Emma Sofie Søvik
Nils Olav Tuv

# Information Retrieval in Semantic Data

Semantically Enriched Ranking Approaches

Master's thesis in Datateknologi
Supervisor: Trond Aalberg
June 2023

Emma Sofie Søvik
Nils Olav Tuv

# Information Retrieval in Semantic Data

Semantically Enriched Ranking Approaches

Master’s thesis in Datateknologi
Supervisor: Trond Aalberg
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

# Abstract

The amount of semantic data available has grown rapidly over the years. Semantic data offers numerous advantages in information retrieval, however, traditional information retrieval technologies have shortcomings when utilizing the information in this structured data. Therefore, new techniques that leverage the semantics of the data are needed. To address this, previous research has introduced new techniques known as semantically enriched rankings. These techniques can be combined with traditional information retrieval systems and are meant to improve the ranking further. This thesis aimed to implement and investigate some of these enrichments, to see if they could help information retrieval systems perform better for semantic data.

The thesis explores four different enrichment techniques, including target entity, Tonon's hybrid rank, entity linking, and centrality. These approaches were then implemented and modified together with a test search system. Multiple queries with different information needs were used to assess each enrichment, and the result from the enrichment was compared to those of the initial ranking with only traditional methods.

The results gathered from different evaluation metrics and observation suggests that all four of the implemented enrichments have positive features. Because they all leverage the semantics in different ways, each enrichment worked differently for different types of queries. It was therefore concluded that a combination of the enrichment might be the best option to create an optimal search system for semantic data. These findings are a step forward in the process of finding methods for information retrieval systems to be able to exploit semantic data.

# Sammendrag

Mengden semantisk data tilgjengelig har økt betraktelig de siste årene. Semantisk data tilbyr mange fordeler i informasjonsgjenfinning, men tradisjonelle informasjonsgenfinningsmetoder greier ikke utnytte informasjonen i denne strukturerte dataen. Derfor er det behov for nye teknikker som utnytter semantikken i slik data. Tidligere forskning har introdusert nye teknikker kalt semantically enriched ranking. Disse teknikkene kan bli kombinert med tradisjonelle informasjonsgjenfinningsmetoder og er ment å forbedre rangeringen. Denne oppgaven implementerer og undersøker noen av disse teknikkene for å se om de kan hjelpe informasjonsgjenfinnings systemer til å prestere bedre for semantisk data.

Denne oppgaven tar for seg fire forskjellige teknikker for å forbedre tradisjonell informasjonsgjenfinning. Dette inkluderer target entity, Tonon's hybrid ranking, entity linking, og sentralitet. Disse teknikkene ble deretter implementert sammen med et søkesystem. Flere ulike spørringer med forskjellige informasjonsbehov ble brukt for å evaluere de ulike teknikkene, før resultatet ble analysert og sammenlignet opp mot resultatene fra tradisjonell informasjonsgjenfinning.

Resultatene fra forskjellige evalueringsmetoder, samt observasjon, tilsier at alle fire teknikker har positive egenskaper. Siden teknikkene bruker semantikken i dataen på ulike måter, gir teknikkene forskjellige resultater for de ulike spørringene. Derfor ble det konkludert at en kombinasjon av teknikkene er å foretrekke for å kunne optimalisere søkesystemer for semantisk data. Disse funnene er et skritt i riktig retning i arbeidet med å finne metoder som gjør at semantiske informasjonsgjenfinningssystemer kan utnytte informasjonen i semantisk data.

# Preface

This thesis was written by Emma Sofie Søvik and Nils Olav Tuv for the Department of Computer Science at the Norwegian University of Science and Technology (NTNU). This thesis marks the end of our master's degree in Computer Science.

