution of expert  $j$  is proportional to the distance between the two distributions. The authors define this distance such that the process converges towards a consensus. We will refer to this resulting consensual probability vector by the name *finaldistrib*.

### 2.3.5 LDRANK

The PageRank [Page et al., 1999] algorithm transforms the adjacency matrix ( $M$ ) of a network of web pages into a matrix  $H$  which is both *stochastic* (i.e., each row of  $H$  sums to 1) and *primitive* (i.e., there is an integer  $k$  for which  $H^k > 0$ ), thus assuring the existence of a stationary vector (i.e., the positive eigenvector corresponding to the eigenvalue 1). This stationary vector is a probability vector that can be interpreted as representing the *importance* of each web page by rigorously modeling the intuitive proposition according to which an important web page is referenced by important web pages. Moreover, this stationary vector can be computed efficiently with the power iteration algorithm by taking into account the sparsity of the stochastic matrix.

In the original version of the PageRank algorithm, no assumption is made about the probability of importance of the web pages before the link analysis takes place. In other words: first, the matrix  $M$  is transformed into a stochastic matrix  $S$  by replacing each null row by the equiprobable distribution (*equidistrib*); second, the matrix  $S$  is transformed into a primitive matrix  $H$  by a linear combination with the so-called teleportation matrix ( $T$ ):  $H = \alpha S + (1 - \alpha)T$  with each row of  $T$  being the equiprobable distribution (*equidistrib*).

In algorithm LDRANK, instead of using the equiprobable distribution, we use the consensual distribution (*finaldistrib*) introduced above (see section 2.3.4). We obtained the best experimental results for  $0.6 \leq \alpha \leq 0.8$ . Moreover, we set at  $1E - 10$  the value of the convergence threshold controlling the termination of the power iteration method that computes the stationary vector.

LDRANK is available online under an open-source license<sup>6</sup>.

## 2.4 Evaluation

### 2.4.1 Introduction

In this section, we show that LDRANK produces rankings of a significantly better quality than the ones produced by comparable strategies from the state of the art.

---

<sup>6</sup>Source code available online under an open-source license <http://liris.cnrs.fr/drim/projects/ensen/>

## 2.4.2 Build the evaluation dataset

LDRANK is meant to solve the problem of ranking LOD entities automatically detected in web pages. These web pages were themselves returned by a web search engine to answer a user’s query expressed as a set of keywords. To our knowledge, there is no evaluation dataset adapted to this context (i.e., query-biased ranking of LOD entities automatically detected in plain text). Therefore, we used a crowdsourcing approach to build our evaluation dataset (available online<sup>7</sup>).

In the next sections, we describe the process through which we obtained our dataset, then how we use it to evaluate LDRANK.

### 2.4.2.1 Data Collection

We now describe the data collection process. We start with 30 queries randomly selected from the “Yahoo! Search Query Tiny Sample” dataset offered by the Yahoo! Webscope project<sup>8</sup>. We submit the queries to the Google search engine. For each query, we keep the top-5 web pages. For each one of the 150 HTML web pages, we extract the main raw textual content with the algorithm proposed by Kohlschütter, Fankhauser, and Nejdil [Kohlschütter et al., 2010]. After this process, we keep on average 467 words by web page. Then, we use the DBpedia Spotlight [Mendes et al., 2011] service to detect LOD entities in these texts. On average, 81 entities are found in each web page.

For each web page, we build an RDF graph by issuing SPARQL queries over the DBpedia dataset to discover all the RDF triples that link the entities detected in the web page. For each entity of the graph, we associate a text which is the concatenation of the entity abstract as defined in DBpedia and of a 300 characters text window centered on excerpts of the web page at the locations where the entity has been detected. Finally, we process this text by removing stop words and by reducing the words to their stems.

### 2.4.2.2 Microtasks Generation

We have to evaluate the relevance of the annotated entity to the user’s information need as expressed by the initial query. After the processing described in the previous section, the text we keep for each web page is too lengthy to be the object of a single crowdsourcing task (i.e., it would be too much work for a single worker to verify the relevance of all the detected entities in a web page). Therefore, we divide this task

<sup>7</sup><http://liris.cnrs.fr/drim/projects/ensen/>

<sup>8</sup><http://webscope.sandbox.yahoo.com/catalog.php?datatype=1>

into smaller “microtasks”. In our case, a microtask consists in scoring the relevance of the annotated entities in a single sentence. Thus, we split the text of a web page into sentences by applying the ICU BreakIterator algorithm<sup>9</sup>. On average, there are 22 sentences by document. Moreover, if a sentence still contains more than 10 annotated entities, the work will be split over multiple microtasks.

We used the CrowdFlower<sup>10</sup> crowdsourcing platform. This platform distributes the work to about 5 millions contributors in roughly 154 countries while maintaining quality metrics for each contributor. The design of a microtask is specified in CML, an XML language provided by CrowdFlower. For each microtask, we give the contributor a short list of instructions about how to complete the work (we tested many formulations until we found a suitable one understood by all contributors). We provide the contributor with a topic made of a title (viz., the initial query from which were obtained the web pages where we discovered LOD entities) and a short text (viz., the sentence that contains the entities that the worker will have to evaluate, extended with the previous and the following sentences to add some context and ease the task of the worker).

For each entity in the sentence, we ask the worker to rank it on the following ordinal scale (this scale was used by Järvelin and Kekäläinen when they introduced the DCG graded relevance [Järvelin and Kekäläinen, 2000]: “irrelevant” (0), “marginally relevant” (1), “fairly relevant” (2), and “highly relevant” (3). Moreover, the worker is helped in his task with a short text that describes the general meaning of the entity (viz. the beginning of its DBpedia abstract). For each microtask we collect 10 judgments. A microtask is paid \$.01.

### 2.4.2.3 Quality Control

We only accept contributors that have already completed over a hundred microtasks with a high overall accuracy. This accuracy is a quality factor provided by the CrowdFlower platform. To measure the accuracy of a contributor, CrowdFlower introduces test-questions. Contrary to a normal question, a test-question is accompanied by its correct answer given by the designer of the question. Thus, it is considered a good practice to include test-questions during the design of a task.

We give a contributor at most 30 minutes to provide her answer. Also, contributors have to spend at least 10 seconds on a task before giving an answer. We measured the agreement between contributors with the Krippendorff’s alpha coefficient [Krippendorff, 2012]. By default, this coefficient uses a binary distance to compare answers. However,

<sup>9</sup>[http://icu-project.org/apiref/icu4c/classicu\\_1\\_1BreakIterator.html](http://icu-project.org/apiref/icu4c/classicu_1_1BreakIterator.html)

<sup>10</sup><http://www.crowdfunder.com/>

this metric has been designed to accept other distances if required. Thus, to take into account the fact that we used an ordinal scale encoding both correctness (i.e., = 0 VS > 0) and relevance (i.e., from 1 to 3), we used an ad-hoc symmetric distance:

d	0	1	2	3
0	0.00	0.50	0.75	1.00
1	0.50	0.00	0.25	0.50
2	0.75	0.25	0.00	0.25
3	1.00	0.50	0.25	0.00

With these parameters, we obtained an alpha of 0.22. According to Landis and Koch’s scale [Landis and Koch, 1977], this can be considered a fair agreement (the scale was designed for Fleiss’ kappa, but Krippendorff’s alpha is in most ways compatible with Fleiss’ kappa). However, by comparison with existing works that applied crowdsourcing in an information retrieval context, we could expect a higher alpha. For example, Jeong et al. obtained a Fleiss’ kappa of 0.41 (i.e., moderate agreement) for a search engine that appeals to the wisdom of crowds to answer questions. However, Alonso et al. obtained a Krippendorff’s alpha between 0.03 and 0.19 for a more subjective task: deciding if a tweet is or is not interesting. Thus, to better understand the characteristics of our dataset, we also tried to isolate the contributors that often disagreed with the majority. In fact, by removing the contributors that disagree with the majority in more than 41.2% of the cases, we obtained a Krippendorff’s alpha of 0.46. Then, 96.5% of the tasks have at least 3 judgments, 66% of the tasks have at least 5 judgments, and only 0.7% of the tasks have only 1 judgment.

#### 2.4.2.4 Aggregation of the Results

We used the majority voting for aggregating the results within each sentence. We tested two different methods to break ties: (i) the maximum of the mean of the contributors’ accuracy (the accuracy of a contributor is the aforementioned metric provided by the CrowdFlower platform), or (ii) the highest value. We discovered later that these two choices result in very similar outcomes when we use the dataset to compare query-biased entity ranking algorithms. We used the same majority voting strategy to aggregate the results at the level of a web page.

The dataset we just described will be used in the next section 2.4.3 to evaluate LDRANK and to compare it to the state of the art.

### 2.4.3 Experiments

We compared four ranking strategies, each one of them is based on a different source of prior knowledge used to inform a PageRank-like algorithm:

- the traditional version in which there is no available prior knowledge about the importance of the entities. This absence of knowledge is rendered as an equiprobable distribution (we name this strategy EQUI);
- the version modified with the hitscore as prior knowledge (due to Fafalios and Tzitzikas [Fafalios and Tzitzikas, 2014], see section 2.3.2, we named it HIT);
- the version modified with our new source of prior knowledge based on the iterative application of the SVD (see section 2.3.3, we named this strategy SVD);
- LDRANK that uses the consensual combination of the three previous sources of prior knowledge.

In order to compare the four strategies (EQUI, HIT, SVD and LDRANK), we used the NDCG (Normalized Discounted Cumulative Gain) metric. The DCG (Discounted Cumulative Gain) at rank  $r$  is defined as:  $DCG_r = rel_1 + \sum_{i=1}^r \frac{rel_i}{\log_2 i}$ . (with  $rel_i$  representing the estimated relevance of the result to the rank  $i$ , the relevance is represented in a discrete scale which generally contains much less values than the total number of the results, we use a scale of four gradations, see section 2.4.2). The NDCG at rank  $r$  is equals to the DCG at rank  $r$  normalized by the DCG of an ideal ranking at rank  $r$ .

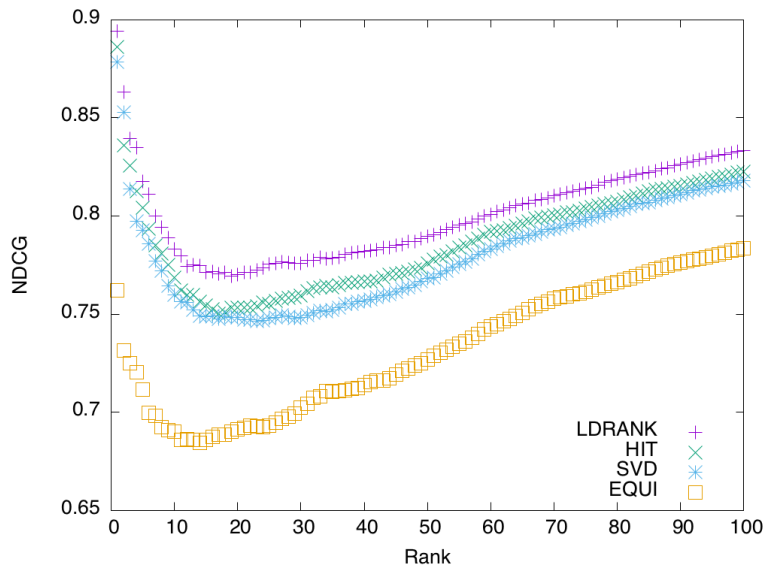


FIGURE 2.8: Comparison of the NDCG scores for the four different strategies (*EQUI*, *HIT*, *SVD* and *LDRANK*)

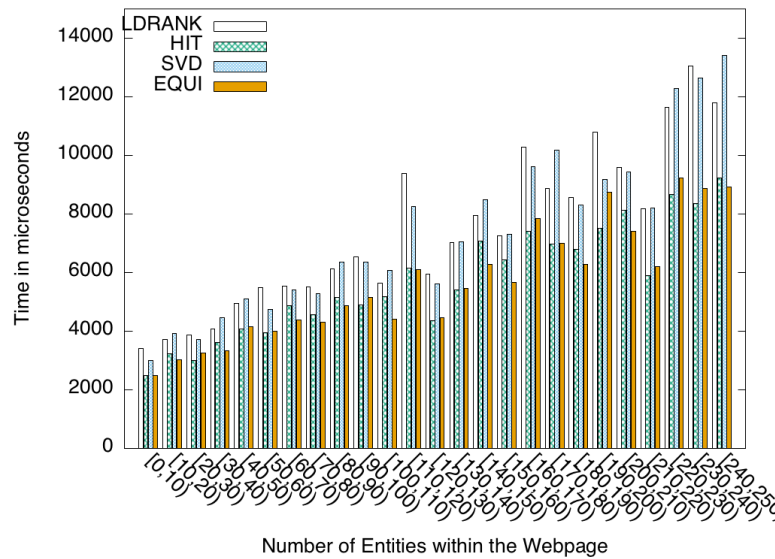


FIGURE 2.9: Comparison of the execution time for the four different strategies (with processor: 2.9 GHz Intel Core i7, and memory: 8 GB 1600 MHz DDR3)

#### 2.4.4 Results and discussion

The results are presented in Figure 2.8. We can see that the SVD and HIT strategies obtain similar performances. However, they are clearly outperformed by their consensual combination (LDRANK). Moreover, since we systematically take into account the sparsity of the data, we obtain good execution times (see Figure 2.9). The SVD strategy takes more time than the HIT strategy since it has to compute the SVD. The additional time spent by the combined strategy (LDRANK) is due to the time necessary to converge towards a consensus. Finally, we did similar experiments by considering the edges of the graph bidirectional. Indeed, the direction of an RDF predicate is often arbitrary in practice. The relative performance and accuracy of the algorithms were similar, but the absolute NDCG scores were slightly better.

It should be noted that through these experiments, beside introducing a new efficient ranking strategy based on an original use of the SVD dimensionality reduction, we are also offering evidence that different strategies based on a modification of the teleportation matrix for the PageRank algorithm can profitably be combined when considered as various sources of *prior* knowledge about the importance of the entities.

## 2.5 Conclusion

We proposed a new algorithm, LDRANK, for ranking the entities of a graph coming from the web of data. This graph can be sparse and noisy. However, descriptive textual data

is associated to its nodes. Also, the ranking must be performed given the knowledge of an information need expressed as a set of keywords. Such graphs appear in particular as the result of the automatic detection of LOD entities in a web page (e.g., with DBpedia Spotlight). In our approach we take into account both the explicit structure provided by the web of data, and the implicit relationships that can be discovered by an analysis of the textual content of the web page.

LDRANK being a variation of the PageRank algorithm, we represent the structure of the graph with a stochastic transition matrix. To make the corresponding Markov chain ergodic, we do a convex combination of the transition matrix with a rank-1 matrix corresponding to the linear consensual combination of the sources of prior knowledge. We should note that the prior knowledge matrix is of rank 1 because the prior importance of a node is obtained without knowledge about the graph's structure, and thus it remains independent of the previous state. Since the Markov chain is now ergodic (i.e., each state is aperiodic and positive recurrent), we can use iterative algorithms (e.g., the power iteration method) that converge to its steady state. Moreover, since the prior knowledge is kept as a rank-1 matrix, LDRANK can still benefit from the sparsity of the stochastic transition matrix. Finally, as usual for PageRank-based algorithms, we interpret the steady state as a ranking of the graph's nodes.

We built a dataset, using crowdsourcing, in order to evaluate our algorithm. We found that LDRANK computes a ranking of a significantly better quality than the rankings produced by the approaches of the state of the art.

In this context, the good accuracy of LDRANK made possible the generation of generic, useful and usable semantic snippets (Chapter 3). Thus, it opens the way to new applications that will benefit from a symbiosis between the web of data and the web of documents.



## Chapter 3

# The Semantic Snippets

### 3.1 Introduction

The variability and the complexity of the information (explicit or implicit) available on the web, places a burden on both the IR researchers and IR system designers. Many studies (such as [Callan et al., 2007, Kelly, 2009]) insist that the search success and the searcher’s satisfaction with a system depend not only on the quality of the IR model or the efficacy of the system, but it also depends on what interactive features a system offers and how it encourages searchers to employ these features. Indeed, the success and satisfaction depend on offering support for the searchers’ personal strategies. They also depend on letting the searcher understand how the system operates.

The area of interactive information retrieval (IIR) studies the searchers’ behavior. The most common method for information seeking and retrieval on the web is the query-based search engines. Query-based systems force searchers to pull information out of the web by expressing a request.

IIR focuses on encouraging and improving users’ interactions with an IR system. IIR systems propose both novel interfaces and novel interactive functionalities that help the users to organize the information, to structure their investigation of an information resource, or to make interactive decisions. Designing interfaces that support more difficult interactive decisions, such as refining a query, is challenging for researchers. Many studies proposed new interactive artifacts or some enhancements on existing ones.

The rise of Linked Data (and the ontologies), as part of the semantic web initiative offers new important sources of information that can be used to propose new methods and interactive artifacts in IIR.

We propose to benefit from the revolution of the semantic web and Linked Data, to improve a traditional artifact in the search interface, the *snippet*<sup>1</sup>. Thus, we present the *Semantic Snippets*, a new semantic and interactive artifact that facilitates the knowledge transfer (collecting and transferring information from the web to the user) by introducing new elements in the SERP that semantically contextualize the user information need in each result.

We propose to analyze the results returned by a web search engine using information from the result itself (web of document) and structured data from the web of data (via Automatic Annotation), then to generate semantic snippets that we integrate into the final SERP.

To our knowledge, at the moment of writing this thesis, there are no works focused on generating snippets from the automatic detection of LD entities in the text of the web

---

<sup>1</sup>In some works, in this domain, the authors call it the *surrogate*

pages, and not restricted to an a priori fixed domain of knowledge but generic enough to suit any web page. This missing approach would be highly beneficial since it could provide rich snippets for all the web pages, even the ones (viz., the vast majority of them) that lack any embedded structured metadata. We make the hypothesis that an efficient query-biased ranking algorithm for LD entities detected automatically in web pages will make possible the generation of generic semantic snippets. It seems that today there is a clear need for this algorithm, to allow the emergence of applications that enhance the snippets and the SERPs. Therefore, in chapter 2, we studied approaches in the query-biased ranking domain, pointing out that none of them were well adapted to our context, and we proposed the *LDRANK* algorithm.

In the following sections, we discuss the state of the art in the domain of enhancing snippets for the web of documents and the web of data. Then, we describe the elements of the semantic snippet, the semantic snippet as an interactive artifact, and its generation. Finally, we present in details the machine learning methodology that we applied to support the generation of semantic snippets.

A full description of the system that generates the semantic snippets is presented in chapter 4.

## 3.2 Related Work

We first mention works that generate snippets for native RDF documents. Next, we introduce works that generate snippets for traditional web pages by relying either on embedded structured metadata (e.g., RDFa, microformats, etc.) or on ad-hoc scripts that extract structured information from a few pre-determined domains (e.g., `en.wikipedia.org`, `youtube.com`, etc.).

### 3.2.1 Enhancing Snippets for the Semantic Web

Many approaches tried to enhance SERPs' snippets. [Ge et al., 2012], and [Penin et al., 2008] focus on the generation of snippets for ontology search. [Bai et al., 2008] generate snippets for a semantic web search engine.

In [Penin et al., 2008], the authors first identify a topic thanks to an off-line hierarchical clustering algorithm. Next, they compute a list of RDF sentences (i.e., sets of connected RDF statements) semantically close to the topic. Finally, using a similarity measure based on WordNet, they rank the selected RDF statements by considering both

structural properties of the RDF graph and lexical features of the terms present in the ontology (by way of a WordNet-based similarity measure).

In [Ge et al., 2012], the authors first transform the RDF graph into a term association graph in which each edge is associated with a set of RDF sentences. Their goal is to produce a compact representation of the relationships existing between the terms of the query. These relationships are to be found in the RDF graph. To do this, they decompose the term association graph into maximum  $r$ -radius components to avoid long distance relationships between query terms. Next, they search sub-snippets in these components (i.e., connected subgraphs that link some of the query terms). Finally, they select some of the sub-snippets to form the final snippet.

In [Bai et al., 2008], the authors first assign a topic to the RDF document. To do this, they use a property such as *p:primaryTopic* if it exists, otherwise they rely on a heuristic based on the comparison of the URI of the candidates topic-nodes with the text of the URL of the RDF document. In this regard, it seems interesting to note how they use the properties: for each property, they define its relative importance to other properties of a given schema, they also introduce the notion of correlative properties (e.g. `foaf:name` and `foaf:family`) and exclusive properties (e.g. `foaf:name` and `foaf:surname`). Finally, they use this ranking algorithm to give the user a set of relationships between the query-related statements and the topic-related statements.

To sum up, we are in harmony with [Ge et al., 2012] and [Bai et al., 2008] regarding the great opportunities offered by highly structured RDF data from which one can find non-trivial relationships among the query terms, and between the query terms and the main concepts of the document. Moreover, we join Penin *et al.* [Penin et al., 2008] when they stress the necessity to design a ranking algorithm for RDF statements that considers both the structure of the RDF graph and lexical properties of the textual data.

### 3.2.2 Enhancing Snippets for the Web of documents

The previous works exploit native RDF documents, but, in general, the LD entities can either come from: (i) a LD dataset (e.g. by way of SPARQL queries), (ii) semantic annotations embedded in a web page (i.e., by using RDFa, Microdata, or Microformats<sup>2</sup>), or (iii) automatic detection of entities in the web page (for example through DBpedia Spotlight [Mendes et al., 2011]).

---

<sup>2</sup>[www.w3.org/TR/xhtml1-rdfa-primer/](http://www.w3.org/TR/xhtml1-rdfa-primer/) ; [microformats.org/](http://microformats.org/) ; [www.w3.org/TR/microdata/](http://www.w3.org/TR/microdata/)

Among the approaches that offer to enhance the snippets on the web of documents using the web of data, [Haas et al., 2011] and [Steiner et al., 2010], do not rely on the automatic annotation, they use only explicit embedded annotations.

[Haas et al., 2011] employed structured metadata (i.e., RDFa and several microformats) and information extraction techniques (i.e., handwritten or machine-learned wrappers designed for the top hostnames, e.g., `en.wikipedia.org`, `youtube.com`,...) to enhance the snippets with multimedia elements, key-value pairs, and interactive features. By combining metadata authored by the documents' publishers with structured data extracted by ad-hoc wrappers designed for a few top host names, they can build enriched snippets for many results of a typical query. They chose explicitly not to use directly the LOD graph to avoid the problem of the transfer of trust between the web of documents and the web of Data. Indeed, they argue that the quality of the editorial processes that produce the web of Data from the web of documents (e.g. the transformation from Wikipedia to DBpedia) cannot be controlled. Therefore, from their point of view, using the LOD through an automatic entity detection process to enhance the snippet would come with a risk considered too important of introducing uncontrolled noise in the results.