We would like to thank all the participants of the survey. With your help, we were able to gather relevance labels for our data, which helped us immensely when evaluating the various results. Thank you for taking the time to partake in the survey. We would also like to thank our supervisor, Trond Aalberg, for his support during the last two semesters.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Topic and Motivation

The Semantic Web has previously had many names: web 3.0, web of data, web of linked data, etc. The goal of the Semantic Web is to enrich data with meaningful information, thereby enabling computers to comprehend and utilize the data effectively. This is done by linking data sources together, and building knowledge graphs of data. The word *semantic* implies meaning or understanding, and the Semantic Web originates from Tim Berners Lee's vision in the 90s of creating an internet with data understandable for machines. Semantic data has since then had constant growth and is now utilized in various industries, including finance, bibliographic, science, healthcare, and more.

With this information rich in semantics, new technology that is able to exploit it is necessary. An area this is important is information retrieval (IR). SPARQL is the query language of semantic data, however, similar to other query languages only people with experience with the language and the dataset is able to use it to retrieve desired information. For this reason, IR systems are needed, to make it accessible to a wider range of users. Every day, hundreds of millions of people engage themselves with IR, and it has grown to be an important technology. Whether the goal is to find a specific movie on Netflix or find information on a subject from Google, IR technology is needed to understand the user's information need and retrieve the relevant documents, without requiring expertise from the user.

Traditional IR methods are developed for documents and involve comparing terms from queries with terms from documents. Semantic Web differs from the classic documents by concerning the meaning of the data and that it contains information other than just a cluster of terms. Because of this, classic IR technology is not fully appropriate for Semantic Web data, and new technology is needed to exploit this knowledge. Appropriate IR technology for the Semantic Web can not only improve the search and retrieval of Semantic Web data, but can arguably revolutionize IR technology in general and motivate more people to create and use semantic data. This is where the motivation for our thesis arose, and we want to research new IR technology for the Semantic Web.

Information retrieval in the Semantic Web has been a topic for researchers since Tim Bernes-Lee's vision in the 90s. Previous research has been addressing the challenges this technology faces and has tried different techniques to be able to exploit the knowledge given by the data. One way to do this is to separate the data into predefined entities and perform an entity search. This technology enables extensions of the traditional term-frequency retrieval technology, by adding enrichments like target entities, entity linking, and centrality, among others. These enrichments leverage the structure of the entities, creating a semantically enriched ranking.

In our project, we aim to investigate semantically enriched ranking, by looking at extensions and enrichments that can complement traditional information retrieval. The aim is to enhance the ranking for a better search for semantic data. The project aims to investigate different IR techniques and evaluate their effects on semantic information retrieval, as well as experiment with how these work together in an integrated solution.

## 1.2 Goal and Research Questions

**Research Goal:** The goal of this project is to investigate how new information retrieval technology can improve the search and ranking results of linked Semantic Web data.

Traditional IR approaches have limitations when applied to semantic data due to the data's richness and complexity. Therefore, the project aims to find and evaluate new approaches and enrichment methods to improve information retrieval for semantic data.

To achieve this goal, the following four research questions have been formulated:

**R.Q. 1:** How can existing IR technologies be modified to effectively leverage the linked Semantic Web data?

**R.Q. 2:** In what way can the various enrichment techniques improve the relevance of search results compared to traditional methods?

**R.Q. 3:** What is the impact of different enrichment techniques on the performance of semantic data retrieval?

**R.Q. 4:** Which enrichment techniques are the most promising in improving the retrieval of semantic data?

Answering these research questions will help us identify the most effective techniques for information retrieval on semantic data and contribute to the advancement of IR technology in general.

## 1.3 Contributions

The main contribution of this thesis is the investigation of different techniques and approaches that can be implemented to improve the search and ranking of semantic data.

To achieve this, we will develop a search engine that incorporates various methods and techniques for information retrieval on semantic data. We will conduct extensive research to identify the most effective approaches for leveraging linked semantic data. Then, we will integrate these approaches into our test search engine and evaluate their performance using various metrics.

The contributions of this thesis are:

- A comprehensive review of existing literature on IR for semantic data, which provides insights into the challenges and opportunities of this field.
- A detailed description of different techniques and enrichments that can be used to improve the search and ranking of semantic data.
- An implementation of a search engine that incorporates these techniques and enrichments, which serves as a tool to visualize and evaluate the proposed methods.
- An evaluation of the performance of the search engine using various metrics, which provides insights into the strengths and weaknesses of the proposed methods and their potential for practical applications.

## 1.4 Thesis Structure

The remainder of this thesis is organized as follows:

**Chapter 2:** Background Theory - Introduces the fundamental theory for this project, including information on The Semantic Web, entities, and information retrieval.

**Chapter 3:** Information Retrieval in Semantic Data - Provides a detailed overview of relevant previous work done on information retrieval in semantic data.

**Chapter 4:** Search Engine Development - Provides a description of the tools and implementation of the test search engine.

**Chapter 5:** Methodology - Presents various enrichment approaches for IR in semantic data and describes the implementation of these.

**Chapter 6:** Evaluation Strategy - Describes the evaluation strategy of the thesis.

**Chapter 7:** Results and Discussion - Presents the results from the various enrichment techniques, as well as a critical evaluation and analysis of these.

**Chapter 8:** Conclusion - Concludes the thesis. Describes how each research question have been answered, as well as new proposals for further research.

## Note

This thesis is a continuation of the pre-project that was initiated this fall. Because of this, some of the text in this document is revised versions of the text in the pre-project report and might appear similar or even identical to this thesis. This is the case for some parts in chapter 1, chapter 2, chapter 3, and chapter 4.

# 2 Background Theory

To fully understand the rest of this thesis, it is important to have some basic knowledge on the two main topics of the thesis: Semantic web and information retrieval. This chapter is therefore included to introduce these two topics and present the relevant background theory of the project.

The chapter starts with an introduction to the Semantic Web, followed by a description of entities and entity search. Next, comes a description on information retrieval and various related topics.

## 2.1 The Semantic Web

The Semantic Web was originally seen as an extension of the web, by adding machine-readable information to the internet. The idea originated from Tim Berners-Lee, the inventor of the World Wide Web, who had a grand vision for the Semantic Web, or Web 3.0, and how it would make the web meaningful to computers. Although the vision of Web 3.0 to many may seem forgotten and outdated, the Semantic Web still has widespread use. Today, the Semantic Web is an area with a large amount of information-rich data called semantic data. The web contains many knowledge bases with this data, which covers various different domains. For instance, Wikidata[1], DBPedia[2], YAGO[3], and British National Bibliography[4].

Semantic data is built up by triples, more specifically the data consist of resources that are linked or connected together creating a graph or web of linked data. This is created with technologies such as RDF, OWL, and SPARQL. Where RDF is the data modeling framework for the Semantic Web. OWL is a web ontology language, which is designed to give more meaning to semantic resources and their relationships. SPARQL is the query language of the data and provides a way to extract and manipulate it.

RDF, Resource Description Framework, is used to describe the resources and relationships of the Semantic Web. As mentioned, this is with graphs composed of triples. The triples in RDF consist of three components, the subject, the predicate, and the object, which is visualized in figure 1. The subject is the resource being described by the triple. The object is also a resource or a literal value and is related to the subject through the predicate, which describes the relationship between the subject and the object. These three components are then used to state facts about resources. For example, *J.K Rowling wrote the children's fantasy series Harry Potter* could be expressed through the simple triples that are shown in table 1.