Also, Google Rich Snippet (GRS) [Steiner et al., 2010] is a similar initiative that relies exclusively on structured metadata authored by the web pages' publishers. Google introduced rich snippets on May 2009 as a means of displaying structured data in the SERP with the objective of highlighting the searched-for properties to the user in a visually outstanding way [Steiner et al., 2010].

Google Rich Snippets uses its own restricted vocabulary and supports only a few data types:

1. **Product:** Information about a product, including price, availability, and review ratings.
2. **Recipe:** Recipes that can be displayed in web searches and Recipe View.
3. **Review:** A review of an item such as a restaurant, movie, or store.
4. **Event:** An organized event, such as musical concerts or art festivals, that people may attend at a particular time and place.
5. **Software Application:** Information about a software application, including its URL, review ratings, and price.

We agree with both [Haas et al., 2011] and [Steiner et al., 2010], about the importance of the embedded data in the SERP enhancement. However, a study made in 2013 [Bizer

et al., 2013] on the over 40 million websites of the Common Crawl corpus<sup>3</sup> showed that there is a lake of rich embedded structured data on the web. Only 5.64% of the websites contained embedded structured data, however, nearly 50% of the top 10,000 websites of the Alexa list of popular websites<sup>4</sup> had structured data.

Moreover, the authors of the study say that: “The topics of the [structured] data [...] seem to be largely determined by the major consumers the data is targeted at (Google, Facebook, Yahoo!, and Bing)”. Therefore, there is still a clear need for a high quality process that, given a document relevant to a Web search query, can select the most relevant resources among those automatically discovered within the document (e.g., through state of the art NLP algorithms), and this, whatever the document’s provenance may be. An efficient algorithm for ranking the resources of a LOD graph while taking into account their textual context could serve this purpose.

### 3.3 Elements composing a Semantic Snippet

In this section, we present the elements of the proposed semantic snippets to make it clear to the reader what enhancements we propose, and in what it is different from traditional textual snippets.

The *Semantic Snippet* is more informative, more expressive, richer and (in contrast to the traditional snippet) interactive. It is made of five elements (see Figure 3.1):

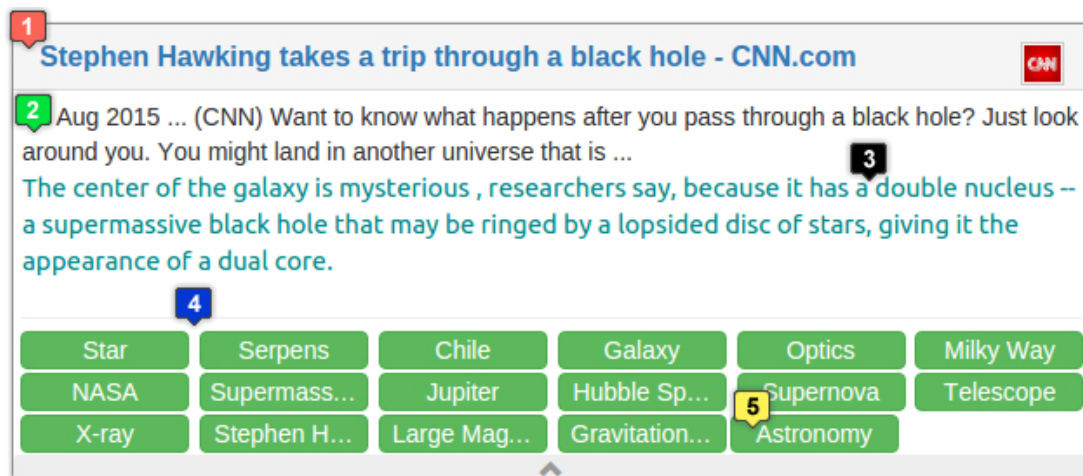


FIGURE 3.1: The semantic snippet layout

1. **The title of the document**, which is also a link to the corresponding web page.

<sup>3</sup><http://commoncrawl.org>

<sup>4</sup><http://www.alexa.com/topsites>

2. **The original snippet of the target search engine**, enhanced with annotation of the top-ranked entities<sup>5</sup> belonging to it, highlighted and linked to a human readable description of the entity (a web page describing the entity).
3. **The main sentence**, selected by machine learning to represent, as faithfully as possible the relationship between the query, the document and the top-ranked entities (see section 4.2.3.4).
4. **The primary concepts**, made of the top-ranked entities chosen by LDRANK, they are clickable, and when the user clicks on the name of the entity, we provide her with an *Entity Description*.
5. **The entity description** is a GUI panel, shown on-demand (see figure 3.2), that regroups different elements describing the entity. It starts with the entity's abstract and a picture, and then the context of the entity (see section 4.2.3.4), the related entities (see section 4.2.3.4) and finally a graphical representation of the subgraph centered around this entity.

### 3.4 Semantic Snippet as an Interactive Artifact

As the semantic snippet (and the resulted SERP) is an interactive artifact, and the interaction with the user plays the leading role, we discuss here how we selected and designed the interactive components of the interface.

The semantic snippet (and the resulted SERP) presents a vast amount of information in the same place in an interactive way. Therefore, it can be analyzed under the “information seeking mantra” proposed by Ben Shneiderman [Shneiderman, 1996].

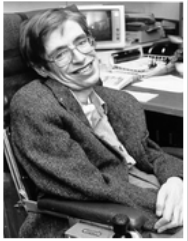
The information seeking mantra is a list of high-level guidelines for designing information visualization applications. It was presented as inspirational guidelines rather than a prescriptive guide. It proposes to have the next elements in the interface of any information visualization system:

- **Overview first:** gain an overview of the entire collection.
- **Zoom:** zoom in on items of interest.
- **Filter:** filter out uninteresting items.

---

<sup>5</sup>The term “top-ranked entities” describes the selected entities (the top ones) from the result of LDRANK applied to the corresponding web page (see section 4.2.3.3).

**Stephen Hawking** W



**Stephen William Hawking** CH CBE FRS FRSA (/ˈstiːvən ˈhɔːkɪŋ/; born 8 January 1942) is an English theoretical physicist, cosmologist, author and Director of Research at the Centre for Theoretical Cosmology within the University of Cambridge. His scientific works include a collaboration with Roger Penr...

**Context**

[Stephen Hawking takes a trip through a black hole - CNN.com](#)  
 CNN Leadership **Stephen Hawking's** fantastic vision of **black** holes may solve 40-year **paradox** Updated 1[...]


Hide Caption **Stephen Hawking** takes a trip through a **black** hole - [CNN.com](#) Want to know what happens a[...]

▼ Read more...

**Related Concepts**


**Theoretical physics**

[...] 11:00 am Comment Share This article For the past few decades, **black** holes have been at the center of a paradoxical problem — a problem famed **physicist** Stephen Hawking now believes he's solved. [...]



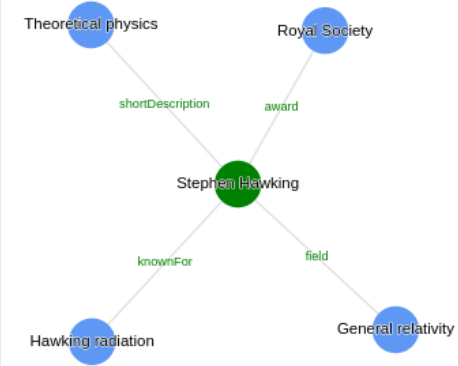
**Black hole**

[...] **Stephen Hawking** may have finally solved the **black hole** 'information' problem



▼ Read more...

**Concept's Graph**



```

graph TD
    SH((Stephen Hawking)) ---|shortDescription| TP((Theoretical physics))
    SH ---|award| RS((Royal Society))
    SH ---|knownFor| HR((Hawking radiation))
    SH ---|field| GR((General relativity))
    
```

▼ Show full graph

FIGURE 3.2: The Entity Description panel



- **Details-on-demand:** select an item or group and get details when needed.
- **Relate:** view relationships among items.
- **History:** keep a history of actions to support undo, replay and progressive refinement.
- **Extract:** allow extraction of sub-collections and the query parameters.

The correspondence between these guidelines and the interactivity of the semantic snippets is listed hereafter:

- **Overview first:** listing the results (the semantic snippets) and the top-ranked entities.
- **Zoom:** zoom in on a snippet and show more details about it (more text or more top-annotated entities), show more on a given entity by clicking on it.
- **Filter:** filter out the results using the top-ranked entities.
- **Details-on-demand:** select an entity and get all the details about it, in addition to “load more” results when needed.
- **Relate:** view the relationships among entities (*Entity Description - Related Concept*), and between an entity and the text (*Entity description - Context*).

For now, we do not support the **History** functionality. Thus, we do not have a user profile that would enable us to keep the history of the user’s actions and to support actions such as undo, replay, and progressive refinement. We also do not support the **Extract** functionality. However, these functionalities are among the future improvements we envision for the system (see section 5.2.5).

### 3.5 Semantic Snippets’ Generation

Given the query, we obtain the SERP from the search engine. For each result of the SERP, we extract the text, annotate it to have a list of the LD entities. We transform this list into an RDF graph by connecting the entities using RDF links.

Each entity is associated with a text (abstract and windows from the text), thus, we have an RDF graph enriched by textual data. We use this data structure as an input to LDRANK algorithm to obtain a ranking of the entities. The top-ranked entities (using LDRANK) will be employed in the *Primary Concepts* component (see Section 3.3).

---

Then, we extend the RDF graph using a SPARQL endpoint to increase the number of triples among which we will then search for the most important ones (using tensor analysis). That allows us to associate each top-ranked entity with a set of relevant triples that will appear within its description (*Entity Description*).

Finally, we used a machine learning approach (see next section) to select an excerpt from the web page for the textual snippet and short excerpts for the description of each top-ranked entity.

## 3.6 Learning to rank documents' excerpts

We propose to use a machine learning strategy to discover web pages' excerpts likely to highlight interesting relationships between the user's information need and the entities from the top of the LDRANK ranking. In other words, we use the text of the web pages where the entities have been found to contextualize the entities. Thus, we are building a bridge between the web of data and the web of documents.

We reduce the problem of associating an entity with excerpts from a web page to a binary classification problem. Thus, for each web page's excerpt with a surface form of an entity discovered by LDRANK, one should decide whether or not this excerpt is likely to highlight an interesting relationship between the entity and the user's information need.

We start this section by giving some background information about the learning algorithms we used (section 3.6.1). Section 3.6.2 introduces existing works that address similar issues. In section 3.6.3, we describe the features we used for learning. Then, in section 3.6.4, we explain how we obtained the training dataset. We describe in section 3.6.5, how we managed the imbalanced dataset to make the two classes comparable. In section 3.6.6, we discuss the results obtained with different learning algorithms while using all of the features. Then, in section 3.6.8, we introduce the strategies that we applied to limit the number of useful features while preserving the quality of the classification. In the same section, we discuss the new results (after feature selection). Finally, in section 3.6.10 we stress the importance of the features derived from LDRANK.

### 3.6.1 Background

#### 3.6.1.1 Supervised Machine Learning

Supervised machine learning is the process of learning a function from examples. This function should, with a high probability, have a low generalization error (i.e. it will be able to predict on new data). The process of applying supervised ML to a problem consists of the next steps:

1. Considering the problem, we start by identifying the required data.
2. Data preprocessing.
3. Definition of the training set.
4. Algorithm selection.

5. Training.
6. Evaluation with test set.
7. If the Evaluation gives good results, we got our *Classifier*. If not, we can go back to the *Training* step and do some parameter tuning, or we can go back to any other step and try to change our method to work (e.g. choosing a different algorithm).

The first step is to collect labeled data. Depending on the problem at hand, it can be challenging to collect enough labeled data so that the function selected by the learning algorithm will have with a high probability a low generalization error (cf. the Probably Approximately Correct (PAC) theory of learning proposed by Leslie Valiant [Valiant, 1984]). As introduced later, we use a crowdsourcing- based collecting strategy to solve this problem.

Next is the data preparation and preprocessing, such as handling the missing data, outliers detection. If domain experts are available, then they could choose which fields (attributes, features) are the most informative. If not, we can measure everything available hoping that the informative and relevant features can be isolated later. In this case, the collected dataset may contain noise and missing feature values, and therefore, it requires significant preprocessing.

The *Definition of the training set* is the process where we take all the collected data and try to optimize it as a training dataset. Optimizing the collected dataset can be done by handling the main two aspects of the data, i.e. the features and the instances.

The features optimization step is based on the method of *Feature subset selection*, i.e. the process of identifying and removing as many irrelevant and redundant features as possible. This method will reduce the dimensionality of the data. Dimensionality reduction is often crucial since with fewer features, the learning-algorithm will need less data to discover , with a greater probability, a function with low generalization error.

In the case of features dependency, a method called *feature construction/transformation* can be used (see section 3.6.1.4), one constructs new features from the basic feature set, and the newly generated features may lead to the creation of more concise and accurate classifiers.

The second aspect for optimizing the dataset is *Instance Selection*, where one tries to handle the noise, and at the same time, attempts to maintain a small generalization error while minimizing the sample size. Thus, *Instance selection* reduces the size of the data and enables a learning algorithm to function effectively with very large datasets.

The choice of a specific learning algorithm (see section 3.6.1.2) is critical. Classifiers' evaluation is based on a given metric, e.g. prediction accuracy (the percentage of correct prediction divided by the total number of predictions). The two main techniques to evaluate a classifier and calculate its accuracy are: (1) split the training set and use two-thirds of the data for the training and the other third for estimating performance, (2) *Cross-Validation*, where the training set is divided into mutually exclusive and equally-sized subsets and for each subset, the classifier is trained on the union of all the other subsets. The average of the error rate of each subset is an estimate of the error rate of the classifier.

In the *Intelligent Systems* domain, the *Supervised classification* is considered as one of the most common approaches. Thus, a large number of classifiers have been developed based on Artificial Intelligence (Logical/Symbolic techniques), Perception-based techniques and Statistics (Bayesian Networks, Instance-based techniques).

In the next section, we present a short background about the classifiers that we consider as relevant to our context.

### 3.6.1.2 Classification Algorithms

**Logistic Regression** Logistic regression is a discriminative approach to build a probabilistic classifier. The idea is to directly fit a model of the conditional distribution  $p(y/x)$  where  $y$  is the dependent variable, i.e. the label to predict, and  $x$  is the independent variable, i.e. the vector of features), whereas, with a generative approach, one first creates a model in the form of a joint distribution,  $p(y,x)$  before conditioning on the independent variables.

Logistic regression is more adapted to binary classification than linear regression since the dependent variable is supposed to follow a Bernoulli distribution instead of a Gaussian distribution. It is a linear model since it depends on a linear combination of the independent variables. But this combination is mapped to  $[0,1]$  through the logistic function. The decision rule can be determined with a threshold on the output probability (i.e. the dependent variable) to for example 0.5.

**The Naive Bayes Classifier** This classifier is based on the Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods. The Naive Bayes classifier tries to reduce this complexity by making a conditional independence assumption that reduces the number of parameters to be estimated when modeling  $P(X—Y)$  [Mitchell, 1997].

**C4.5** This algorithm has been developed by Ross Quinlan to generate a decision tree. It comes from an earlier system called ID3 and is followed in turn by C5.0. C4.5 builds decision trees from a set of training data using information entropy. J48 is an open source Java implementation of the C4.5 algorithm in the Weka data mining tool [Quinlan, 1993].

**Radial basis function networks (RBF)** RBF networks were introduced into the neural network literature by [Broomhead and Lowe, 1988]. The RBF network model is motivated by the locally tuned response observed in biologic neurons. The theoretical basis of the RBF approach lies in the field of interpolation of multivariate functions.

**Support Vector Machines (SVMs)** SVM is a popular machine learning method for classification, regression, and other learning tasks. It has been shown to be highly effective at traditional text categorization. The basic idea behind the training procedure is to find a hyperplane, that not only separates the document vectors in two classes, but for which the separation, or margin, is as large as possible. LIBSVM is an open source Java implementation of the SVM algorithm in Weka [Pang et al., 2002].

### 3.6.1.3 Imbalanced Classes Problem

A dataset is imbalanced if the classification categories are not approximately equally represented. Data sets with imbalanced class distributions are quite common in many real applications, and the degree of imbalance varies from one application to another. In [KrishnaVeni and Sobha Rani, 2011], the authors summarize the most commonly used techniques to handle this problem as the following:

- **Undersampling.** This method removes examples from the majority class to make the data set balanced. The drawback of the undersampling method is that it discards potentially useful information that could be important for classifiers.
- **Oversampling.** Oversampling is a sampling approach that balances the data set by replicating the examples of the minority class. It is also called *upsampling*. The advantage of this method is that there is no loss of data as in undersampling technique. The disadvantage of this technique is it may lead to overfitting and can introduce an additional computational cost.
- **Cost-Sensitive Training.** Cost Sensitive Learning (CSL) is another commonly used approach. It uses an asymmetric cost function to balance artificially the training process.

- **One Class Learning.** When negative examples greatly outnumber the positive ones, certain discriminative learners have a tendency to *over fit*. A recognition-based approach provides an alternative to discrimination where the model is created based on the examples of the target class alone. This approach attempts to measure the amount of similarity between a query object and the target class, where the classification is accomplished by imposing a threshold on the similarity value.

An important mixed approach is SMOTE; it is a sampling method designed to address the problem of imbalanced classes. It combines two methods, the under-sampling of the frequent classes and over-sampling of the minority class. In [Chawla et al., 2002], the authors compared SMOTE to other alternative sampling techniques on several real world problems using several classification algorithms. They showed that SMOTE performs better than methods like Rippers loss ratio or Naive Bayes, and it can improve the accuracy of classifiers for a minority class.

#### 3.6.1.4 Features Selection

Features selection can improve the prediction performance by providing faster and more cost-effective prediction by selecting a subset of the features. This selection also gives the prediction more flexibility against overfitting. It is used to find an “optimal” subset of features such that the overall accuracy of classification is increased while the data size is reduced and the comprehensibility is improved.

Feature selection approaches can be divided into two categories:

**Filters Approach** The methods of this approach select the optimal subset of features as a preprocessing step, independently of the chosen classifier. A filter method scores and ranks each feature on a particular metric, then it selects the top-k features as the new machine learning features set.

For example, one can use the information gain method. It measures the information carried by a single attribute:  $InfoGain(Class, Attribute) = H(Class) - H(Class|Attribute)$ , where H is the information entropy.

Thus, the information gain is the reduction in the entropy of the random variable representing the class achieved by learning the value of the attribute under consideration.

Filters can be used to reduce the dimensionality and overcome overfitting. They are faster than other approaches, and they provide generic feature selection. On the other side, filters methods have three main weaknesses: (1) they do not consider features

redundancy, i.e. two redundant features will have the same score, and will be selected (or not) together. (2) they do not detect dependencies between features. (3) they do not propose an ideal subset of features, leaving this task to users.

**Wrappers Approach** This type of methods need the knowledge of a chosen learning algorithm to score subsets of features. The used score is the prediction performance of a subset. The main idea is to build a space of all possible combinations of features, and search this space using the prediction performance as a guide. Many search strategies can be used (best-first, branch-and-bound, ...), the validation is usually done by cross-validation in order to assess the prediction performance of the selected subset.

Wrappers methods can be of two types:

(1) *forward selection* where we start with a small subset of features and progressively add more features.

(2) *backward elimination* where we start with all features, and iteratively eliminate the weaker ones.

This iterative strategy for building subsets of features allows wrappers to take into account dependencies between features. This was not possible with filter-based approaches. The only drawback of wrappers is the performance: the search space being the power-set of the features, an exhaustive search would be prohibitively costly, therefore heuristics must be used to approximate the optimal solution. Trivial greedy heuristics are often proposed.

When the feature selection is integrated with the training, we speak of **embedded wrappers**.

We can note that, with the wrapper methods, there is no need to define the initial number of features.

### 3.6.2 Related Work

We start by mentioning the works that address a problem similar to ours, viz. selecting web pages' excerpts given a user's query. It should be noted that our context also includes a semantic layer made of the LOD entities discovered in the web page and ranked with LDRANK. For this reason, our problem differs from the ones of the state of the art. However, it remains close enough so that we will benefit from situating our proposal in the setting of these related works. Thus, we introduce the related works that rely on a supervised learning strategy to address the problem of the selection of web pages' excerpts given a user's query.

In [Wang et al., 2007], to design their features, the authors use both the content of the web page and the contextual information that can be obtained from the values of the



anchors for the hyperlinks pointing to the web page. In this work, the granularity of the excerpt is the sentence. They measure the relevance of a sentence by counting the number of occurrences of terms that are also present in the query. They measure the importance of a sentence by counting: (i) the number of occurrences of terms that are among the most frequent in the web page, (ii) the number of occurrences of terms that are also present in the web page's title, and (iii) the number of occurrences of terms that are also present in the anchors of the web page's incoming links.

The authors compare two learning strategies. For the first strategy, they reduce the problem of selecting the best sentences to a binary classification problem that they can solve by training an SVM classifier. To take into account the imbalanced number of instances in each of the two classes, the authors split into two components the parameter that sets the penalty for the classification's errors. Thus, they can use a different factor for each of the two classes. For the second strategy, they adopt a learning-to-rank approach. They learn a ranking function that, *for a given web page* of the training dataset, should place in top positions the relevant and important sentences. To do this, they use the SVM ranking algorithm.

The authors validate their approach on the dataset provided by the web track of TREC-2003. They randomly select ten queries. For each query, they keep ten relevant pages, five irrelevant pages chosen among the top-ranked ones (from the point of view of the BM25 model for information retrieval), and five irrelevant pages selected randomly. For each page, two human workers wrote a summary by keeping the best sentences (usually three of them) given the query and its context (i.e., a detailed description offered by the TREC dataset).

For the two strategies introduced above, the authors use a Gaussian kernel for the SVM. The gaussian's standard deviation is fixed with a 3-folds cross-validation approach. They used the following quality metrics: precision, recall, and F1-measure. The best results are attained when taking into account the context (through the incoming hyperlinks' anchors) and when using a learning-to-rank strategy.

In [Metzler and Kanungo, 2008], the authors take an approach similar to the one introduced above. However, they compare three kinds of learning-to-rank algorithms (Ranking-SVM, Support Vector Regression, and Gradient Boosted Decision Tree). Also, they add some features that are independent of the query (e.g., sentence's length and position within the page). Finally, they experiment on a much larger dataset (viz. TREC Novelty Track 2002, 2003, and 2004). The best results were achieved by using the Gradient Boosting Regression Tree (GBRT) algorithm.

In [Ageev et al., 2013], the authors improve on the results obtained in [Metzler and Kanungo, 2008] (see above). Thus, they also use the GBRT algorithm. However, they introduce behavioral features (e.g., the time spent by the mouse cursor over a text fragment, the time during which a text fragment was visible on screen, etc.). To build a training dataset with these additional features, they used the Amazon MTurk crowdsourcing platform. The experiment was introduced to the worker in the form of a game: he had to answer correctly, in a limited time, to as many questions as possible. Moreover, in order to efficiently register the behavioral features, the authors did some preprocessing on the dataset. First, they kept only the web page's sentences that contained at least one of the query's terms. Next, they split the sentences with a flying window of at least three words (the exact number of words is a parameter of their proposal), and they keep only the fragments where at least one of the query's terms is present.

Finally, in [Lehmann et al., 2012], the authors propose the DeFacto algorithm to check the veracity of a LOD RDF triple by discovering web pages' fragments that confirm the fact expressed by the triple. At the core of this work, there is again a supervised machine learning approach. This proposal is based on BOA, a system designed by the authors of [Lehmann et al., 2012] in one of their previous work. BOA associate a natural language pattern to an RDF predicate. First, during a preprocessing step, the authors remove all but the sentences that contain the subject/object's labels of the predicate of which the veracity is being checked. The learning features include the presence of a BOA pattern in the sentence, the distance between the subject's label and the object's label, etc. Furthermore, this work introduces a set of metrics to quantify the trust one can have in a web page given the RDF triple that is being checked. We will not detail this part of their work since it is not related to our problem.

From this study of the related works, we gained some insights about which features to select in our context. In the next section, we introduce these features.

### 3.6.3 Feature Engineering

Specific to our approach is the use of LOD entities detected in a web page. As introduced previously, to each entity we associate a label, a summary (obtained from the DBpedia abstract of the entity), the entity's neighbors in the RDF graph built from the entities detected in the web page, and the score computed by LDRANK.

We designed features among which some of them are using the data coming from this semantic layer. We now introduce the features we used as raw data for a supervised learning algorithm that, given a selected entity, will find the sentences conducive to

explaining the relationship between the selected entity and the user's information need (as expressed by his query). We group the features into four categories:

- independent of the query and of the selected entity,
- dependent on the query, but independent of the selected entity
- dependent on the selected entity
- dependent on other LOD entities detected in the sentence.

### 3.6.3.1 Features Independent of the Query and of the Selected Entity

This category of features exposes the sentence's readability, content, and location. It reflects how these aspects affect the selection of the sentence. It consists of the following features:

- (LEN) The sentence's length.
- (SSS) The number of short sub-sentences (i.e., the sub-sentences separated by punctuation marks with a length of less than four words).
- (SS, SE) The nature of the sentence's first character (alphabetic, numeric, other) and last character (dot, dots, etc.).
- (HL) The presence of hyperlinks in the sentence, a binary feature that equals to 1 if the text contains a hyperlink.
- (D) The presence of a date in the sentence, a binary feature that equals to 1 if the text contains a date.
- (SL) The sentence's position in the web page, normalized by the total number of sentences.
- (LS) Is it the last sentence? A binary feature that equals to 1 if  $S$  is the last sentence in the document.
- (NCH) The ratio of non-alphabetic to alphabetic characters.

### 3.6.3.2 Features Dependent on the Query but Independent of the Selected Entity

This category of features captures the relevance of a sentence to the query independently of the selected entity. It consists of the following features:

- (QT) The number of terms shared between the query and the sentence.
- (QR) The number of annotated entities shared between the query and the sentence.
- (KTKR) The number of entities detected in the sentence and among the top-k entities ranked by LDRANK.

### 3.6.3.3 Features Dependent on the Selected Entity

This category of features reflects how much this sentence is related (relevant) to a particular entity. It consists of the following features:

- (SEL) The number of terms shared between the entity's label and the sentence.
- (SEA) The number of terms shared between the entity's summary and the sentence.
- (SEN) The number of neighbors to the selected entity that are present in the sentence.
- (SENL) The number of terms shared between the sentence and the labels of the selected entity's neighbors.
- (SEP) The relative position of the entity in the sentence (normalized by the sentence's length).
- (SES) The LDRANK score for the selected entity (normalized by the maximum LDRANK score obtained by an entity in the same page).

### 3.6.3.4 Features Dependent on LOD entities Detected in the Sentence

As aforementioned, we annotate the text of the documents (each sentence) using DBpedia Spotlight, this annotation tool return a list of annotated entities in the documents. Then, we build an RDF graph from these entities, and we rank them using LDRANK. This category reflects the effect of having annotated entities in the sentence, the structure of the generated RDF graph and the entities' ranking (LDRANK). It consists of the following features:

- (AE) The number of entities detected in the sentence.
- (TKE) The number of entities ranked among the top-k by LDRANK.

- (AET) The number of terms shared between the sentence and the labels of the entities detected in the sentence.
- (TEL) The number of terms shared between the sentence and the labels of the top-k entities (from the point of view of LDRANK).
- (AES) Average LDRANK score for the entities detected in the sentence (normalized by the maximum LDRANK score obtained by an entity of the same page).
- (AEL) Given the RDF graph built from the web page: the number of predicates that connect together the entities detected in the sentence.

### 3.6.4 Training Dataset

For training purposes, we used the dataset already introduced in section 2.4.2. We only have to interpret this dataset from another perspective: “the relevance of an entity given a sentence and an information need expressed by a query” becomes “the aptness of a sentence to highlight a relationship between a detected entity and the user’s information need”.

Furthermore, we aggregate the judgments to reduce the problem to binary classification: a score of 0 (i.e., *irrelevant*) means that the sentence should not be selected, the other scores (1 for “marginally relevant”, 2 for “fairly relevant” and 3 for “highly relevant”) all mean that the sentence should be selected.

In Figure 3.3, we present an excerpt from the generated dataset, for example, we have the instance:

*Q0D0S0C\_http://dbpedia.org/resource/William\_Ewart\_Gladstone.*

It represents the resource *William Ewart Gladstone* that has been annotated in the first sentence *S0* in the first result (or document) *D0* when using the query *Q0*; this instance has been classified by the workers (aggregated result) as *Relevant*. Thus, we can say (from the new perspective) that the sentence *S0* in the document *D0* is relevant to the resource *William Ewart Gladstone* in the context of *Q0*.

Instance_ID	Relevant
Q0D0S0C_ http://dbpedia.org/resource/William_Ewart_Gladstone	true
Q0D0S1C_ http://dbpedia.org/resource/African_American	true
Q0D0S1C_ http://dbpedia.org/resource/Bookmark_(World_Wide_Web)	true
Q0D0S1C_ http://dbpedia.org/resource/Collecting	true
Q0D0S1C_ http://dbpedia.org/resource/Gramophone_record	false
Q0D0S1C_ http://dbpedia.org/resource/Hard_Times_and_Nursery_Rhymes	false
Q0D0S1C_ http://dbpedia.org/resource/Intimate_relationship	false
Q0D0S1C_ http://dbpedia.org/resource/Library	true
Q0D0S1C_ http://dbpedia.org/resource/Panorama	true
Q0D0S1C_ http://dbpedia.org/resource/Photograph	true
Q0D0S1C_ http://dbpedia.org/resource/United_States_Congress	true
Q0D0S1C_ http://dbpedia.org/resource/VMware_Horizon_View	false
Q0D0S1C_ http://dbpedia.org/resource/William_Ewart_Gladstone	true
Q0D0S2C_ http://dbpedia.org/resource/Moral_responsibility	true
Q0D0S2C_ http://dbpedia.org/resource/Rights	true
Q0D0S3C_ http://dbpedia.org/resource/Academic_publishing	false
Q0D0S3C_ http://dbpedia.org/resource/Bibliotheca_(Pseudo-Apollodorus)	false
Q0D0S3C_ http://dbpedia.org/resource/Collecting	false
Q0D0S3C_ http://dbpedia.org/resource/Distribution_(business)	false
Q0D0S3C_ http://dbpedia.org/resource/Filesystem_permissions	false

FIGURE 3.3: Example of the generated datasets

We did experiments with other aggregation strategies, but the best results were obtained with the one introduced above (see section 3.6.7 from more results of other aggregation strategies).

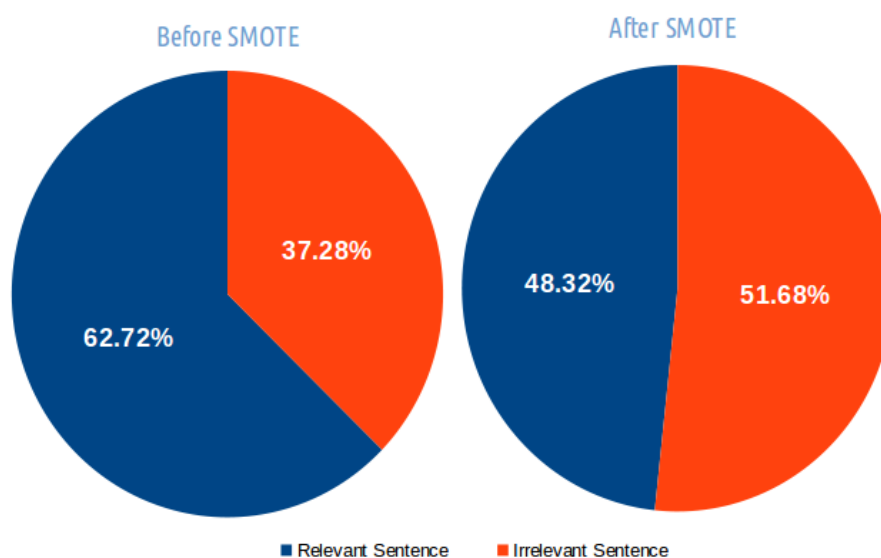


FIGURE 3.4: The unbalance of the dataset

### 3.6.5 Re-Balancing the Training Dataset

The training dataset includes 15841 instances of entities detected in web pages (see Figure 3.4). 9936 of them belong to sentences apt to illustrate the relationship between the entity and the information need. Thus, 5905 instances of entities appear in

a context which is not appropriate to explain the relationship between the entity and the information need. Thus, the positive class represents 62.7% of the total number of instances in the training dataset. We correct this slight imbalance by applying the SMOTE (Synthetic Minority Over-Sampling Technique) algorithm to grow the minority class with synthetic data, and also reduce the size of the majority class. The application of SMOTE over our dataset makes the two classes quasi-equals with 49.3% of the class relevant and 51.7% for the other class.

### 3.6.6 First Results

We did experiments with five supervised learning algorithms: logistic regression, naive Bayes (with a Gaussian distribution as a conditional distribution for numeric attributes), the J48 implementation of the decision tree algorithm *C4.5*, Radial Basis Function Network (RBF) and SVM (LIBSVM implementation, with a Radial Basis Function core with the nu-SVC version of the algorithm, and with a value of 0.5 for the nu parameter which sets a lower bound over the fraction of classification errors caused by a too large margin). We adopted a ten folds cross-validation approach. We used the Weka platform [Hall et al., 2009]. We used F1-score metric for the evaluation of the positive class (F-True), for both classes (F-All), and the area under the ROC curve. Table 3.1 presents the results of this experiment.

	F-True	F-All	ROC
<b>Logistic Regression</b>	<b>0.802</b>	<b>0.588</b>	<b>0.658</b>
Naive Bayes	0.759	0.633	0.644
J48	0.772	0.659	0.652
LIBSVM	0.802	0.537	0.588
RBF	0.802	0.584	0.631

TABLE 3.1: Results for learning to select sentences apt to explain the relationship between an entity and the user’s information need

### 3.6.7 Results using other regrouping strategies

We tried other strategies to re-group workers’ answers (from the crowd-sourcing). So instead of considering *irrelevant* as false and grouping the [*marginally relevant, fairly relevant, highly relevant*] as true (*relevant*), we generated two datasets: (1) *dataset-2* where we grouped [*irrelevant, marginally relevant*] as false (*irrelevant*) and [*fairly relevant, highly relevant*] as true (*relevant*); and (2) *dataset-3* where we grouped [*irrelevant, marginally relevant, fairly relevant*] as false (*irrelevant*), moreover we kept only *highly*

	F-True	F-All	ROC	F-false
Logistic Regression	0.528	0.604	0.655	0.675
Naive Bayes	0.357	0.523	0.634	0.695
<b>J48</b>	<b>0.689</b>	<b>0.697</b>	<b>0.772</b>	<b>0.704</b>
LIBSVM	0.606	0.644	0.644	0.68
RBFClassifier	0.528	0.606	0.654	0.679

TABLE 3.2: Results for *dataset-2*

	F-True	F-All	ROC	F-false
Logistic Regression	0.557	0.594	0.64	0.632
Naive Bayes	0.351	0.511	0.624	0.676
<b>J48</b>	<b>0.638</b>	<b>0.645</b>	<b>0.679</b>	<b>0.653</b>
LIBSVM	0.616	0.614	0.614	0.613
RBFClassifier	0.553	0.592	0.638	0.632

TABLE 3.3: Results for *dataset-3*

*relevant* as true (*relevant*). The results presented in the tables 3.2 and 3.3 shows that the adopted strategy (in section 3.6.4) gives better results than the others strategies.

### 3.6.8 Feature Selection

Given these first result, we try to reduce the number of the useful feature while maintaining the quality of the prediction. We are inclined to do feature selection for various reasons. First, with fewer features the prediction will consume less memory and can be faster. Computing the features' values can be expensive. Therefore, execution time can be gained during the prediction. Moreover, using only a subset of the original features can reduce estimation error and avoid over-fitting. Finally, in our case, we are especially interested in verifying indirectly the effectiveness of LDRANK: if the LDRANK-based features remain, it will be a good indication in favor of the usefulness of LDRANK.

We initially used a filter-based feature-selection approach. It means that we filter the features before using a specific supervised learning algorithm. Thus, we used the *information gain (InfoGain)* metric to rank the features. We tried two strategies: (i) keeping only the top-10 features, (ii) cutting at a *nosedive* in the features' ranking based on their information gain score (which led us to retain only 16 features, see the "Infogain (top)" line in Table 3.4).

We then used a wrapper approach for feature selection. In this case, we use the chosen learning algorithm as a black box, and we explicitly seek to minimize the error rate. Thus, naively, one should explore the whole space of features' subsets, and for each point in this space, one should execute the chosen learning algorithm. The advantages of this kind of approach over the blind filter-based one are (i) to be adapted to a specific



	J48	N.Bayes	Logistic	SVM	RBF
All var.	0.772	0.759	<b>0.802</b>	0.802	0.802
Infogain (top 10)	0.799	0.785	<b>0.803</b>	0.799	0.801
Infogain (nosedive)	0.792	0.771	<b>0.803</b>	0.801	0.801
Wrapper back. best-first	0.772	0.802	<b>0.804</b>	0.802	0.801
Wrapper back. greedy	0.787	0.802	<b>0.802</b>	0.802	0.802

TABLE 3.4: Feature selection results for (i) the filter approach with the information gain metric, and for (ii) the wrapper approach with two greedy strategies for exploring the space of features' subsets (values are the F1 score for the positive class (F1-True))

Nb var start	10	<b>6</b>	4
Nb var end	16	<b>13</b>	19
F-True	0.804	<b>0.804</b>	0.803
F-All	0.588	<b>0.587</b>	0.589
ROC	0.658	<b>0.658</b>	0.659

TABLE 3.5: Feature selection results using the forward selection wrapper approach with the logistic regression algorithm and while the number of initial features varies from 0 to 10 according to their information gain score

learning algorithm, and (ii) to take into account the dependencies between features. However, the cost of applying such an approach naively would be prohibitive. Thus, the search space will often be explored guided by a greedy heuristic. Two classic strategies will be considered: (i) the forward selection for which we start with a small subset of features that we grow progressively, and (ii) the backward elimination for which we start with all the features before removing the weakest one by one. We tested a naive greedy strategy (gradient descent without backtracking, see the line “Wrapper (Backward) - Greedy” in Table 3.4), and a greedy strategy with the possibility of backtracking by allowing the choice of at most two consecutive points of the search space that do not improve on the error rate (see the line “Wrapper (Backward) - Best First ” in Table 3.4).

We obtained the best results with the *logistic regression* algorithm. Thus, for this algorithm, we applied a forward selection wrapper approach. We made the number of initial features vary from 0 to 10 in ascending order of their information gain (see Table 3.5).

### 3.6.9 Selected Features

Thanks to this feature selection strategy, we reduce the number of features to 13:

1. (TEL) The number of terms shared between the sentence and the labels of the  $k$  best entities (LDRANK).

2. (SES) The LDRANK score of the selected entity (normalized by the maximum of the LDRANK scores obtained by the entities in the same page).
3. (SEL) The number of terms shared between the entity's label and the sentence.
4. (SEA) The number of terms shared between the entity's summary and the sentence.
5. (SEN) The number of neighbours of the entity that are present in the sentence.
6. (SENL) The number of terms shared between the labels of the neighbours of the selected entity and the sentence.
7. (SEP) The position of the entity in the sentence (normalized by the sentence's length).
8. (SE) The nature of the last character (dot, dots, etc.) of the sentence.
9. (QT) The number of terms shared between the query and the sentence.
10. (AE) The number of entities annotated in the sentence.
11. (AET) The number of terms shared between the sentence and the labels of the entities annotated in the sentence.
12. (SL) The position of the sentence in the web page (normalized by the total number of sentences).
13. (LS) Is it the last sentence? (binary feature)

We notice that we find among the selected features: (i) those based on LDRANK, and (ii) those depending on the semantic layer offered by the LOD entities detected in the web pages.

### 3.6.10 Conclusion

To conclude, we designed features based: (i) on the query, (ii) on the text of the web page, and (iii) on the entities ranked by our LDRANK algorithm. We applied a process of feature selection. First, we used the information gain metric to select a small subset of features. Next, we used this subset as a starting set for a forward selection wrapper approach. We observed that the features based on LDRANK were systematically selected. We consider it an indirect element of proof for the usefulness of LDRANK.

## Chapter 4

# Enhanced Search Engine (ENsEN)

## 4.1 Introduction

In this chapter, we present *ENsEN*<sup>1</sup> (Enhanced Search Engine), a software system that we built to generate *Semantic Snippets*.

By introducing ENsEN, our intention is to convince of the usefulness and efficiency of LDRANK.

We start by describing in details the software design and the architecture of ENsEN, then we present a user evaluation of ENsEN by crowdsourcing.

## 4.2 Software Design and Architecture

This section describes in details (i) the external services that ENsEN uses to enhance the search results, (ii) the system architecture and how it can be decomposed into components and subcomponents, (iii) the data model we use to interchange data inside the system, (iv) and finally the full workflow of the system.

In order to give standard description of the system design, we adopted the Unified Modeling Language (UML)<sup>2</sup> for all the diagrams of this section (e.g. Component diagram, Activity diagram, Deployment diagram...).

### 4.2.1 External Services

ENsEN relies on multiple external service providers exploiting the web of documents, the web of data, and even lexical sources such as WordNet (see Figures 4.1 and 4.2). In the following, we describe these services and how ENsEN employs them.

**Google Custom Search.** It is a service proposed by Google to give the user the possibility of creating a search engine for a website, a blog, or a collection of websites. In ENsEN, we are interested in this service because it allows us to retrieve the SERP for a given query and customize the look and feel of search results. Custom Search Engine comes in two flavors: (1) Custom Search Engine: it is free but limited by the number of possible queries per day, and (2) Google Site Search: a paid service (per query). As ENsEN is a prototype for academic purposes, we used the free version to retrieve the results list as input to our system of snippets generation.

---

<sup>1</sup>a *live* demonstration is available online <http://liris.cnrs.fr/drim/projects/ensen/>

<sup>2</sup>UML v2.0: <http://www.uml.org/>

**WordNet.** a lexical database for English. It contains nouns, verbs, adjectives and adverbs grouped into sets of cognitive synonyms (synsets). WordNet consists of 117000 synsets; these synsets are interlinked by conceptual-semantic and lexical relations giving us a network of meaningfully related words and concepts. The main relation among words in WordNet is the synonymy (words that denote the same concept and are interchangeable in many contexts) [Miller, 1995], [Oram, 2001]. ENsEN uses WordNet to extend the query by adding synonyms to its keywords.

**The DBpedia Knowledge Base.** As Wikipedia has grown into one of the central knowledge sources, maintained by thousands of contributors, the DBpedia project leverages this gigantic source of knowledge by extracting structured information from it and by making this information accessible on the web as a general knowledge base. The English DBpedia describes 4.58 million things, with 4.22 million classified in a consistent ontology. DBpedia also provides localized versions in 125 languages [Lehmann et al., 2014]. ENsEN uses and interacts with DBpedia indirectly using the two following services (Spotlight and SPARQL Endpoint).

**DBpedia Spotlight.** It applies named entity extraction (entity detection and name resolution or disambiguation) on a text in order to annotate mentions of DBpedia resources, with the objective of providing a solution for linking unstructured information sources to the Linked Open Data cloud through DBpedia [Mendes et al., 2011], [Daiber et al., 2013]. DBpedia Spotlight associates with each entity a *relevance score*. This *score* is calculated by a disambiguation model using individual scores: (1)  $P(\text{entity})$  the probability of the entity, (2)  $P(\text{surface form}|\text{entity})$  the probability of the entity knowing the surface form and (3)  $P(\text{context}|\text{entity})$  the probability of the entity knowing the context (text around the surface form).

$$\text{Score}(e, sf, context) = P(e) \times P(sf|e) \times P(context|e).$$

ENsEN uses the DBpedia Spotlight API to annotate and extract structured data from the unstructured text of a web page.

**DBpedia SPARQL Endpoint.** DBpedia can be queried via an SPARQL Endpoint; it is a web interface over OpenLink Virtuoso (a multi-purpose and multi-protocol Data Server). ENsEN uses this web interface to query DBpedia using SPARQL queries to find relations between the annotated resources and to build a connected RDF graph. ENsEN also uses it to get more information about a resource, such as its abstract, label, etc.