Table 1: Harry Potter triples

| Subject | Predicate | Object |
|---|---|---|
| J.K Rowling | wrote | Harry Potter |
| Harry Potter | is | series |
| Harry Potter | has category | children's fantasy |

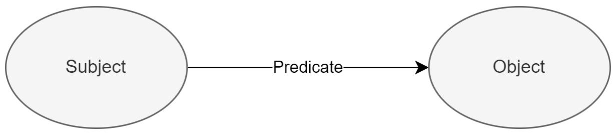


Figure 1: Components of an RDF triple

To identify the resources and relationships, RDF uses Uniform Resource Identifiers. According to Cure (2015), URI is a standardized format for identifying a resource. Because of this, the components in the simple Harry Potter example would be exchanged with URIs. For instance,

[1]https://www.wikidata.org/
[2]https://www.dbpedia.org
[3]https://yago-knowledge.org
[4]https://search.bl.uk/

*J.K Rowling* could be exchanged with the URI *http://www.example.com/people/JKRowling.* The well-known URL (uniform resource locator) is a type of URI and can also be used in RDF triples when referring to a web resource.

One big element of semantic data is ontology, as it provides meaning to the data. In the context of the Semantic Web, an ontology formally defines classes, concepts, and relationships to describe and represent an area of knowledge, a domain (Yu 2011). The use of ontologies decreases ambiguities and makes the data more machine-readable. For instance, in the example about *J.K. Rowling* and *Harry Potter*, there could be hundreds of different definitions of authors or fantasy series, however having one single way of defining them would make the Semantic Web and its data more comprehensible as more references could be added with the same definition. There are many different ontology languages, where one of the most used is OWL. OWL, or Web Ontology Language, is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is an extension of the primitive language RDFS. These languages are used to create shared vocabularies or a common language for specific domains.

SPARQL is short for SPARQL, Protocol, and RDF Query Language and is a language to query and manipulate RDF graphs. SPARQL has similarities with the well-known relational query language SQL, as it has the calls SELECT, WHERE, GROUP BY, HAVING, etc from SQL. Another similarity to SQL is that the user must know the language and data to retrieve needed information. According to the study by Dadzie and Rowe (2011), the general public finds using structured query languages difficult. For this reason, another way of retrieving information is needed.

## 2.2 Entities

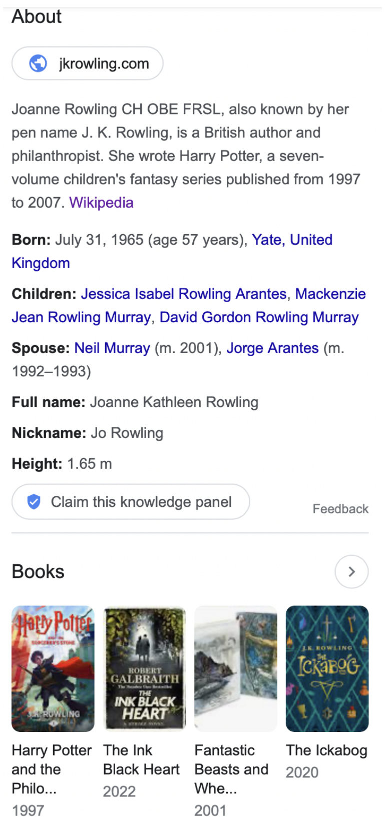
As described in the previous section, the Semantic Web consists of semi-structured resources that are linked together instead of textual documents. Instead of returning a relevant document, a common way to search on the Semantic Web is to return relevant entities along with the relevant entity properties. Balog (2018) describes an entity as an object in the real world that can be referred to, and has created the following definition of an entity: "An entity is a uniquely identifiable object or thing, characterized by its name(s), type(s), attributes, and relationships to other entities.". Entities are used to semantically enrich unstructured text in documents as well as allow computers to understand the meaning of text. The most common entities are called named entities, which are used in this thesis. One way to describe named entities is world-like objects. Examples of this can be books, authors, titles, and publishers.

All information associated with entities is called entity properties. These properties include unique identifiers, names, types, relationships, and attributes. Attributes describe the characteristics of an entity. The attributes of an author could for example include name, last name, birth date, birth place, books etc. An example of an entity is shown in figure 2 with Google's Knowledge Graph Search[5] on *J.K. Rowling.*

In the Semantic Web, entities have an essential role. From the information-seeker perspective, letting the user search for specific entities is more efficient and accurate than normal document retrieval. According to a study performed in 2010, around 40% of queries on the web are searching for specific entities (Pound et al. 2010).