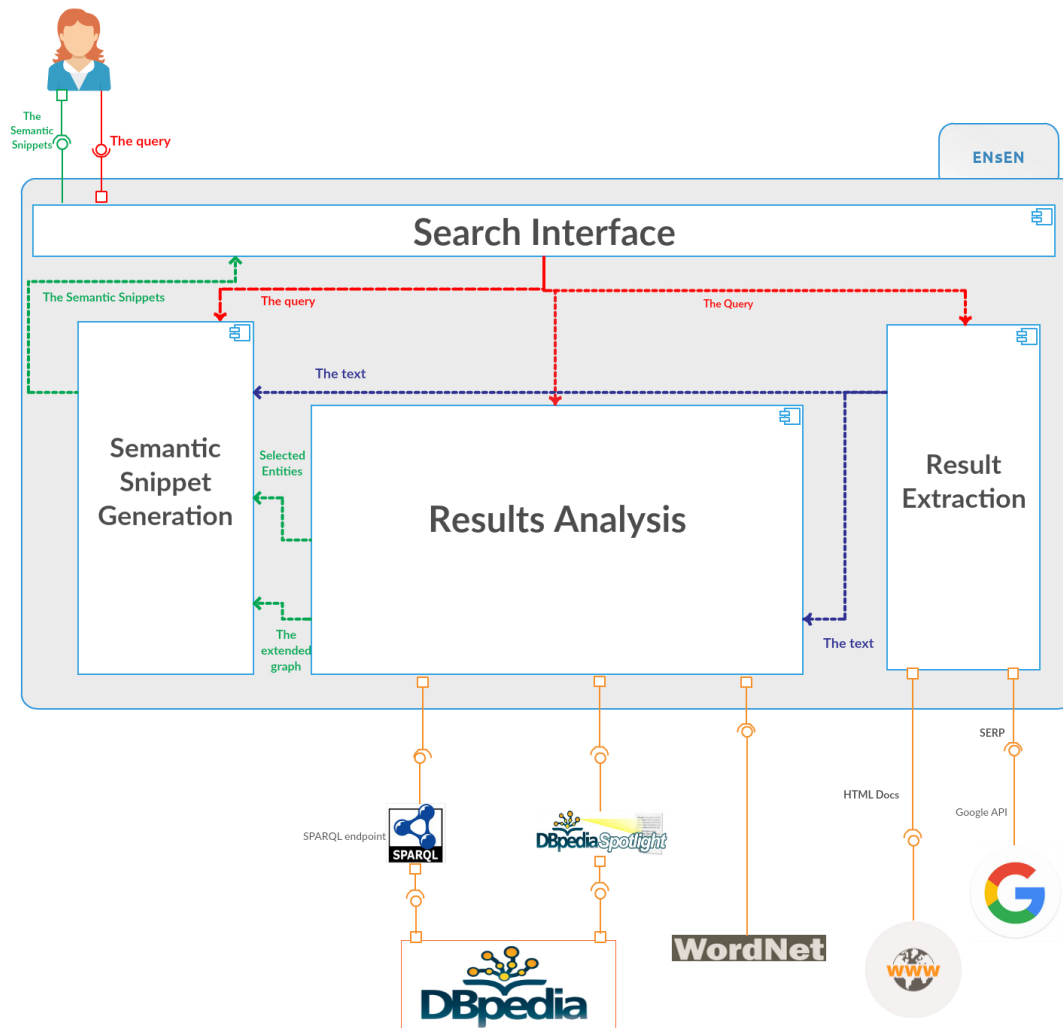


FIGURE 4.1: ENsEN Context and High-level Architecture (Component Diagram)

## 4.2.2 High-level Architecture

At a high level, our architecture follows and adapts the generic architecture used by most of the systems enhancing a given search engine (see Figure 4.1). The architecture is document-centric and extensible, it consists of four high-level components, each of which communicates with the external world and with other components (using APIs) in order to achieve its (required) task.

Since the primary purpose of a component diagram is to show the structural relationships between the elements of a system, and with the external components, we selected it to represent our architecture. The detailed architecture is shown in Figure 4.2.

## 4.2.3 Internal Components

Now, follows a brief description of each component (and sub-component) of ENsEN:

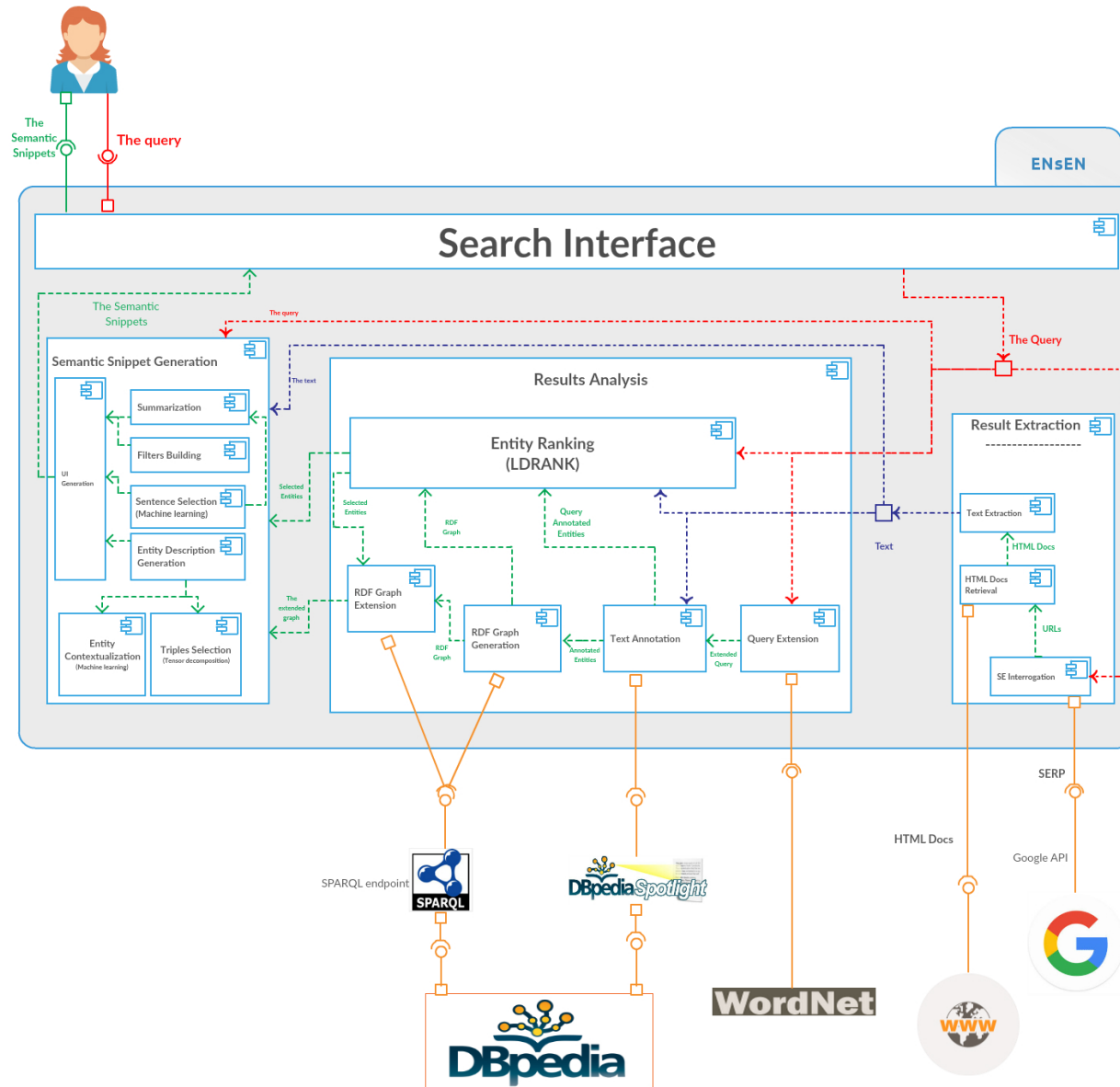


FIGURE 4.2: ENsEN Full Detailed Component Diagram

### 4.2.3.1 Search Interface

The *search interface* is a User Interface (UI) component; it serves as a communication layer between the system and the user. This component has three principal functions: (1) transfer the user's query to the system's internal components; (2) show the generated semantic snippets to the users; (3) capture and react to users' interactions.

### 4.2.3.2 Results Extraction

As a black box, the *Results Extraction* component takes a keyword query and returns the text of each result. More details will be discussed in the workflow section 4.2.5.

The primary processes (sub-components) of the *Results Extraction* are the following:

**SE Interrogation.** Its role is to communicate with a web search engine (in the implementation we used Google SE) and retrieve the SERP, from which it extracts the URLs of the results. It can use an API provided by the SE. Otherwise, it has to extract the results' URLs directly from the HTML code of the SERP.

**HTML Docs Retrieval.** It retrieves a web page using its address (URL) and extracts the actual HTML code of the page. The HTML documents are retrieved using the HTTP protocol.

**Text Extraction.** It applies the *Boilerpipe* algorithm [Kohlschütter et al., 2010] to obtain the web page's textual content by detecting and removing the surplus "clutter" (boilerplate, templates) around the main textual content.

### 4.2.3.3 Results' Analysis

This component encapsulates the core process of our system and its preprocessing steps; its objective is to generate for each result a ranked (considering the query) list of linked data entities. Besides, it creates for each result an extended RDF graph that interlink the entities and represent the result.

**Query Extension.** ENsEN is a keyword SE. Queries often consist of two or three words. This process extends the query by adding synonyms retrieved from WordNet synsets.



**Text Annotation.** In this process, we use the automatic annotation service of DBpedia Spotlight to obtain a set of DBpedia entities. In the same way, we find entities from the terms of the extended query. The input of this process is the text (to be annotated); the output is a list of entities in which each entity has a relevance score (sent by DBpedia spotlight).

**RDF Graph Generation.** This process generates an RDF graph from a set of entities, by issuing SPARQL queries to an endpoint connected to the DBpedia dataset. These queries introduce all the triples (statements) where subject and object are in the input set of entities. The retrieved triples (also called RDF links) allow to transform the set of entities into an RDF graph.

**Example:** *Tim Berners-Lee*<sup>3</sup>, *England*<sup>4</sup>, *London*<sup>5</sup> and *World Wide Web Foundation*<sup>6</sup> are entities in the input set, we can generate the corresponding RDF graph (see Figure 4.3) using the following query:

---

```

1  /*Define Prefix*/
2  PREFIX dbp: <http://dbpedia.org/property/>
3  PREFIX dbr: <http://dbpedia.org/resource/>
4  PREFIX dbo: <http://dbpedia.org/ontology/>
5  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
6
7  /*Construct a graph from the results statements*/
8  CONSTRUCT { ?subject ?predicate ?object }
9  where
10 {
11   ?subject ?predicate ?object.
12   /* restrict the graph to the entities of the input set*/
13   FILTER(
14     (
15     ?subject IN
16       (dbr:Tim_Berners-Lee, dbr:England,
17       dbr:London, dbr:World_Wide_Web_Foundation)
18     )
19     and
20     (
21     ?object IN

```

---

<sup>3</sup>[http://dbpedia.org/resource/Tim\\_Berners-Lee](http://dbpedia.org/resource/Tim_Berners-Lee)

<sup>4</sup><http://dbpedia.org/resource/England>

<sup>5</sup><http://dbpedia.org/resource/London>

<sup>6</sup>[http://dbpedia.org/resource/World\\_Wide\\_Web\\_Foundation](http://dbpedia.org/resource/World_Wide_Web_Foundation)

```

22     (dbr:Tim_Berners-Lee , dbr:England ,
23     dbr:London , dbr:World_Wide_Web_Foundation)
24 )
25 ).
26 }
27 }

```

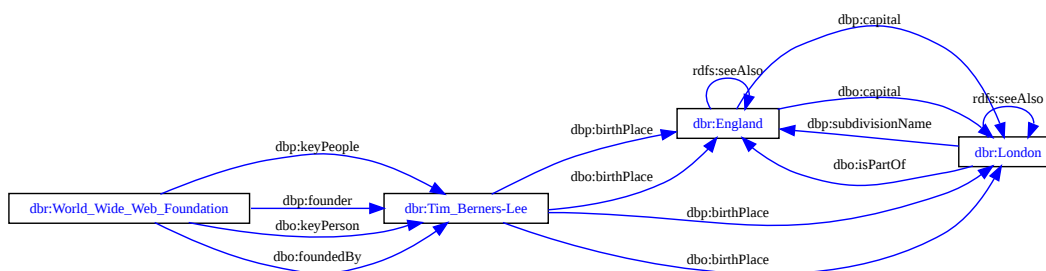


FIGURE 4.3: Example of a generated RDF graph

Each entity in the generated graph is associated with a short text. This text is obtained by merging the entity’s DBpedia’s abstract (extracted using a SPARQL query) and windows of text (in the current implementation we use windows of 300 characters) extracted from the web page and centered around the entity’s surface forms (associated to the entity by DBpedia Spotlight).

**Entity Ranking.** This component represents the core of our system. It encapsulates the application of our algorithm LDRANK presented in the chapter 2.

The inputs of this process are the user’s query (as keywords and annotated entities) and the results extracted from the SERP and processed to obtain for each result the raw

textual data and an RDF graph of the entities detected in the text. Moreover, additional textual data is associated with the entities if these RDF graphs.

Before applying LDRANK, some preprocessing steps are needed. We remove all the stop words and use a stemming process <sup>7</sup> on the query's keywords and the text associated with each entity. Then we generate two matrices: (1) the matrix  $[Entity \times Entity]$  that represents the connectivity of the entities in the RDF graph, and (2) the matrix  $[Term \times Entity]$  that represents the terms' frequencies for the text associated with the entities. We also detect the entities present in the stemmed query. These two matrices, and the information relative to the query represent the input of our algorithm. We then apply the LDRANK algorithm that ranks the entities of each result. Finally, we select the top-k entities to represent the result.

**RDF Graph Extension.** This process takes a poorly-connected graph and extends it by adding triples from Linked Data. However, a systematic 1-hop extension of all the nodes would produce a graph too large and too noisy. To address this problem, we do a 1-hop extension of only the top-ranked entities (according to LDRANK). We should note that multi-hop also can be used, but in this process, 1-hop extension is enough as our main objective is to find direct relationships between the entities annotated in the text, and to describe these entities by key-value information. We achieve this extension by issuing the next query to a DBpedia SPARQL endpoint and add the resulting RDF triples to the original RDF graph:

---

```

1  /*Construct a graph from the results statements*/
2  CONSTRUCT { ?subject ?predicate ?object }
3  where
4  {
5    ?subject ?predicate ?object.
6    /* restrict the graph to the top-ranked
7       entities [E1, E2, E3...] */
8    FILTER(
9      ?subject IN
10     (E1, E2, E3...)
11   ).
12 }
13 }
```

---

We do this extension for many reasons:

(1) The original generated graph is poorly connected. Therefore, we need the extension

---

<sup>7</sup><http://snowball.tartarus.org/>

to make it more connected.

(2) We would not benefit much from the LOD if we did not extend the graph to find facts not present in the original document. The original generated graph represents how the entities that have been annotated in the text are connected. With this extension, we collect more information about the entities (especially the textual RDF triples) that we use for the generation of an entity's description. We increase the number of triples among which we will then search for the more important ones to be shown (see *Triples Selection* in 4.2.3.4).

(3) We use the structured data from the extended RDF graph as features in our machine learning processes (see *Sentence Selection* 4.2.3.4 and *Entity Contextualization* 4.2.3.4) that learn how to select sentences from the document to describe an entity or summarize a document.

#### 4.2.3.4 Semantic Snippets Generation

This high-level component encapsulates the processes of generating the semantic snippets using the data produced by the *Results Analysis*.

**Filters Building.** We use the newly acquired knowledge of the most important entities to semantically restrict the list of results. We only keep the results within which the top-ranked entities appear (see left panel in the figure 4.7).

**Sentence Selection.** As mentioned in Section 3.3, our snippets include a short excerpt. This excerpt represents as faithfully as possible the relationship between the *Query*, the *Document*, and the *Top-ranked entities*. To select this excerpt, we use a machine learning approach that we discussed in chapter 3.6. This component represents the implementation of our machine learning approach; it takes the query (keywords and entities), the text and the top-ranked entities for each result. Then it splits the text into sentences using the algorithm *Boundary Analysis*<sup>8</sup>. For each sentence, we generate the features mentioned in 3.6.8, then we apply the machine learning classifier. The classifier return for each sentence a score that we use to rank the sentences of each web page (result) and select the best one for the excerpt.

**Entity Description Generation.** It generates for each top-ranked entity a full description (see figure 3.2). It is based on an RDF subgraph centered around the entity.

---

<sup>8</sup>developed by ICU project <http://site.icu-project.org/>

This subgraph is generated thanks to the *Triples Selection* process. It is also based on the entity’s context, built by the *Entity Contextualization*.

**Triples Selection.** Each top-ranked entity is associated with a set of explanatory triples that represent the best this entity. In this process, we introduce the tensor decomposition analysis (TDA) that ranks and selects the most relevant triples in an RDF graph given an entity.

An RDF graph consists of triples, as in [Franz et al., 2009], we propose to represent the RDF graph by a 3-way tensor (denoted  $T$ ). The three modes of the tensor are associated respectively with the subject, the object and the predicate of the triples that constitute the graph. Thus, for each predicate, there is one horizontal slice that represents the adjacency matrix of the subgraph only made of the triples that link the resources (the entities) thanks to that predicate. To analyze the tensor, we start by applying a PARAFAC [Kolda and Bader, 2009] tensor decomposition algorithm to the tensor. This decomposition computes a representation of the tensor as a sum of rank-one tensor (a rank-one three-way tensor is the outer product of three vectors), i.e.

$$T = \sum_{r=1}^R s_r \circ o_r \circ p_r$$

If there are  $n_r$  resources and  $n_p$  predicates, the lengths of each  $s_r$ ,  $o_r$  and  $p_r$  vectors are respectively  $n_r$ ,  $n_r$  and  $n_p$ . The components of the vectors  $s_r$  and  $o_r$  represent, for the number  $r$  factor, the importance of each resource as it plays respectively the part of subject and object in relations involving the high-scored predicates of  $p_r$ .

We should also note that there is no efficient way to compute the number of factors for a tensor decomposition. In [Franz et al., 2009], the authors used an implementation of the CORCONDIA heuristic-based algorithm [Andersson and Bro, 2000] to find a suitable number of factors. However, although we generate tensors in similar way, we discovered that the CORCONDIA heuristic did not apply well to our tensors. Thus, we found by experiment that, with the data we generate, the optimal number of factors is close to 10. Therefore, we test multiple decompositions with a number of factors varying around 10, and we keep the decomposition that offers the best fit, i.e. for which the recomposed tensor is closer to the original one.

The TDA results can be interpreted as groups, where each group has a set of ranked list of the predicates, subjects and objects of the graph. The high-scored predicates represent factors in the graph; they describe important sub-graphs. The high-scored subjects represent hubs in the corresponding subgraph. The high-scored objects represent authorities.

So, for each top-ranked entity (according to LDRANK), we select the factors to which it contributes the most (as a subject or as an object), and for each one of these factors we select the triples with the best-ranked predicates.

Thus, we can associate to each top-ranked entity a set of triples. We use this subgraph to build the entity's description.

**Entity Contextualization.** As aforementioned in the introduction of this chapter, the primary objective of the semantic snippets is to facilitate the knowledge transfer to the user. In order to take advantage of the way humans approach knowledge acquisition, we can assist the user in its answer's conceptualization by providing information about individual concepts and their context. The proposed context consists of the excerpts (from each result), where the considered entity was mentioned. Considering the limited space in the user interface and the possibility to have multiple contextual excerpts, we need to rank these excerpts and limit the context to the top ones. As we already ranked each document's sentences using machine learning (see section 4.2.3.4), we employ the same ranking to select the excerpts that will provide a context to the entity.

**Related Entities.** We associate each entity with two types of related entities; the ones related directly by an RDF link, and the ones related indirectly because they are mentioned in similar context (viz. they appear in the same sentences).

As for the previous component (viz. entity contextualization), considering (i) the limited screen space offered by the UI, and (ii) the possibility of many related entities, we use the LDRANK ranking to limit the number of related entities.

For each selected related entity, we show its label and the excerpts where it appears near the entity currently under consideration.

**Summarization.** Since for a same query, the results may cover many topics, the task of summarizing the results can be studied as multi-topic multi-document summarization. We generate query-dependent summaries centered on the topics of the main entities detected by LDRANK. The summaries, made of excerpts from the web pages where the entities were detected, are thus expressed in natural language.

To summarize documents containing multiple topics, we first need to define the topics and then rank them. In our case, a topic is defined by a set of entities. For each topic, we generate a summary by extracting the sentences specific to this topic, i.e. sentences contain the topic's entities.

As we have multiple topics, we score each one, and we show them in the order (most significant first), this ranking is query-focused; therefore it is based on the LDRANK ranking, and the search engine results ranking.

In order to define the score (importance) of a topic, we aggregate the importance of its entities:  $Score\_topic(t) = \sum_{e_i} Score\_entity(e_i) : e \in t$

An entity's score is proportional to its normalized rank in the document (according to the LDRANK ranking) and the rank in the search results' list, of documents within which this entity has been detected.

$$Score\_entity(e) = \sum_{D_i} \frac{LDRANK\_order(e)}{|D_i|} \times SE\_order(D_i) : e \in D_i$$

Where  $|D_i|$  is the number of entities annotated in  $D_i$ .

The final summary will consist of the selected sentences for each topic (in order of their importance). The entities appearing in the topic's summary are highlighted by the UI.

**UI Generation.** It is a user interface generating process that aggregates the output of the previous components (filters, textual snippet, and entity's description) and generates the corresponding UI elements.

#### 4.2.4 Data Model

As shown in figure 4.4, ENsEN data model consists of four main (data) tables (Query, Document, Entity, and Summary) and four secondary (data) tables (Sentence, Annotation, Topic, and Tensor). In the following we describe these data tables:

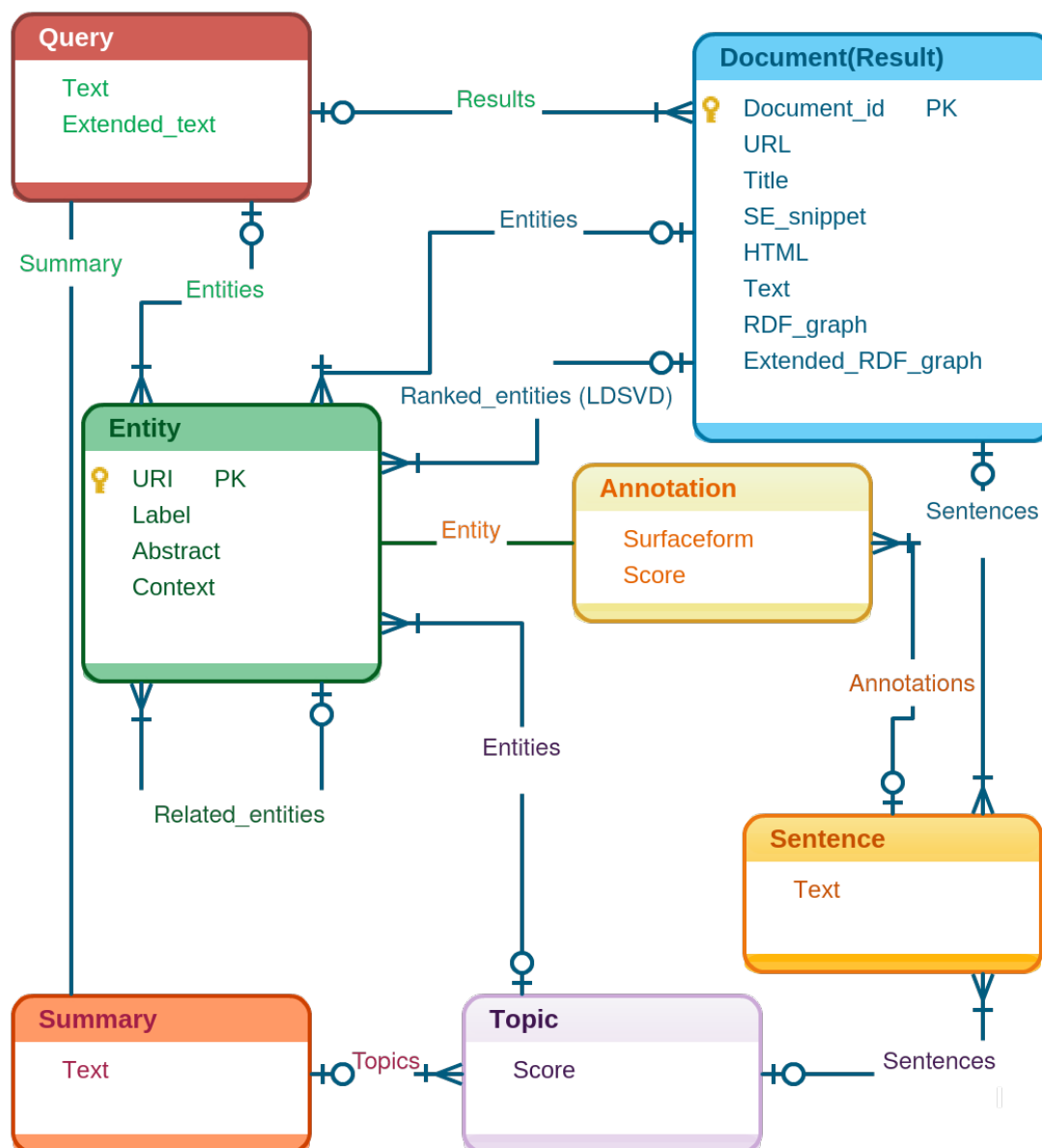


FIGURE 4.4: ENsEN Data Model Diagram

**Query.** It represents the user's query by (i) the *Extended Text* (from the *Query Extension* process), (ii) the *Entities* annotated by the *Text Annotation* process, (iii) the *Results* set retrieved by the *SE Interrogation* process and (iv) a *Summary* (textual) that summarizes the content of all the results (from the *Summarization* process).

**Document.** It represents the data structure for a result, and groups all its related data, that were either extracted from the web (of documents or data) or generated by the system.



At the beginning of a search session, this table contains the information from the web of documents, i.e., the *URL*, the *Title*, the *HTML* code, the extracted *Text* split into *Sentences* and the *SE Snippet* returned by the search engine.

After the annotation phase, the table's content is extended to include: the annotated *Entities*, the *RDF graph* and its extended version the *Extended RDF graph*. Applying LDRANK permit to add to this structure a list of *Ranked entities* and finally the mathematical representation of the graph, i.e., the *Tensor*.

**Entity.** It modelizes a Linked Data resource; thus it has a *URI*, a *Label*, an *Abstract* and a list of related resources (*Related\_entities*).

**Sentence.** Each document's text can be split into sentences, each sentence contains (thanks to DBpedia Spotlight) a set of *Annotations*.

**Annotation.** It is associated with an *Entity*, annotated in a *sentence* as *Surfaceform* (the original text snippet from which the association was induced) with a relevance *Score*.

**Summary.** The summarization process generates a *Textual* summary for each query. This summary consists of a ranked list of *Topics*.

**Topic.** It is defined as a group of *Entities*, annotated in a same *Sentence* and it has a relevance *Score* (see *Summarization* in section 4.2.3.4).

#### 4.2.5 Workflow

We present now the workflow and the activities that produce a semantic snippet from a query, and we highlight the role played by the LDRANK algorithm in this process.

From an architectural point of view, the snippet generation process follows the following steps (see Figure 4.5).

We should note that the activities are in **bold**, and the components are in *Italic*.

The user **enters the query**; the *Search Interface* sends this query to the *Results Extraction* process where (using the query) the *SE Interrogation* process **gets the results from the search engine** as a SERP (Search Engine Result Page). For each result (i.e.

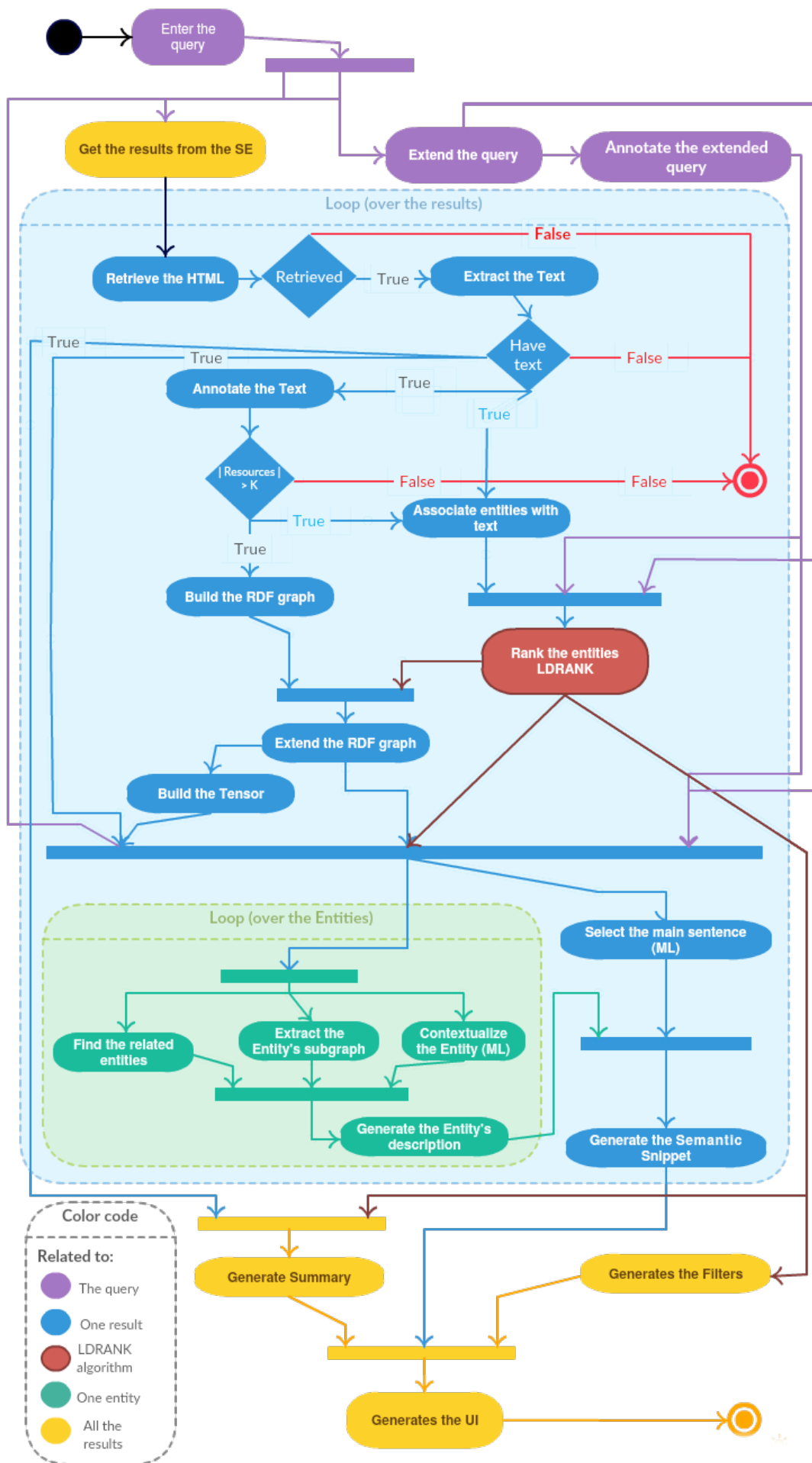


FIGURE 4.5: ENsEN Activity Diagram

a SERP's entry), first the *HTML Docs Retrieval* **retrieves the HTML** code from the web, if it was not able to **retrieve** the HTML it moves to the next result, else the *Text Extraction* **extracts the text** from this HTML.

In the same way, if there is not enough text, the system moves to the next result. Otherwise, the *Text Annotation* calls DBpedia spotlight to **annotate the text**, and the system **associates a text with each annotated entity** (see section 4.2.3.3).

In parallel with the text annotation, the *Query Extension* uses WordNet to find synonyms and **extend the query**, and then the system applies again *Text Annotation* to **annotate the extended query** in order to discover LD entities that are likely to be semantically close to the query, these entities will be used by the LDRANK algorithm.

The system ignores results with too few annotated resources (this threshold  $K$  is a system parameter). Otherwise, the *Entity ranking* process calls LDRANK to rank the entities annotated in the text of the results.

In parallel with the application of LDRANK, the *RDF Graph Generation* uses the list of the annotated entities to **build the RDF graph** by discovering the existing RDF links between the entities.

The *RDF Graph Extension* uses the top-ranked entities (by *Entity ranking*) to **extend** the RDF graph and **build the corresponding tensor**.

At this point, we have all that we need to build the semantic snippet and the new SERP: we have the query (text and entities) and the documents (text, ranked annotated entities, extended RDF graph and the corresponding tensor).

The *Semantic Snippet Generation* component takes this data and starts **generating the semantic snippet**. First, it uses a trained classifier to **select the main sentence** (*Sentence Selection*), and then for each entity, the system **extracts its subgraph** using *Triples Selection*, **find the Related Entities** and do the *Entity Contextualization* **forming the entity description**.

Finally, we use the top-ranked entities in each result, to **generate the filters** and the **summary**, and, eventually, to **generate the UI** for the new SERP.

### 4.3 User Interface

In Chapter 3, we introduced the elements of a Semantic Snippet (Section 3.3), and how our design follows the “information seeking mantra” guidelines (Section 3.4). In this section we describe the whole system's UI, especially the enhanced SERP.

ENsEN’s SERP has an interactive visual design, where we try to expose a great amount of information within a unified interface.

### 4.3.1 Visual Design

Our system is a web Application; this application has two main modules (pages): Query module and Results module. The query module (see Figure 4.6) was inspired by the design of the most used search engines (Google, Yahoo, etc.). In this interface we provide, in addition to the logo, simple input to enter the query and a simple go button.

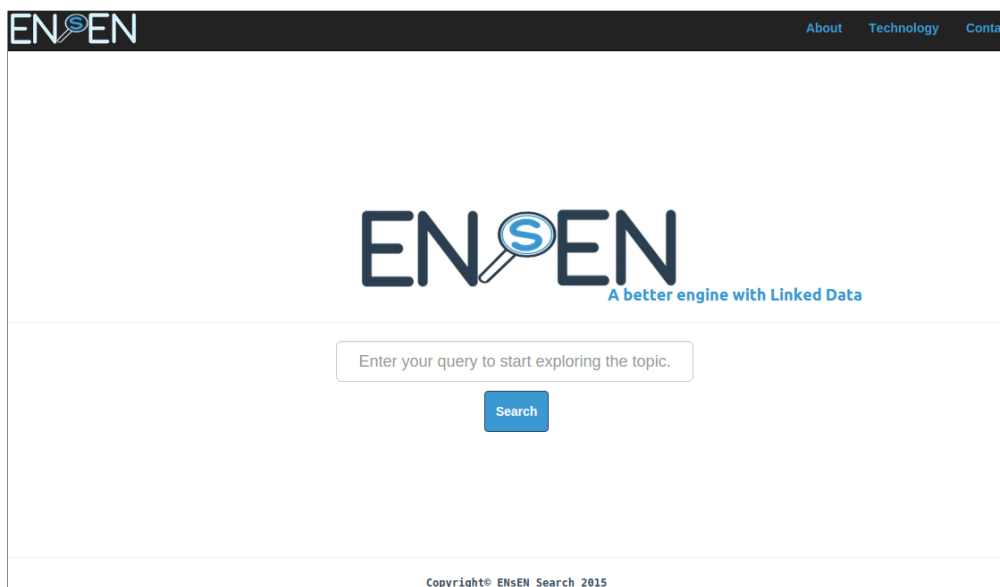


FIGURE 4.6: ENsEN Homepage, the query page

The results module (see Figure 4.7) consist of three panels:

- (1) the main panel in the middle encapsulates the generated semantic snippets,
- (2) the restriction panel at the left side encapsulates the filters and the summary,
- (3) the infobox panel at the right is used to show the *Entity’s Description* (presented in section 3.3).

### 4.3.2 ENsEN design and Crowdsourcing

In order to put the *user-friendliness* factor at the core of our visual design, we considered users’ feedback to guide our design. Thus, we asked the workers via the crowdsourcing platform (in addition to evaluating the design) to propose some interface enhancements “*In your opinion, what (and how) can we enhance in the ENsEN’s interface?*” , and to

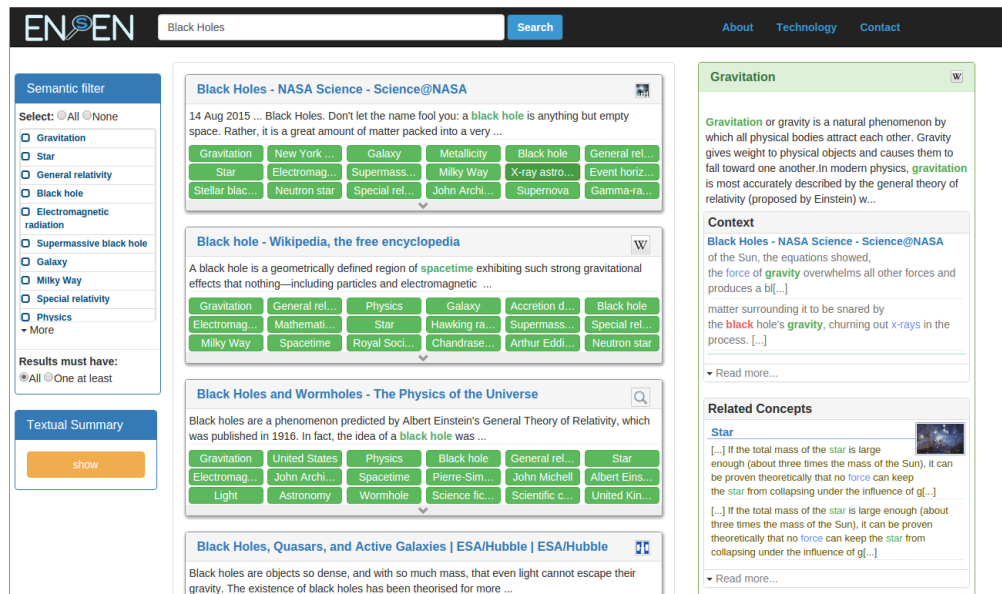


FIGURE 4.7: SERP of ENsEN

send us an open text feedback *“Please, give us any additional feedback about the user interface of ENsEN”*.

Most of the feedbacks were positives and encouraging, like *“I like the variety of concepts”*, *“Interesting Interface”*, *“It is a nice search engine”* and *“It is very promising”*.

However, we got a lot of feedbacks about the long response time (*“The Response time should improve”*, or *“More speed please”*), which is not related to the design but a result of using a server with limited resources.

Some workers found that the interface is too complex *“Maybe it is a bit unclear”*, *“Interesting but complicated”*, or *“the interface is sometimes confusing”*. Their feedbacks helped us to improve the system, and we understand that the workers found the interface complicated because, as one of the workers said *“ENsEN is not like Google. Its new and not familiar yet. It will take time to use and understand this interface”*.

In general, the workers were very interactive and sent us a lot of propositions, like:

- “Make it simple”
- “Try to make it faster and underline key words”
- “The color Red in EnsEN can be psychological bad, red is a hot color, needs others like blue or green”
- “Images are nice, yes ... but many images , colorful on one page usually mean Ads, so I ignore them”
- “Do some filtering”

## 4.4 Implementation

This section describes the design and implementation of our system (ENsEN). We implemented the ENsEN's prototype at *LIRIS*<sup>9</sup> laboratory. This prototype was tested by the laboratory research scientists internally, and used in all our crowdsourcing campaigns. It is also opened to external users for more testing.

This section starts with description of the current deployment architecture and explains our technological and architectural choices, as well as the current hardware configurations. We end by a description of potential extensions, and required improvements to transfer the current prototype into a more robust product.

### 4.4.1 Technological and architectural choices

As shown in the deployment diagram (Figure 4.8), our prototype follows the high-level architecture described in Section 4.2.2.

ENsEN is a web application; it was implemented to run over an *Apache Tomcat*<sup>10</sup> application server. Our application server is installed in a virtual machine under Linux Debian v6.0.10.

The current implementation uses the MVC (model-view-controller) software architectural pattern. In addition to the clear separation between the information, how to handle it, and how to present it, it provides a high level of flexibility and extensibility.

The web application (server-side) is implemented in Java, but in order to get high performance, the core algorithms have been implemented in *C* (for LDRANK) and *Python* (for the Tensor Decomposition Analysis).

The Search Interface is the front-end component of ENsEN. It was implemented using the latest technologies in web development: *JSP* for the server side and *HTML5*, *CSS3*, *JavaScript*, *Ajax*, *jQuery*, and *Bootstrap*<sup>11</sup> for the client side.

The system queries Google's index using the *Google Custom Search API*<sup>12</sup>, and retrieves the web pages using the *HTTP protocol*.

The system annotates the text by querying the *DBpedia spotlight* REST web service that answers in a custom *JSON* format. *DBpedia spotlight* is installed locally with the

<sup>9</sup> Laboratoire d'Informatique en Image et Systèmes d'information (LIRIS): <http://liris.cnrs.fr>

<sup>10</sup> Apache Tomcat: <http://tomcat.apache.org/>

<sup>11</sup> Bootstrap is a sleek, intuitive, and powerful first front-end framework for faster and easier web development.

<sup>12</sup> Google Custom Search: <https://developers.google.com/custom-search/>

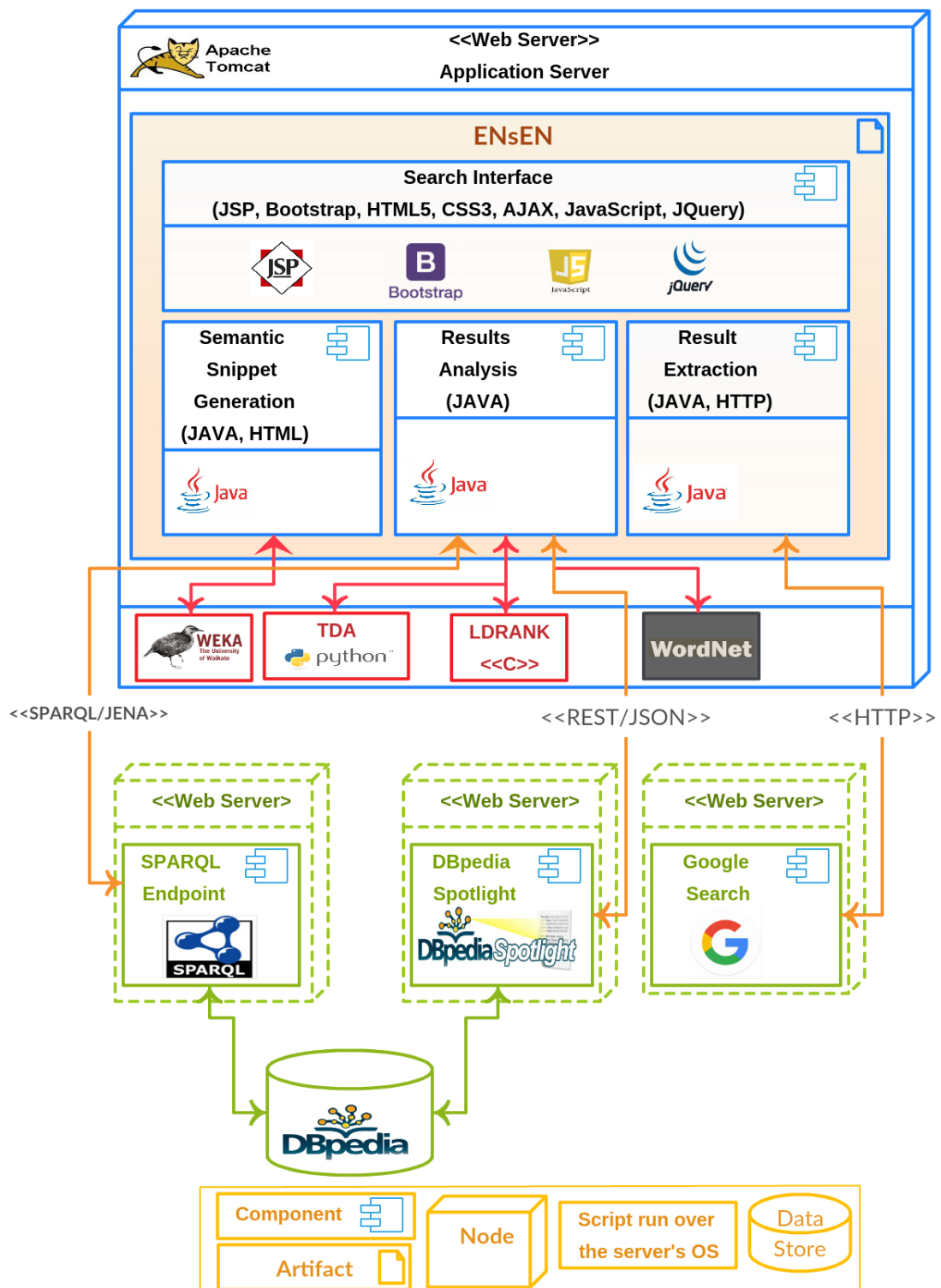


FIGURE 4.8: ENsEN Deployment Diagram

English dataset. However, the *SPARQL endpoint* is a remote one with also an English DBpedia dataset.