According to Balog (2018), Entity-Oriented Search and Semantic Search are often mentioned in the same context, and there are no clearly defined differences between the two. When retrieving entities, there are several challenges that differentiate it from normal document retrieval.

[5]https://support.google.com/knowledgepanel/answer/9787176?hl=en


About

jkrowling.com

Joanne Rowling CH OBE FRSL, also known by her pen name J. K. Rowling, is a British author and philanthropist. She wrote Harry Potter, a seven-volume children's fantasy series published from 1997 to 2007. Wikipedia

**Born:** July 31, 1965 (age 57 years), Yate, United Kingdom

**Children:** Jessica Isabel Rowling Arantes, Mackenzie Jean Rowling Murray, David Gordon Rowling Murray

**Spouse:** Neil Murray (m. 2001), Jorge Arantes (m. 1992–1993)

**Full name:** Joanne Kathleen Rowling

**Nickname:** Jo Rowling

**Height:** 1.65 m

Claim this knowledge panel

Feedback

Books

Harry Potter and the Philo... 1997

The Ink Black Heart 2022

Fantastic Beasts and Whe... 2001

The Ickabog 2020

Figure 2: J.K. Rowling entity-card from Google

## 2.3 Information Retrieval

Information retrieval (IR) is the process of retrieving relevant information from a collection (Singhal et al. 2001). IR has been a big field of interest since the World Wide Web became public. IR systems help people retrieve desired information without having to learn an advanced query language and can be used to retrieve both structured and unstructured data. Today, most people engage themselves with IR on a regular basis through web search engines. These typically work by having the user insert a query, often containing multiple keywords, into the search engine. The IR system then uses an index to match the documents in the collection with the query, or a parsed and extended version of the query, before returning a ranked list of these relevant documents. A visualization of this is shown in figure 3.

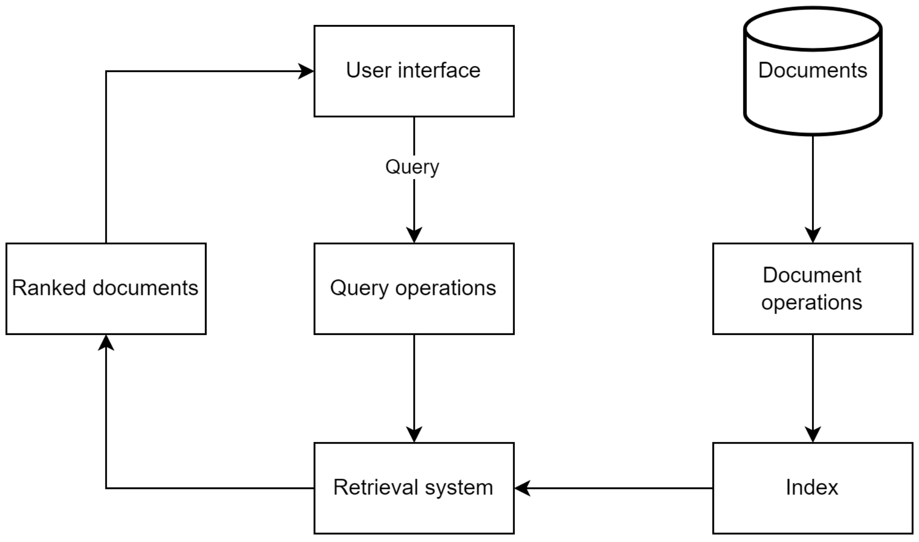


Figure 3: A traditional IR System

Information retrieval in the Semantic Web differs from traditional IR, as it does not consist of unstructured text documents, but semi-structured resources. There are multiple ways to retrieve information on semantic data. It is for example possible to retrieve a ranked list of the resources, preserving the original structure, and getting parts of the graph as result (see Han et al. (2017)).

Another way, and the method that is used in this thesis, is to combine the resources into various entities and return a ranked list of these. Entity retrieval can be performed by gathering the entity terms in a bag-of-word and using the traditional TF-IDF weight to rank the entities (Balog 2018). With this method, an indexing of the entities is needed.

### 2.3.1 Indexing

Indexing is essential for information retrieval systems since it makes the retrieval process efficient. Information retrieval systems often have a large number of documents, which can make the process time-consuming. When using these retrieval systems, time and resources are of the essence. To solve this issue, indexing is commonly used in information retrieval systems. Indexing reduces the documents to an index that can be accessed without searching through the data each time. The reduction of the documents to an index minimizes the resources needed and simplifies the rest of the process (Kaur and Gupta 2016).

Indexing consists of four stages: content specification, tokenization of documents, processing document terms, and index building (Huang and Zhang 2009). In content specification, the set of documents that are being retrieved is specified. Tokenization is the process of separating the words in the document into groups of tokens that are similar. When processing the document terms, punctuation and stop words can be removed. The words can also be stemmed. After this process is complete, the index is built. The purpose of this process is to ensure the documents are fast and easy to retrieve.

Indexing semantic data can be performed in a number of different ways. It is for example possible to index structured RDF data directly. This can be done in two ways: horizontal indexing and vertical indexing (Blanco, Mika et al. 2011). Horizontal indexing represents the resources using three fields, one field for the tokens, one field for the properties, and one field for the tokens of the subject URI. The fields for the tokens and the properties are aligned so the token field and property field corresponds. This means that the property and the token it belongs to have the same index. An example where the author *Joanne Rowling* is indexed horizontally can be seen in table 2. The other option, vertical indexing, uses a new field for each property in the dataset. The position of the data is still useful, for instance in cases where one needs to make sure the first and last name of an entity are matched as consecutive words. The same example is used to visualize vertical indexing and can be seen in table 3.

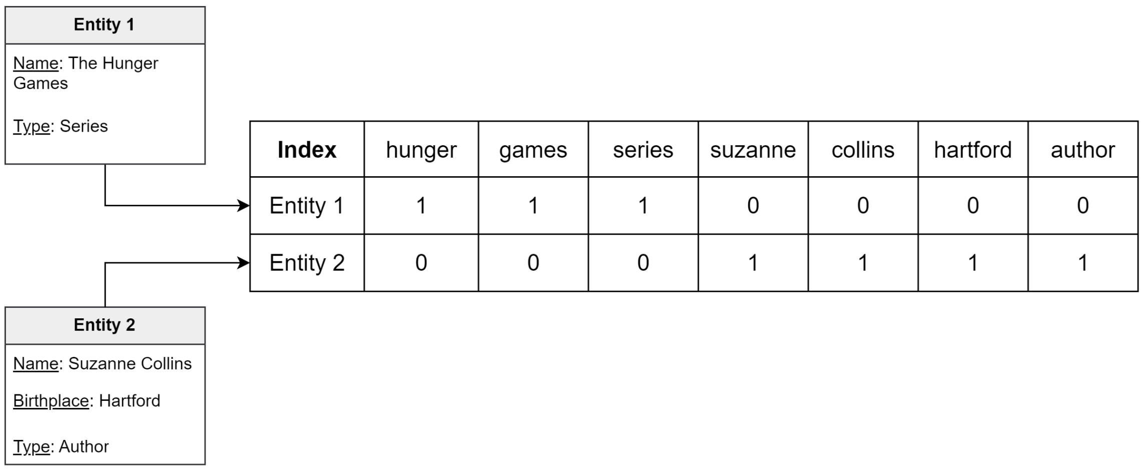
Table 2: Horizontal indexing example

| **Field** | **pos1** | **pos2** | **pos3** | **pos4** | **pos5** |
|---|---|---|---|---|---|
| Token | Joanne | Rowling | Author | 57 | |
| Property | ex:name | ex:name | ex:occupation | ex:age | |
| Subject | http | example | org | ns | joannerowling |