We use Apache Jena<sup>13</sup> to manage the RDF graphs, and to issue queries to the *SPARQL endpoint*.

WordNet<sup>14</sup> [Fellbaum, 2006], a large lexical database of English, was used for the lexical expansion of query.

The machine learning classifiers are implemented using the Weka<sup>15</sup> library for Java.

Most of the development was done using *IntelliJ IDEA*<sup>16</sup>, and *Apache Subversion*<sup>17</sup> for the versioning.

#### 4.4.2 Hardware Configurations

The ENsEN application and DBpedia Spotlight server run on a virtual machine with:

CPU Cores	8 CPUs x 1.87 GHz
Processor Type	Intel(R) Xeon(R) CPU E7520
Memory	31 GB
OS	Debian v6.0.10

### 4.5 Crowdsourcing Evaluation of ENsEN

We aim to evaluate the effectiveness and usefulness of our semantic snippets generation system, ENsEN, by comparing it to a traditional search interface, such as Google.

#### 4.5.1 Methodology

We adopt a crowdsourcing approach with the *CrowdFlower* platform (see section 2.4.2 for the description of this platform).

---

<sup>13</sup> Apache Jena: A free and open source Java framework for building Semantic web and Linked Data applications (<https://jena.apache.org>).

<sup>14</sup> WordNet: <https://wordnet.princeton.edu/>

<sup>15</sup> Weka is a collection of machine learning algorithms for data mining tasks: <http://www.cs.waikato.ac.nz/ml/weka/>

<sup>16</sup> IntelliJ IDEA is an IDE for enterprise, mobile and web development with Java, Scala, and Groovy from JetBrains: <https://www.jetbrains.com/idea/>

<sup>17</sup> Apache Subversion is an open source version control system <https://subversion.apache.org/>



#### 4.5.1.1 Task design

As our objective is to compare the interface and the quality of our system ENsEN with the ones of Google, the main task in this crowdsourcing campaign is to answer search questions like, “What division (weight) did the boxer Floyd Patterson win?” using both systems, then to evaluate and compare the two interfaces. Therefore, the first half of a participant’s tasks has to be answered using the Google search engine, and the other half must use ENsEN.

The requirements that guided us while designing the task are the following:

- The main requirement is to answer the search questions and to evaluate the interfaces.
- We need to evaluate the workers’ knowledge of search user interfaces in order to analyze the results according to the workers’ levels of expertise.
- In addition to the workers’ evaluation of the interface when answering a given search question, we need to have their overall evaluation of the system.
- We are also interested in analyzing the results according to the workers demographic information.
- We also want to benefit from this contact with the workers (as ENsEN’s users) to ask for any ideas about how to enhance the system.

In order to satisfy these requirements, we designed tasks made of the following parts:

- **Entry questionnaire** is a background information survey used to evaluate the worker’s knowledge of search interfaces.
- **Search tasks** is the main part, where we ask the worker to answer search questions.
- **Overall Preference:** Once the worker answered all of the questions, we ask for an overall comparison of the two search interfaces.
- **About you:** a short demographic survey.
- **Feedback:** At the end, we give the worker the possibility to express her feedback, impressions and any propositions.

The search questions have been extracted from the TREC 2004 QA dataset. We randomly chose 8 topics. Each topic includes three questions with their corresponding

answers. Two questions are factual, and the third expects an answer in the form of a list. (see all the topics and the questions in the Appendix [A.3](#)).

To determine the participant’s expertise in the domain of web search, we used questions such as:

**To find the pages that include an exact sentence, we should:**

- (i) surround the sentence with quotation marks,
- (ii) simply type the sentence as it is,
- (iii) place an asterisk at the end of the sentence?

A full task example is presented in the Appendix [A.2](#).

#### 4.5.1.2 Quality control

We collected 11 judgments for each of the 8 topics. We only accept the judgments for which the answer was provided within less than 30 min.

We keep only the best participants in terms of their score (metric) on the CrowdFlower platform. We briefly described this metric in section [2.4.2](#), it is based on an analysis of the answers to the test questions (i.e., questions for which the designer provided the correct answer).

We also applied a traditional method for controlling the quality in a crowdsourcing campaign, i.e., the **Test Questions**.

A test question is a search question that we ask using the crowdsourcing platform, and for which we provide the answer, allowing the system to control the workers’ performance. The test questions can be used in so called *Quiz Mode*, i.e. to keep only the “good” workers before offering them to enter the main task (the *Normal Mode*). They can also be used during the main task to monitor the workers’ performance and update their scores.

In our campaign, for each of the 8 topics, we generate a test question where the participants must answer using the Google search engine. Before they can be paid for answering questions, the participants must go through the Quiz Mode.

In this mode, the participants must answer the test questions, and if they make more than 50% of errors, they will be denied the access to the normal paid mode.

In normal mode, a page consists of two tasks offered to the participants: the first task they must answer with Google, the other one with ENsEN.

The work necessary to complete such a page (made of two tasks) is paid \$0.25.

### 4.5.2 Results Analysis

First, we found that most of the participants have a low experience in searching on the web (see Figure 4.9), 43.2% of the participants have an accuracy less than 0.25, and only 21.6% have a very good experience.

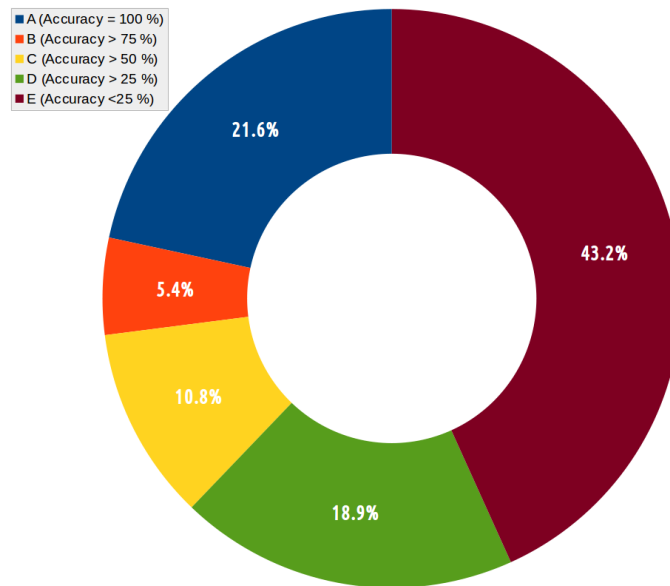


FIGURE 4.9: Participants' expertise for searching on the web, based on the accuracy of their answers to the questionnaire

But, we found that the accuracy of the answers does not depend on the level of expertise (see Figure 4.10).

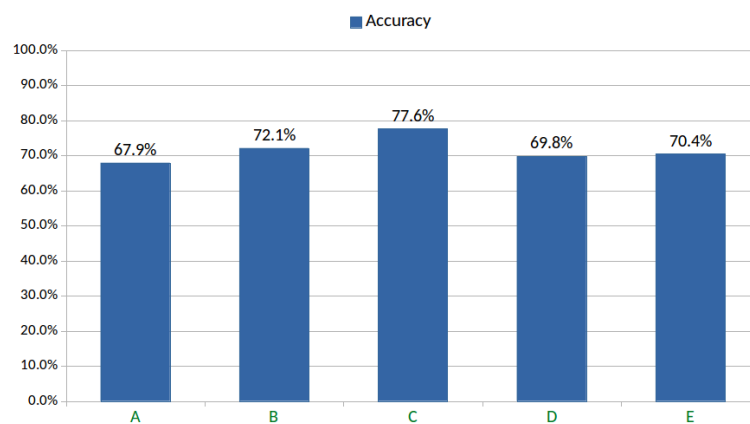


FIGURE 4.10: Accuracy of the answers depending on the level of expertise (A,B,C,D,E) of the participants for the web search domain

We observed that the accuracy of the answers is most often comparable for the two systems (see Figure 4.11).

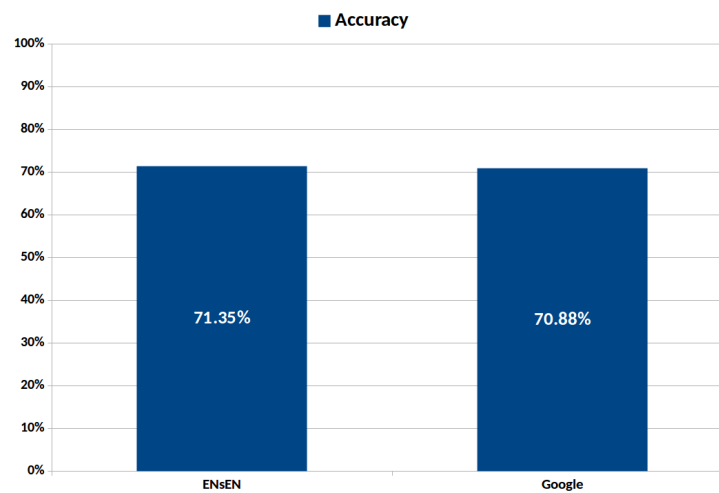


FIGURE 4.11: Accuracy of the answers per system

However, for some topics which obtained the lowest scores of accuracy, such as Topic 1 (Rhodes scholars) and Topic 7 (Crip gang), which are probably more difficult, ENsEN is significantly better than Google (see Figure 4.12).

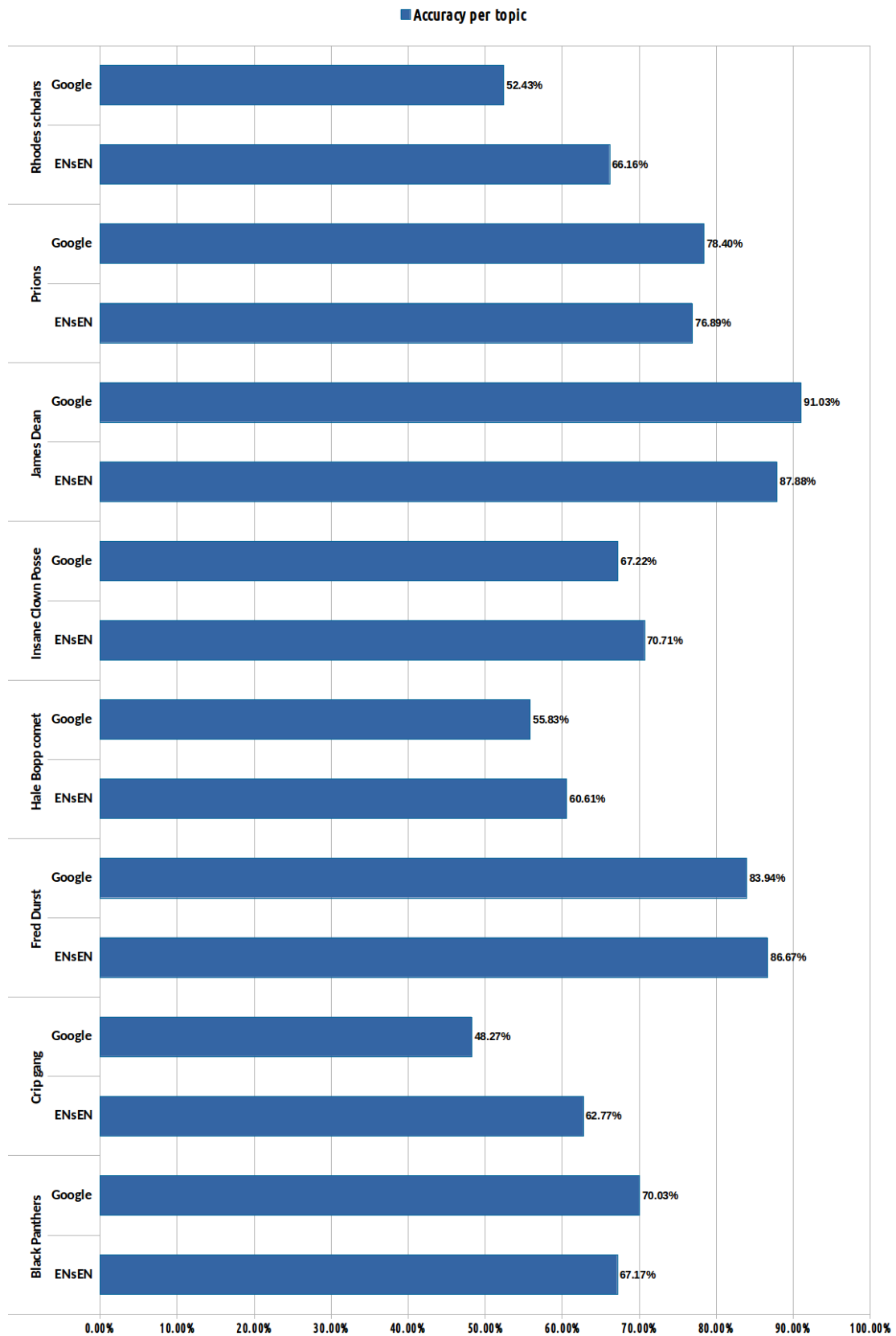


FIGURE 4.12: Accuracy of the answers by topic and for each of the two systems

Moreover, when we analyzed the answers per question type (i.e., factual questions Vs. list questions), we noticed that, even if the participants prefer the Google GUI, they feel that it is easier to find the correct answer using ENsEN (see Figures 4.13 and 4.14).

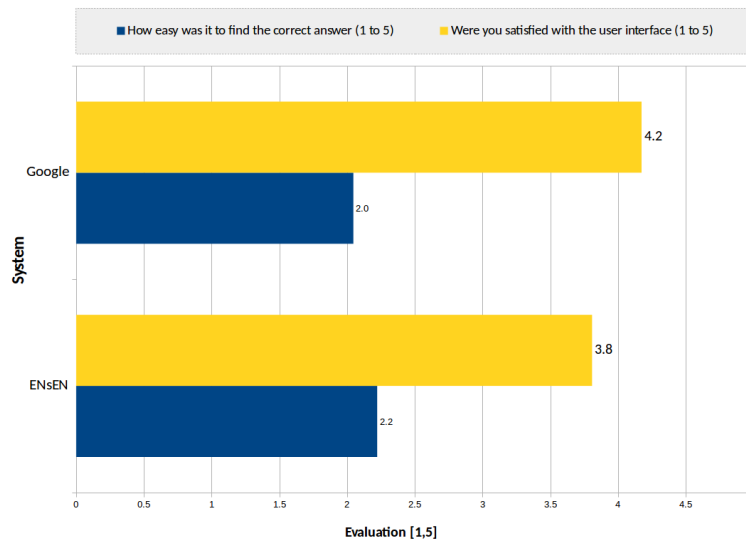


FIGURE 4.13: Overall evaluation of the effectiveness and the ease of use for the factual questions

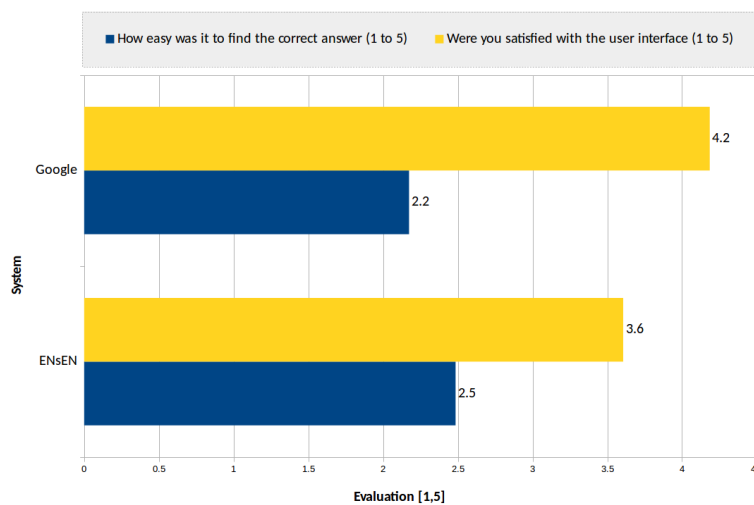


FIGURE 4.14: Overall evaluation of the effectiveness and the ease of use for the questions with a list answer

We also analyzed the collected data about the participants' age and their search habits (i.e., do they mostly use exploratory, informational, navigational or transactional queries?). And we found that most of the workers (86%) are between 20 and 40 years, and use mainly informational queries (67.68%). See Figures 4.15 and 4.16 for full results of this investigation.

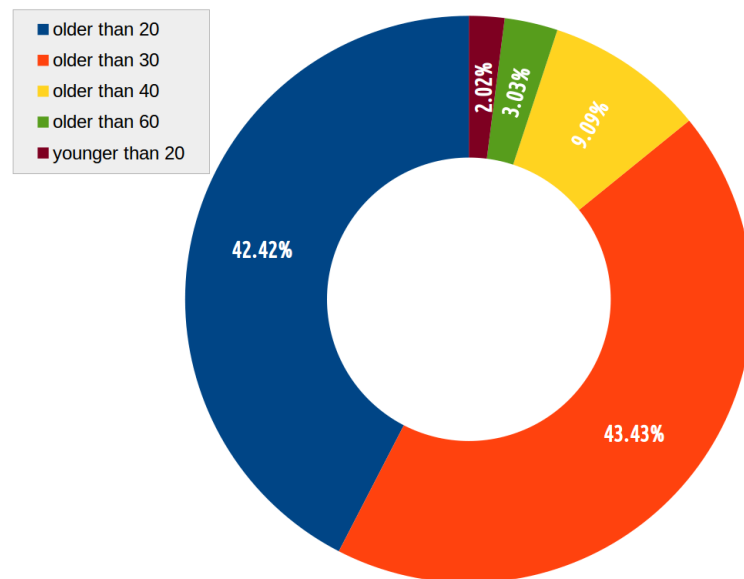


FIGURE 4.15: Ages of the participants

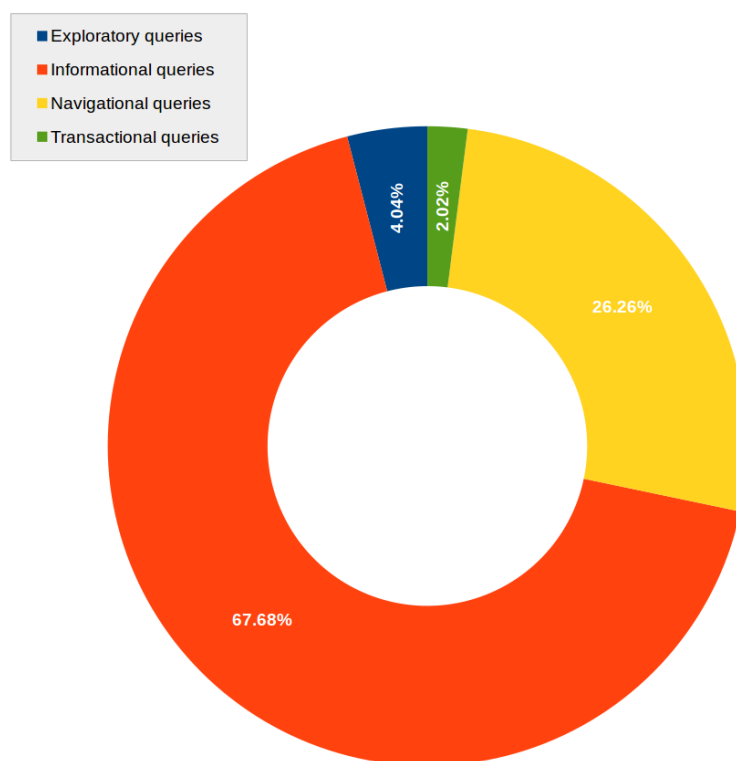


FIGURE 4.16: Participants' search habits in terms of the kind of queries they are most likely to perform

Finally, when we asked the participants about the elements of the ENsEN's interface they found the most useful, we discovered that the GUI elements that show the best entities discovered by LDRANK were highly valued (see Figure 4.17).

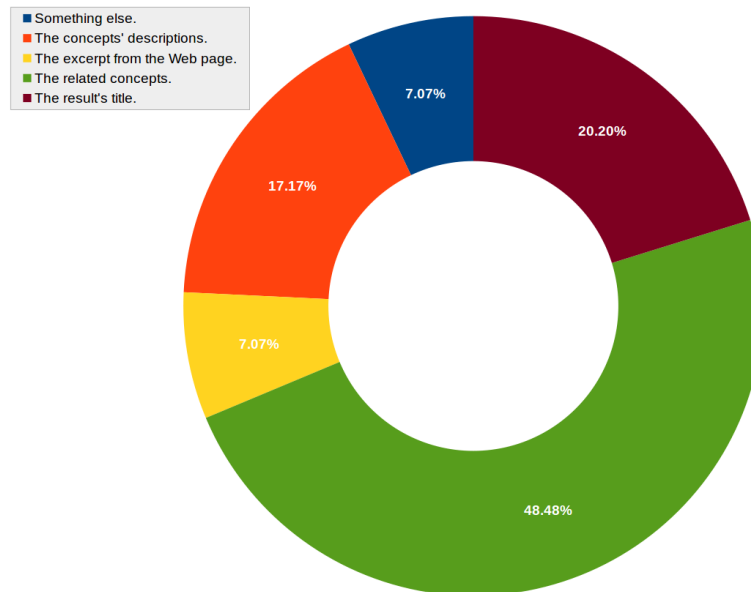


FIGURE 4.17: Preferred elements of the GUI

## 4.6 Conclusion

In this chapter, we built a complete system for generating semantic snippets. The experiments show the utility of these snippets to the final user. At the same time, this system proves the efficacy of the last two contributions, i.e. the ranking algorithm and the trained classifier.

In addition, the fact that we got good results in the experiments proves that integrating the web of data and the web of documents can be an interesting and useful approach.

Our proposal is only a step towards more efficient search systems. We can go farther by proposing extensions and applications. In the next chapter we discuss that in details.



## Chapter 5

## Conclusion

In the following conclusion, we summarize the context of our research, starting by a reminder of the principal research questions that we asked in the introduction, then recalling our contributions that aimed to answer these questions.

In a second time, we open the reflection about the future work, by listing the opportunities we see for the entity ranking and the semantic snippets generation, revealing the short-terms (and long-term) improvements we see for ENsEN.

## 5.1 Research summary

In this thesis, we have investigated the possibility of combining the linked data from the web of data, with the information from the web of documents for the purpose of enhancing user experience when interacting with a SERP. To do this, we introduced a new query-biased ranking algorithm for entities of the web of data.

Before summarizing the contributions of this thesis, we restate the three research questions that guided us:

1. *In an IR system, can we introduce a new, interactive, richer and more informative artifact to the SERP to replace traditional textual snippets? Can we employ the automatic annotation of Linked Data entities over the results' text for the construction of this artifact?*
2. *Given that automatic annotation identifies many entities in the text, how can we choose the most relevant ones? In other words, how to filter out the many irrelevant annotated entities? And how to select the relevant entities according to the original query?*
3. *How to objectively measure the impact of the new proposed artifact on the user experience?*

We now review our two main contributions in terms of the research challenges (questions) stated above.

- We propose LDRANK (Chapter 2), a query-biased ranking algorithm for the LOD entities discovered in the results of a traditional search engine through a process of automatic annotation. It consists in a modified PageRank algorithm that combines multiple sources of prior-knowledge through a linear consensual combination. This algorithm considers the query, the explicit structure of the graph, the latent

semantic analysis of the texts associated with the entities, and the ranking provided by a web search engine. The produced ranking is used to select the most relevant entities at the core of our enhanced semantic snippets.

This contribution answers the second research question: we can choose the most relevant entities with LDRANK, and then select the top-ranked ones.

- The second contribution (Chapter 3) proves (as an answer to the first question) that we can build high-quality semantic snippets while using linked data entities coming from the automatic annotation of the text of the web pages listed in a SERP.
- We take a machine-learning-based approach (Section 3.6) to discover within the web page excerpts that explain an interesting relationship between the most important entities and the user’s information need.
- As an answer to the third question, we present in Chapter 4 the ENsEN system that integrates the proposed algorithms in a system for generating Semantic Snippets that enrich the SERP. We also propose an evaluation methodology based on crowdsourcing to objectively measure the impact of the new proposed artifact.

In Figure 5.1, we illustrate our contributions and their research dependency. The proposed system *ENsEN* (Chapter 4) depends on an IR system (Search Engine) to retrieve the results for a given query. Then it generates the *Semantic Snippets* for the results. These snippets are presented in our second contribution (Chapter 3) as a new semantic and interactive artifact. This artifact consists of two main elements, the *Primary Concepts*, and the *Contextualizing Excerpts*.

The selection of the primary concepts is achieved by our proposed algorithm *LDRANK* presented as our first contribution in Chapter 2. This algorithm modify the *PageRank* algorithm by applying a *Belief Aggregation Strategy* (Section 2.3.4) on two prior knowledge sources; the first one (Entity Hit Ranking) is generated by applying the work of *Faflios and Tzitzikas* (Section 2.3.2) which itself is a modification of PageRank, and the second one generated by our proposed algorithm *LDSVD* (Section 2.3.3) that applies a *Text Analysis* process on the result’s text using *SVD*.

The contextualizing excerpts (shown in the *Main Sentence* and the *Entity Description*) were selected by a learning-to-rank process (Section 3.6) that analyzes the text and the *LD* annotated entities (via the *Automatic Annotation*) in order to rank and select the most relevant sentences.

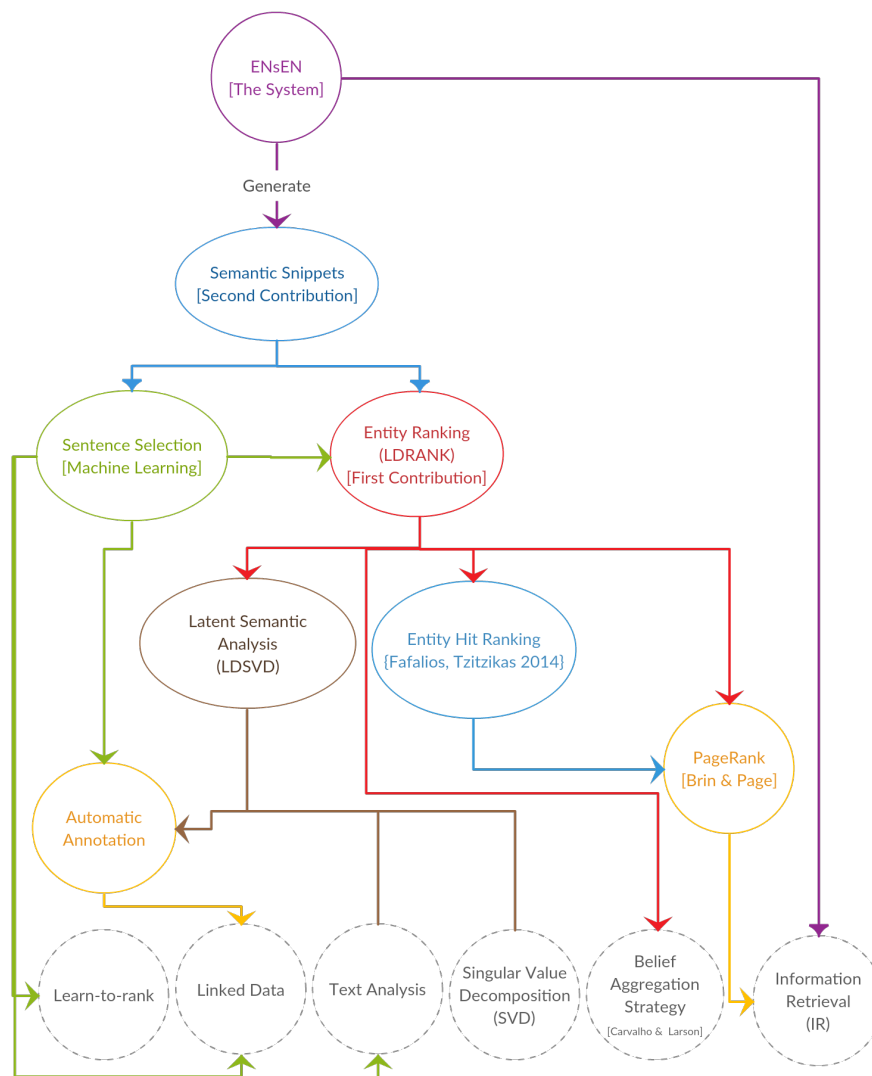


FIGURE 5.1: The research dependency

## 5.2 Future Work and Perspectives

We discuss some of the perspectives while focusing on two main aspects, the performance and the quality of the results.

### 5.2.1 LDRANK: Exploring more prior knowledge sources

One particular direction for the future work is the exploration of more sources of prior knowledge in the ranking of LOD entities.

In the first contribution (LDRANK), we integrated two sources; the ranking proposed by our algorithm LDSVD and the ranking proposed by Entity Hit Score [Fafalios and Tzitzikas, 2014].

As the combination strategy that we employ, i.e. the *Belief Aggregation*, is generic and accept multiple sources, we believe that exploring other prior knowledge sources may enhance the final ranking. Next, some additional sources that could be explored:

- **Entities' types ranking:**

In the public knowledge graphs such as DBpedia or Freebase, entities are associated with types (using predicates such as “rdf:type”). However, an entity is usually not associated to a single generic type but rather to a set of more specific types which may be relevant or not given the document context. For example, the entity “Tom Hanks” in DBpedia has about 40 types, such as “Person”, “Agent”, “Actor”, “Film Maker”, “Artist”, and “Intellectual”.

Some types may be too generic to be interesting (e.g., Person), while other may be interesting but already known to the user (e.g., Actor), or may be irrelevant given the current browsing context (e.g., Intellectual). We estimate that an entity with a relevant type is more relevant, and this kind of information can be considered as a good prior knowledge source. Thus, we can consider the relevance of an entity's type in our algorithm LDRANK.

In [Tonon et al., 2016], the authors propose an entity type ranking approach using a knowledge graph and the entity textual context. They suggest several methods exploiting the entity type hierarchy, collection statistics such as the popularity of the types or their co-occurrences, and the graph structure that semantically interconnects the related entities.

Other works that may also be interesting are: [Zaragoza et al., 2007], they rank entities of different types as a response to an open (ad-hoc) query, and [Rodríguez and Egenhofer, 2003] where the authors propose a model for semantic similarity across different ontologies. This similarity model provides a systematic way to detect similar entity classes across ontologies. It is based on a matching process for each specification components in the entity class representations (i.e., synonym sets, distinguishing features, and semantic neighborhoods).

- **Ranking model from LDRANK's state of the art:**

In the LDRANK's state of the art, we reviewed a set of entity ranking models. These models can be also considered as prior knowledge about the nodes importance. Among them, *PopRank* and *ReConRank* are the most adapted.

*PopRank*: It is a query-independent, link analysis model to rank objects within a particular domain. It takes into account the semantic of the relationships among different objects, by assigning a popularity propagation factor to each type of object relationship. Even if the ranking proposed by PopRank is independent of the user's information needs, it can be a relevant prior knowledge source.

*ReConRank*: It is a query-dependent ranking algorithm that ranks resources in a topical subgraph. This subgraph consists of resources matching the query (nodes directly linked to the query), and of a context (neighbor resources that can be reached after a specific number of hops in the graph).

### 5.2.2 Semantic Index

The Semantic Snippets generation process is completely online, therefore, we believe that an index will enhance the system's performance, and at the same time will help in personalizing the resulting snippets.

As in our approach the documents (the results) are represented by their text and their annotated entities, we think that a semantic index is necessary.

In semantic based IR, sets of words, names, noun phrases are mapped into the concepts they represent. A document is represented as a set of concepts or entities. To achieve this, external semantic structures for mapping document representations to concepts are needed. In our case, we use LOD cloud as a semantic structure and DBpedia Spotlight to map the document representations to entities in LOD.

The indexing approaches on the semantic based IR consists mainly of two phases:

(1) *Concepts Detection and Extraction* that gives for each document a set of representing concepts, and

(2) *index Construction* that groups all the preprocessing and analyzing operations allowing the construction of the semantic index.

Many works tried to handle this problem, they basically combine the knowledge representation and natural language processing techniques to accomplish the semantic indexing task.

In the following, we mention some of these proposed models.

(1) "*Semantic Indexing*":

This model store in the index more information about the indexed documents, to enable a system to retrieve documents based on the words, regarded as lexical strings, or based on the semantic meaning of the words. In [Mihalcea and Moldovan, 2000], the authors designed an IR system which performs a combined word-based and sense-based indexing and retrieval. They applied a disambiguation process relying on contextual information and identifying the meaning of the words based on WordNet senses. Therefore, the index is created using the words as lexical strings (to ensure a word-based retrieval), and the semantic tags (for the sense-based retrieval).

(2) *“Conceptual indexing”*:

On the contrary to the last model, where the index contains only some semantic tags that explain the sense of the indexed terms, this model is conceptual-based where the built index consists of concepts annotated in the documents.

It was introduced by [Woods, 1997], at Sun Microsystems Laboratories. They create some custom ontological taxonomies based on subsumption and morphology for the purpose of indexing and retrieving documents. In 2005, [Baziz et al., 2005a] and [Baziz et al., 2005b] also worked on the conceptual indexing, they build what they call the “Document Semantic Core” that represents the documents’ content by the semantic network. They start by extracting the concepts (mono and multiword) from a document, driven by external general purpose ontology (WordNet). Then, they build the best semantic network by achieving a global disambiguation of the extracted concepts regarding the document. Thus, selected concepts senses represent the nodes of the semantic network while the similarity measure values between them represent the arcs.

(3) *RDF-based semantic indexing*:

Thanks to the developments in the semantic web, researchers proposed a novel semantic indexing technique suitable for knowledge management applications. This technique represents the main concepts of a document in a RDF model, where several techniques could be used to transform a text document into RDF. [Amato et al., 2013] propose an approach that captures the semantic nature of a given document, commonly expressed in natural language, by retrieving the RDF triples (using NLP processing techniques, such as NER and POS tagging) and to semantically index the documents on the basis of the meaning of the triples’ elements (i.e. subject, predicate, object). They also propose to exploit this index by actual web search engines to improve the retrieval effectiveness with respect to the adopted query keywords or for automatic topic detection tasks.

In our case, and as we already applied the automatic annotation, we can use the resulted RDF graphs to index the documents. And as we have an algorithm to rank the entities of these graphs, we can, either integrate the resulted ranking in the index, or use the ranking to optimize this index.

### 5.2.3 SERP from documents to concepts

The LDRANK algorithm does not consider the inter-document (global) entity importance, nor the inter-document entity redundancy. These two aspects may be very interesting, and may enhance the quality of the ranking.

By breaking the document unit of information into smaller units (such as the concepts), we can group fro all the resulting documents (or a limited number of the results) the

concepts to build a huge knowledge graph. This graph can be analyzed using our proposed algorithm LDRANK to select the globally most relevant entities, independently of where they have been annotated.

Sometimes when a user employs a search engine in order to find some information, her first goal is to find this information no matter the document that contains it, retrieving the document or the information context comes in second place.

Therefore, we think that we may improve the user satisfaction, if we transform the SERP from being a viewer for an ordered list of results, to becoming a browser for concepts. These concepts are ranked using LDRANK, contextualized by documents' excerpts selected using machine learning and enriched using external information.

#### 5.2.4 Personalization and Recommendation

As aforementioned, we do not support the whole functionalities proposed by the “The information seeking mantra”. The *History* functionality can not be proposed because we do not have yet a user profile. Having a user profile allows for keeping the history of the user's actions and support actions like undo, replay, and progressive refinement. As our approach is entity-based, we can imagine the user profile to take the form of a RDF graph. This graph consists of entities extracted from the user's history. These entities are linked by triples extracted from the web of data or by applying NLP techniques over the documents in the user's history.

In addition, the user profile is necessary for other functionalities like the personalization and the recommendation.

Personalized IR is a popular topic in traditional web. The personalization aims at improving the user's experience by incorporating the user subjectivity into the retrieval system.

In [Sah and Wade, 2013], the authors propose a novel personalized search and exploration mechanism for the web of Data based on concept-based results categorization. The proposed personalization encloses the results re-ranking, the query refinement and concept lenses (results with the same concepts) suggestion.

Also in [Gauch et al., 2003], the authors employ the personalization to propose an information navigation based on a user profile structured as a weighted concept hierarchy. This profile may be built by the user by creating her own concept hierarchy, or using a reference ontology by “watching over the user's shoulder” while they browse. In this study, they show that these automatically created profiles reflect the user's interests quite well and they are able to produce moderate improvements when applied to search results.



In our work, the ranking process of chapter 2 can be extended to take the user profile as an additional source of prior knowledge.

In addition, we can apply the personalization at the machine learning level by also introducing the user's profile into the features. The sentence selection may then be more relevant.

As several research works have shown the great potential of linked data to compute semantic similarities [Blanco et al., 2013, Damljanovic et al., 2012, Passant, 2010, Zadeh and Reformat, 2012], these similarity measures are heavily used for the recommendation. Recommender systems suggest results and entities to the user that are not necessarily in the results list, but that may be interesting for her. This kind of system takes into account the user's history, priorities, and goals.

Thus, we can extend our model to include recommendations based on the user profile and the results of LDRANK. We can recommend entities or even documents depending on the similarity with the query and/or the results.

### 5.2.5 ENsEN from prototype to product

The ENsEN's architecture consists of separate components that interact using APIs. This type of architecture allows us to add as many extensions as we want. In addition to the extensions, we can propose some short-term, and long-term improvements for the current components transforming our system ENsEN, from a prototype to a real commercializable product.

In the following, we present the proposed enhancements and extensions for ENsEN:

#### 5.2.5.1 Enhance the performance by pre-treatment techniques

ENsEN runs 100% online, in order to understand better the system's performance, we studied its actual response time step by step. Therefore, we issued 3 different queries, in each one we generated the semantic snippets for the first five results. The results of this study are presented in Table 5.1.

We notice that the most expensive process is the *Tensor Decomposition*, applied to analyze the RDF graph and generate a set of semantic features to be used in the machine learning process, it takes **37.85%** of the total response time.

This time can be economized by applying the features selection methodology presented in Section 3.6.8, where we prove that the features generated by the tensor decomposition can be ignored.

Operation	Time (ms)	%	Functionality
Google	2449	2.55%	Send the query to Google and receive the results list
Extract HTML	20743	21.60%	Retrieve the documents by URLs
Extract Text	2400	2.50%	Parse the HTML and extract the main text
Annotate	5487	5.71%	Use DBpedia Spotlight to annotate the text
Build the RDF Graph	6734	7.01%	Use DBpedia Endpoint to connect the entities as an RDF graph
LDRANK	8987	9.36%	Apply the ranking algorithm LDRANK
Tensor decomposition	36343	37.85%	Apply the tensor decomposition to analyze the RDF graph
Build ML Features	6167	6.42%	Analyze the sentences and prepare the input (the features) for the machine learning
Find Main Sentence	5104	5.32%	Apply the machine learning process
Annotate SE Snippet	1600	1.67%	Use DBpedia Spotlight to annotate the SE's textual snippets
All operations	96014	100%	
Grouped results			
Preprocessing	37813	39.38%	Operations before LDRANK (prepare the algorithm's input)
LDRANK	8987	9.36%	
Tensor decomposition	36343	37.85%	
Snippet UI generation	12871	13.41%	

TABLE 5.1: The summary of ENsEN's response time per step (log of 3 queries and 5 results per query)

We notice also that the preprocessing phase takes almost 40% of the total response time. Thus, applying all the preprocessing operation off-line (i.e. indexing phase), can also economize this 40%.

The off-line phase will consist of crawling the web, then for each web page, applying the preprocessing operation, finally, we build the semantic index presented in Section 5.2.2. The off-line phase can significantly enhance the system's performance, we estimate the resulted economy at almost 80% of the actual ENsEN's response time.

### 5.2.5.2 History and Extract

In the ENsEN interactive design (Section 3.4), we showed how our system implements almost all the guidelines of the "information seeking mantra" [Shneiderman, 1996], except the *history* and the *extract*.

The history is the functionality of keeping a trace of the user's actions to support the *undo*, the *replay* and the *progressive refinement*. Integrating the user's profile and the history functionality (presented in Section 5.2.4), will enhance the global user satisfaction.

The extract functionality allows the extraction of results, documents or RDF graphs. This exported information can then be used by other systems. Following the *information seeking mantra*, this functionality will be very useful to the user.

## Appendix A

# Crowdsourcing Microtask Example for the Evaluation of ENsEN's search interface.

## A.1 Introduction

In this appendix, we give an example of the microtasks used in the evaluation of our system *ENsEN* (Chapter 4). We also indicate the topics and the questions selected from TREC 2004 QA track in order to generate the search tasks for the crowdsourcing microtasks.

## A.2 Microtask

### Evaluation Of Search Engines' User Interfaces

#### Instructions

##### Overview

*Welcome to our search user interface study.*

Searching is one of the most prominent action on the web. Search engines are very popular (e.g., Google handles more than three billion queries each day). A search user interface should allow the users to express their information need through a query, and to understand the results of their search. Today's typical search interfaces are vertical ranked lists of results.

The design of search user interfaces has developed dramatically over the years, therefore we propose to evaluate a new kind of interface (ENsEN) and to compare it to a typical one (Google).

You will have to provide answers to search tasks (e.g., "What division (weight) did the boxer Floyd Patterson win?"). For half of them you will be using the Google user interface, while for the other half you will be using the ENsEN user interface. Finally, you will be asked to evaluate and compare the two user interfaces.

We first introduce the main characteristics of the two user interfaces. Then, we will give you guidelines about how to fulfill this job.

##### Google

On the following picture, we can see that the Google's user interface consists of a ranked list of web pages' titles that are followed by an excerpt from the web page. This excerpt is also called a snippet.

The components of the Google's search user interface:

1- The entry form (also called the search box): where you enter your query as a list of keywords.

2- The result's title that links to the corresponding web page.

3- The result's URL.

4- An excerpt from the document.

5- A list of links to subparts of the web page.

PS: Google uses a bold face for each occurrence of a query's keyword within the excerpt.

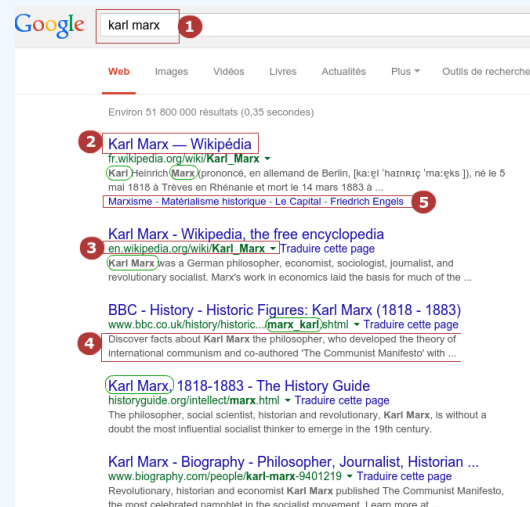


FIGURE A.1: Google interface

## ENsEN

ENsEN is a software system that enhances a traditional search user interface with more meaningful data (see the following picture). To do this, ENsEN identifies the best concepts in each result and uses them to offer a more informative view of the web page.

### The components of ENsEN's search user interface:

- 1- The keywords composing the query: users can remove each keyword using the x button.
- 2- The entry form: where users enter their query as a list of keywords, or reformulate an existing query by adding or removing keywords.
- 3- The related concepts: a ranked list of the best concepts over all the results.
- 4- Each concept of this list can be clicked to see a small description along with selected excerpts from documents within which it was found.
- 5- The result's title, also a link to the corresponding web page.
- 6- A sentence from the web page selected for its capacity to explain the relationship between the query and this result.
- 7- A ranked list of the best concepts found within the web page.
- 8- Each concept of this list can be clicked to see a small description along with selected excerpts from the web page.
- 9- The excerpt proposed by Google.

### ENsEN: The concept description interface:

1. The name of the concept.
2. A definition of the concept.
3. A few selected sentences within which this concept appeared.
4. An icon that can be clicked to see the sentence in the context of its web page.
5. A few facts about the concept.