Table 3: Vertical indexing example

| **Field** | **pos1** | **pos2** | **pos3** | **pos4** | **pos5** |
|---|---|---|---|---|---|
| ex:name | Joanne | Rowling | | | |
| ex:occupation | Author | | | | |
| ex:age | 57 | | | | |

Another way to index semantic data, and the one used in this thesis, is to index the entities and all of their properties. One popular method for this is full-text indexing, where entities are compressed into a bag-of-words consisting of the attributes of the entities. The bag-of-words can then further be used to match the entities to search queries, making it an effective tool in IR systems. Figure 4 visualizes how two entities are compressed into an index. With this index, one can see which and how many terms appear in each entity.



| **Index** | hunger | games | series | suzanne | collins | hartford | author |
|---|---|---|---|---|---|---|---|
| Entity 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Entity 2 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Figure 4: Visualization of full-text indexing

### 2.3.2 Ranking

Traditionally, IR systems consist of a collection of unstructured text documents and rank these documents based on matching the query terms with the terms in the documents. The documents with the most matching terms will then typically be ranked the highest. There exist multiple methods of ranking the documents, often divided into ranking models. Varying from simple models like the boolean model, based on boolean algebra, to more advanced models like the vector space model, which represents text documents as vectors. A detailed description of these ranking models is not relevant in this thesis, as semantically enriched ranking can be added to any of these. One thing these models typically have in common is that they use the popular weighting scheme Term Frequency-Inverse Document Frequency(TF-IDF), a combination of Term-Frequency(TF) and Inverse-Document-Frequency(IDF), which allows for a way to weight documents based on how many times a term appears while also considering the term specificity.

Traditional IR technologies do not have the ability to exploit the knowledge granted by the structure and semantics of data on the Semantic Web (Jain and Singh 2013). The use of TF and TF-IDF is a common measure used to calculate relevance for documents in an IR context. When dealing with semantic data, it might not be how many times a term occurs in a document that is interesting, but the semantic relations and other metadata connected to the terms.

### 2.3.3 Evaluation

When evaluating an IR system, the main measure to assess is how the user's information need was satisfied. Having standard approaches for the evaluation of the systems is important since techniques and technologies are in constant development. There are multiple metrics often used when evaluating IR systems. Precision and recall are two popular metrics and are both based on relevance. Precision is defined as the ratio of relevant and retrieved documents to the total retrieved documents.

$$precision = \frac{|relevantdocuments \cap retrieveddocuments|}{retrieveddocuments} \tag{1}$$

Recall is defined as the ratio of relevant and retrieved documents to the total relevant documents.

$$recall = \frac{|relevantdocuments \cap retrieveddocuments|}{relevantdocuments} \tag{2}$$

Precision and recall are both commonly used as evaluation metrics. Other commonly used evaluation metrics, which are based on precision and recall, are prevision at k (P@k) and recall at k (R@k). These calculate the precision and recall among the top k returned documents.

#### Rank-aware Evaluation Metrics

In information retrieval systems it is important that relevant items are returned as high up the list as possible. In order to evaluate this, rank-aware evaluation metrics are used. Some commonly used rank-aware metrics are Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG).

The Mean Average Precision (MAP) is a widely used metric in information retrieval and machine learning that provides a single score to evaluate the performance of binary classification models. In cases where the user is only interested in the top-ranked items, MAP will be a common choice. Another metric is MRR, which is a straightforward evaluation metric that calculates the average of the first relevant item for different searches.