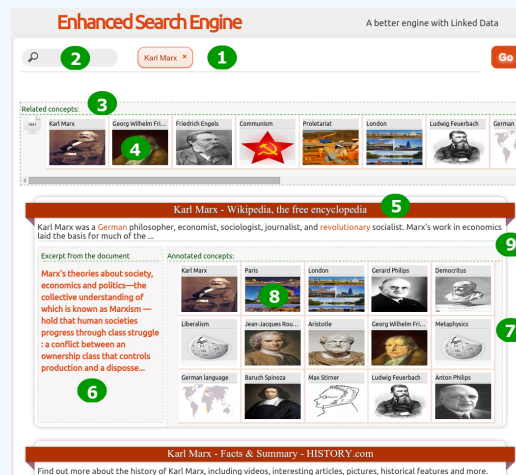


FIGURE A.2: ENsEN interface

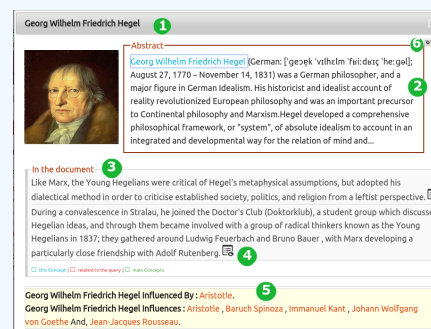


FIGURE A.3: ENsEN concept description interface

## Guidelines

### How to do this job?

A- Entry questionnaire is a background information survey used to evaluate your knowledge of search user interfaces.

B- Search tasks is the main part the job. You are asked to answer a few search tasks. For half of them you will be using the Google user interface, while for the other half you will be using the ENsEN user interface. You will also be asked to evaluate the user interface you used to answer the task.

C- Overall Preference: Once you answered all the tasks, we will ask you for an overall comparison of the two search user interfaces.

D- About you: a short demographic survey.

E- Feedback: at the end we give you the possibility to express your feedback, your impressions and any propositions.

---

### Entry questionnaire (background information)

In this part, we would like you to answer a few questions to evaluate your degree of expertise with the Google web search engine.

#### What must you enter into the Google's search box in order to find an exact phrase?

- Simply enter the exact phrase into Google's search box.
- Use \* on either side of the phrase.
- Put the phrase in quotes.
- Start the phrase with a slash mark.

#### What is the quickest way to find the definition of computer using Google?

- Use your web browser to navigate to [www.dictionnaire.com](http://www.dictionnaire.com).
- Enter into Google's search box: 'define:computer'
- Enter into Google's search box: 'What is a computer?'
- Enter into Google's search box: 'computer definition'

#### What do you type into the Google search box if you only want to search the contents of a specific website?

- There is no easy way to do that.
- Enter into Google's search box: 'admission site:www.stanford.edu'
- Go directly to Stanford's web site and use their search box.
- Enter into Google's search box: 'admission inurl:Stanford'

#### You want to search about 'virus', but you don't want results related to 'computer virus'. What must you enter into the Google's search box?

- There is no easy way to do that.
- Enter into Google's search box: 'virus, not computer'
- Enter into Google's search box: 'virus -computer'
- Enter into Google's search box: 'virus, 0 computer'
- Enter into Google's search box: 'only the word virus'

**How do you evaluate your knowledge of search engines?**

Bad 1 - 2 - 3 - 4 - 5 Very Good

A self-rating question about your ability to find information on the web using a search engine.

---

**Search tasks****The topic is "Crip gang"**

In this part, you are asked to complete a few search tasks.

**Task 1: find the correct answer.****What is their gang color?**

Please, use the ENsEN search user interface. You have up to three minutes.

Please select your answer.

- Yellow
- Red
- Blue
- Black

**Task 2: find a list of answers.****Which cities have Crip gangs?**

Please, use the ENsEN search user interface. You have up to three minutes.

Please, list of answers. (Comma separated list of answer's elements. Please no URL).

**Task 3: find the correct answer with possible query reformulation**

Please, start with the keyword ethnic Crip gangs with the ENsEN user interface, in order to answer the question What ethnic group/race are Crip members?. If you don't find the answer, you can reformulate the query. Please, provide us with the entire sequence of queries you used to find the answer.

Your first query must be: ethnic Crip gangs. You have three minutes.

**The answer is:**

- Gang
- African-American
- Young Gangster
- Violence

**I used the following queries:**

- ethnic Crip gangs
- 

**After task evaluation****How easy was it to find the correct answer?**

Very easy 1 - 2 - 3 - 4 - 5 Very hard

**How do you evaluate the difficulty of this task?**

Very easy 1 - 2 - 3 - 4 - 5 Very hard

**Were you satisfied with the user interface?**

Not at all 1 - 2 - 3 - 4 - 5 Very satisfied



### Overall Preference

**Which search user interface did you prefer?**

Google 1 - 2 - 3 - 4 - 5 ENsEN

**In your opinion, what is the most important component of the ENsEN user interface?**

- The related concepts.
- The result's title.
- The excerpt from the web page.
- The concepts' descriptions.
- Something else.

**What is your overall evaluation of Google's user interface?**

Bad 1 - 2 - 3 - 4 - 5 Very Good

**What is your overall evaluation of ENsEN's user interface?**

Bad 1 - 2 - 3 - 4 - 5 Very Good

**Would you use ENsEN in the future for similar search tasks?**

- Yes
- no

---

### About you

in this part, you are asked to answer some demographic questions about you.

**Your age?**

- older than 60
- older than 40
- older than 30
- older than 20
- younger than 20

**In general, your queries are:**

- Informational queries: Queries that cover a broad topic (e.g., Colorado or trucks) for which there may be thousands of relevant results.
- Navigational queries: Queries that seek a single website or web page of a single entity (e.g., Youtube or delta air lines).
- Transactional queries: Queries that reflect the intent of the user to perform a particular action, like purchasing a car or downloading a screen saver.
- Exploratory queries: Queries about unfamiliar domain, need to learn more about the topic.

---

### Feedback

**In your opinion, what and how can we enhance in the ENsEN's interface?**

Please, give us any additional feedback about the user interface of ENsEN:

### A.3 Used topics and questions

In the following, we present the 8 selected topics from TREC 2004 QA track, with three questions for each topic:

#### **Topic 1: Crip gang**

- What is their gang color?
- Which cities have Crip gangs?
- What ethnic group/race are Crip members?

#### **Topic 2: Fred Durst**

- What is the name of Durst's group?
- What are titles of the group's releases?
- What record company is he with?

#### **Topic 3: Hale Bopp comet**

- How often does it approach the earth?
- In what countries was the comet visible on its last return?
- When was the comet discovered?

#### **Topic 4: James Dean**

- When did James Dean die?
- What movies did he appear in?
- Which was the first movie that he was in?

#### **Topic 5: Rhodes scholars**

- How long does one study as a Rhodes scholar?
- Name famous people who have been Rhodes scholars.
- Where do Rhodes scholars study?

#### **Topic 6: Black Panthers**

- Who founded the Black Panthers organization?
- Who have been members of the organization?
- Where was it founded?

#### **Topic 7: Insane Clown Posse**

- What is their style of music?
- Who are the members of this group?
- What is their biggest hit?

**Topic 8: Prions**

- What are prions made of?
- What diseases are prions associated with?
- Who discovered prions?



# Bibliography

- Ageev, M., Lagun, D., and Agichtein, E. (2013). Improving search result summaries by using searcher behavior data. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 13–22. ACM.
- Alonso, O., Marshall, C., and Najork, M. (2014). Crowdsourcing a subjective labeling task: A human-centered framework to ensure reliable results. Technical report, Microsoft Research.
- Amato, F., Gargiulo, F., Mazzeo, A., Moscato, V., and Picariello, A. (2013). An rdf-based semantic index. In *Natural Language Processing and Information Systems*, pages 315–320. Springer.
- Andersson, C. A. and Bro, R. (2000). The n-way toolbox for matlab. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4.
- Anyanwu, K., Maduko, A., and Sheth, A. (2005). Semrank: ranking complex relationship search results on the semantic web. In *Proceedings of the 14th international conference on World Wide Web*, pages 117–127. ACM.
- Bai, X., Delbru, R., and Tummarello, G. (2008). Rdf snippets for semantic web search engines. In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 1304–1318. Springer.
- Baziz, M., Boughanem, M., and Aussenac-Gilles, N. (2005a). Conceptual indexing based on document content representation. In *Context: nature, impact, and role*, pages 171–186. Springer.
- Baziz, M., Boughanem, M., and Traboulsi, S. (2005b). A concept-based approach for indexing documents in ir. In *INFORSID*, volume 2005, pages 489–504.
- Berners-Lee, T. (2006). Linked data-design issues (2006). <http://www.w3.org/DesignIssues/LinkedData.html>. [Online; accessed 2016-03-01].
- Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

- Berry, M. W. (1992). Large-scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49.
- Bizer, C., Auer, S., Kobilarov, G., Lehmann, J., and Cyganiak, R. (2007). Dbpedi-aquering wikipedia like a database. In *Developers track presentation at the 16th international conference on World Wide Web, WWW16*, pages 8–12.
- Bizer, C., Eckert, K., Meusel, R., Mühleisen, H., Schuhmacher, M., and Völker, J. (2013). Deployment of rdfa, microdata, and microformats on the web—a quantitative analysis. In *The Semantic Web—ISWC 2013*, pages 17–32. Springer.
- Blanco, R., Cambazoglu, B. B., Mika, P., and Torzec, N. (2013). Entity recommendations in web search. In *The Semantic Web—ISWC 2013*, pages 33–48. Springer.
- Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document.
- Butt, A. S., Haller, A., and Xie, L. (2015). A taxonomy of semantic web data retrieval techniques. In *Proceedings of the 8th International Conference on Knowledge Capture*, page 9. ACM.
- Callan, J., Allan, J., Clarke, C. L. A., Dumais, S., Evans, D. A., Sanderson, M., and Zhai, C. (2007). Meeting of the minds: An information retrieval research agenda. *SIGIR Forum*, 41(2):25–34.
- Campinas, S., Delbru, R., and Tummarello, G. (2012). Effective retrieval model for entity with multi-valued attributes: Bm25mf and beyond. In *Knowledge Engineering and Knowledge Management*, volume 7603 of *Lecture Notes in Computer Science*, pages 200–215. Springer Berlin Heidelberg.
- Carvalho, A. and Larson, K. (2013). A consensual linear opinion pool. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2518–2524. AAAI Press.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357.
- Daiber, J., Jakob, M., Hokamp, C., and Mendes, P. N. (2013). Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.
- Dali, L., Fortuna, B., Duc, T. T., and Mladenić, D. (2012). Query-independent learning to rank for rdf entity search. In *The Semantic Web: Research and Applications*, pages 484–498. Springer.

- Damljanovic, D., Stankovic, M., and Laublet, P. (2012). Linked data-based concept recommendation: Comparison of different methods in open innovation scenario. *The Semantic Web: Research and Applications*, pages 24–38.
- Delbru, R., Toupikov, N., Catasta, M., Tummarello, G., and Decker, S. (2010). Hierarchical link analysis for ranking web data. In *The Semantic Web: Research and Applications*, pages 225–239. Springer.
- Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V., and Sachs, J. (2004). Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04*, pages 652–659, New York, NY, USA. ACM.
- Dumais, S., Cutrell, E., and Chen, H. (2001). Optimizing search by showing results in context. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '01*, pages 277–284, New York, NY, USA. ACM.
- Fafalios, P. and Tzitzikas, Y. (2014). Post-analysis of keyword-based search results using entity mining, linked data, and link analysis at query time. In *Semantic Computing (ICSC), 2014 IEEE International Conference on*, pages 36–43. IEEE.
- Fellbaum, C. (2006). Wordnet(s). In Brown, K., editor, *Encyclopedia of Language & Linguistics (Second Edition)*, pages 665 – 670. Elsevier, Oxford, second edition edition.
- Franz, T., Schultz, A., Sizov, S., and Staab, S. (2009). Triplerank: Ranking semantic web data by tensor decomposition. In *The Semantic Web-ISWC 2009*, pages 213–228. Springer.
- Gauch, S., Chaffee, J., and Pretschner, A. (2003). Ontology-based personalized search and browsing. *Web Intelligence and Agent Systems*, 1(3-4):219–234.
- Ge, W., Cheng, G., Li, H., and Qu, Y. (2012). Incorporating compactness to generate term-association view snippets for ontology search. *Information Processing & Management*, pages 513–528.
- Haas, K., Mika, P., Tarjan, P., and Blanco, R. (2011). Enhanced results for web search. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, page 725.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.
- Hearst, M. (2011). *User interfaces for search*. Citeseer. pages 21–55.

- Heath, T. and Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136.
- Hildebrand, M., Ossenbruggen, J. V., and Hardman, L. (2007). An Analysis of Search-based User Interaction on the Semantic Web. *Information Systems*, pages 1386–3681.
- Hogan, A., Harth, A., and Decker, S. (2006). Reconrank: A scalable ranking method for semantic web data with context.
- Hu, H. and Du, X. (2012). Combining n-gram retrieval with weights propagation on massive rdf graphs. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 1181–1185.
- Järvelin, K. and Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM.
- Jeong, J.-W., Morris, M. R., Teevan, J., and Liebling, D. J. (2013). A crowd-powered socially embedded search engine. In *ICWSM*.
- Jiang, J., He, D., and Allan, J. (2014). Searching, browsing, and clicking in a search session: Changes in user behavior by task and over time. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '14*, pages 607–616, New York, NY, USA. ACM.
- Jindal, V., Bawa, S., and Batra, S. (2014). A review of ranking approaches for semantic search on web. *Information Processing & Management*, 50(2):416–425.
- Kelly, D. (2009). Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends in Information Retrieval*, 3(12):1–224.
- Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.
- Kohlschütter, C., Fankhauser, P., and Nejdl, W. (2010). Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450. ACM.
- Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.
- Koumenides, C. L. and Shadbolt, N. R. (2014). Ranking methods for entity-oriented semantic web search. *Journal of the Association for Information Science and Technology*, 65(6):1091–1106.



- Krippendorff, K. (2012). *Content analysis: An introduction to its methodology*. Sage Publications.
- KrishnaVeni, C. and Sobha Rani, T. (2011). On the classification of imbalanced datasets. *IJCST*, 2:145–148.
- Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.
- Langville, A. N. and Meyer, C. D. (2011). *Google’s PageRank and beyond: The science of search engine rankings*. Princeton University Press.
- Lehmann, J., Gerber, D., Morsey, M., and Ngomo, A.-C. N. (2012). Defacto-deep fact validation. In *The Semantic Web–ISWC 2012*, pages 312–327. Springer.
- Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., et al. (2014). Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 5:1–29.
- Lempel, R. and Moran, S. (2001). Salsa: the stochastic approach for link-structure analysis. *ACM Transactions on Information Systems (TOIS)*, 19(2):131–160.
- Marcos, M.-C. and González-Caro, C. (2010). Comportamiento de los usuarios en la página de resultados de los buscadores. un estudio basado en eye tracking. *El profesional de la información*, 19(4):348–358.
- Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems, I-Semantics ’11*, pages 1–8, New York, NY, USA. ACM.
- Metzler, D. and Kanungo, T. (2008). Machine learned sentence selection strategies for query-biased summarization. In *SIGIR Learning to Rank Workshop*, pages 40–47.
- Mihalcea, R. and Moldovan, D. (2000). Semantic indexing using wordnet senses. In *Proceedings of the ACL-2000 workshop on Recent advances in natural language processing and information retrieval: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 11*, pages 35–45. Association for Computational Linguistics.
- Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.
- Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45.

- Neumayer, R., Balog, K., and Nørkvåg, K. (2012). On the modeling of entities for ad-hoc entity search in the web of data. In *Proceedings of the 34th European Conference on Advances in Information Retrieval, ECIR'12*, pages 133–145, Berlin, Heidelberg. Springer-Verlag.
- Nie, Z., Zhang, Y., Wen, J.-R., and Ma, W.-Y. (2005). Object-level ranking: bringing order to web objects. In *Proceedings of the 14th international conference on World Wide Web*, pages 567–574. ACM.
- Nielsen, J. (2004). When search engines become answer engines. *Jakob Nielsens Alertbox*, pages 1–5.
- Oram, P. (2001). Wordnet: An electronic lexical database. christiane fellbaum (ed.). cambridge, ma: Mit press, 1998. pp. 423.
- Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: bringing order to the web. Technical report, Stanford InfoLab.
- Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.
- Passant, A. (2010). Measuring semantic distance on linking data and using it for resources recommendations.
- Penin, T., Wang, H., Tran, T., and Yu, Y. (2008). Snippet generation for semantic web search engines. In *The Semantic Web*, pages 493–507. Springer.
- Pound, J., Mika, P., and Zaragoza, H. (2010). Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, pages 771–780, New York, NY, USA. ACM.
- Quinlan, J. R. (1993). *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.
- Roa-Valverde, A. J. and Sicilia, M.-A. (2014). A survey of approaches for ranking on the web of data. *Information Retrieval*, 17(4):295–325.
- Rodríguez, M. A. and Egenhofer, M. J. (2003). Determining semantic similarity among entity classes from different ontologies. *Knowledge and Data Engineering, IEEE Transactions on*, 15(2):442–456.
- Sah, M. and Wade, V. (2013). Personalized concept-based search and exploration on the web of data using results categorization. In *The Semantic Web: Semantics and Big Data*, pages 532–547. Springer.

- Schmachtenberg, M., Bizer, C., and Paulheim, H. (2014). Adoption of the linked data best practices in different topical domains. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pages 245–260.
- Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55.
- Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343.
- Steiner, T., Troncy, R., and Hausenblas, M. (2010). How google is using linked data today and vision for tomorrow. *Proceedings of Linked Data in the Future Internet*, 700.
- Tanon, A., Catasta, M., Prokofyev, R., Demartini, G., Aberer, K., and Cudré-Mauroux, P. (2016). Contextualized ranking of entity types based on knowledge graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*.
- Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.
- Varadarajan, R. and Hristidis, V. (2005). Structure-based query-specific document summarization. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM '05*, pages 231–232, New York, NY, USA. ACM.
- Wang, C., Jing, F., Zhang, L., and Zhang, H.-J. (2007). Learning query-biased web page summarization. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 555–562. ACM.
- Wei, W., Barnaghi, P., and Bargiela, A. (2011). Rational research model for ranking semantic entities. *Information Sciences*, 181(13):2823–2840.
- White, R. W., Ruthven, I., and Jose, J. M. (2002). Finding relevant documents using top ranking sentences: An evaluation of two alternative schemes. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02*, pages 57–64, New York, NY, USA. ACM.
- Woods, W. A. (1997). Conceptual indexing: A better way to organize knowledge. Technical report, Sun Microsystems, Inc., Mountain View, CA, USA.
- Yumusak, S., Dogdu, E., and Kodaz, H. (2014). A short survey of linked data ranking. In *Proceedings of the 2014 ACM Southeast Regional Conference, ACM SE '14*, pages 48:1–48:4, New York, NY, USA. ACM.

- Zablith, F., Fernandez, M., and Rowe, M. (2011). The ou linked open data: Production and consumption. elearning approaches for the linked data age. In *Extended Semantic Web Conference*.
- Zadeh, P. D. H. and Reformat, M. Z. (2012). Fuzzy semantic similarity in linked data using the owa operator. In *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, pages 1–6. IEEE.
- Zaragoza, H., Rode, H., Mika, P., Atserias, J., Ciaramita, M., and Attardi, G. (2007). Ranking very many typed entities on wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 1015–1018. ACM.

# Abbreviations

**AOR** Ad-hoc Object Retrieval

**CCS** Compressed Column Storage

**DCG** Discounted Cumulative Gain

**EL** Entity Linking

**ENsEN** Enhanced Search Engine

**GBRT** Gradient Boosting Regression Tree

**GRS** Google Rich Snippet

**HTML** HyperText Markup Language

**HTTP** Hypertext Transfer Protocol

**IIR** Interactive Information Retrieval

**IR** Information Retrieval

**LD** Linked Data

**LDRANK** Linked Data Ranking Algorithm

**LDSVD** Linked Data Singular Value Decomposition

**LOD** Linking Open Data

**NDCG** Normalized Discounted Cumulative Gain

**NED** Named Entity Disambiguation

**NER** Named Entity Recognition

- 
- NLP** Natural Language Processing
- RBF** Radial Basis Function Networks
- RDF** Resource Description Framework
- SE** Search Engines
- SERP** Search Engine Results Page
- SMOTE** Synthetic Minority Over-Sampling Technique
- SPARQL** SPARQL Query Language for RDF
- SUI** Search User Interface
- SVD** Singular Value Decomposition
- SVMs** Support Vector Machines
- SW** Semantic Web
- SWD** Semantic Web Document
- SWDB** Semantic Web DataBase
- SWO** Semantic Web Ontology
- UDI** Universal Document Identifier
- URL** Uniform Resource Locator
- W3C** World Wide Web Consortium
- WIR** Web Information Retrieval
- WWW** World Wide Web

# Publications List

## **PUBLICATIONS DIRECTLY RELATED TO THE THESIS (in chronological order)**

1. M. Alsarem : A Generic Approach Based on Linked Data to Enhance Web Information Retrieval and Increase User Satisfaction. In : Conférence en Recherche d'Information et Applications, CORIA13, RJCRI13. pp. 299-304, 2013.
2. M. Alsarem, P. E. Portier, S. Calabretto, H. Kosch : Making Use of Linked Data for Generating Enhanced Snippets. In: Proceedings of the 11th Extended Semantic Web Conference, ESWC14, The Semantic Web: ESWC 2014 Satellite Events (pp. 275-279). Springer International Publishing, 2014, (Demo paper).
3. M. Alsarem, P. E. Portier, S. Calabretto, H. Kosch : Ordonnancement d'entités appliqué à la construction de snippets sémantiques. In: Conférence en Recherche d'Information et Applications, CORIA15, 2015.
4. M. Alsarem, P. E. Portier, S. Calabretto, H. Kosch : Ranking Entities in the Age of Two Webs, an Application to Semantic Snippets. In: Proceedings of the 12th Extended Semantic Web Conference, ESWC15, The Semantic Web. Latest Advances and New Domains. Springer International Publishing. p. 541-555, 2015.
5. M. Alsarem, P. E. Portier, S. Calabretto, H. Kosch : Ordonnancement d'entités pour la rencontre du web des documents et du web des données. In : Document Numérique VOL 18/2-3, pp.123-154, 2015.
6. M. Alsarem, P. E. Portier, S. Calabretto, H. Kosch : SEMashup: Making Use of Linked Data for Generating Enhanced Snippets. In: Proceedings of the AI Mashup Challenge 2014, co-located with 11th Extended Semantic Web Conference (ESWC 2014), (short paper).

## **PUBLICATIONS UNDER REVIEW**

- 
1. M. Alsarem, P. E. Portier, S. Calabretto, H. Kosch : Query-biased Ranking of LOD Entities for Semantic Snippets. Submitted to: Information Processing and Management journal (under review).