Normalized Discounted Cumulative Gain (NDGC) has a similar goal to MAP, as both algorithms prioritize relevant documents high on the recommended list. However, NDGC surpasses MAP and MRR by enabling distinction between the relevance levels of entities. This is achieved by incorporating a logarithmic discounting factor to consistently assess rankings. NDGC, therefore, is a relevance metric that is commonly used to evaluate the effectiveness of information retrieval systems. It is particularly useful in cases where the relevance of items differs, and the user is looking for top-ranked results. By using the logarithmic discounting factor, NDCG can adjust the importance of higher-ranked items, and give them a higher score. Below are the stages of calculating the NDGC.

$$CG_p = \sum_{i=1}^{p} rel_i \tag{3}$$

$p$ denotes the elements in the recommended ranking in equation 3, and in the three following equations. $rel_i$ is the graded relevance of the result at position $i$ (gain).

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} \tag{4}$$

The denominator in equation 4 is a logarithmic reduction factor that penalizes proportionally to the position in the ranking. This equation is the standard Discounted Cumulative Gain (DCG) formula.

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{5}$$

In order to normalize the DCG, the Ideal Discounted Cumulative Gain (IDCG) is calculated, as in equation 5. It uses the list of relevant entities, ordered by relevance, in the set of entities up to position $p$. This ideal ranking is according to the relevance of the entities.

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{6}$$

Finally, in equation 6, the NDGC is calculated by dividing all DCG by the ideal value, IDCG.

### Standarized Datasets

The use of standardized datasets for the evaluation of information retrieval systems is common. These datasets consist of three parts, the data, a set of queries or topics which covers the information needs, and the given results for the queries. Having these given results as a ground truth makes for the possibility to compare all the retrieval systems against each other.

The first standardized dataset for evaluation was the Text Retrieval Conference (TREC), which was organized by the US National Institute of Standards and Technology (NIST). TREC is an annual event and operates in the following way: First, a test dataset and information needs are provided to the participants. The participants develop and test their retrieval systems against the test dataset and submit their results. TREC then evaluates these results and organizes a conference where the participants are able to discuss their systems. According to TREC's official website[6], the goal of this conference is to encourage research in information retrieval systems, increase the communication among different stakeholders, as well as providing an appropriate method for evaluation.

Another commonly used standardized dataset is INitiative for the Evaluation of XML retrieval (INEX). INEX was introduced in 2002 with the aim of establishing an infrastructure for the evaluation of retrieval of structured XML documents (Gövert and Kazai 2002). Similarly to TREC, INEX provides the test collection as well as the appropriate scoring methods for evaluation. In 2007 INEX started an entity ranking track, where the main objective was to return entities instead of documents (De Vries et al. 2008).

[6]https://trec.nist.gov/overview.html

# 3 Information Retrieval in Semantic Data

As mentioned in chapter 1, information retrieval in the Semantic Web has been a topic since the '90s, and a lot of previous work has been done on this. This chapter provides a detailed overview of the relevant previous work for this project. Specifically, the chapter contains a description of various ranking models for semantic data, as well as a description of different approaches to achieve semantically enriched ranking.

## 3.1 Ranking Models

As mentioned in section 2.3.2 there exist various different ranking models, which vary in simplicity. For information retrieval in semantic data all of these traditional models can be used, assuming that the data can be gathered into different bag-of-words, making it possible to use TF-IDF weighting. Beyond the traditional models, there are also advanced models, that consider multiple fields (Balog 2018). This can be appropriate models for entity retrieval, as entities are divided into different fields/attributes, and it allows the model to weigh fields separately instead of gathering them all into a bag-of-words.

One of the more popular ranking formulas used to compute multiple fields is BM25F, which was introduced by Robertson et al. (2004). BM25F is a modification of the probabilistic BM25 ranking model, originally developed for text retrieval. BM25F stands out from BM25 by using the structure of the document in order to provide a more accurate ranking. In a regular text document, the title would most likely be one of the most important texts in the document. BM25F makes sure that these important parts of the document are given the highest weights. Blanco, Mika et al. (2011) describes an adaption of the BM25F ranking function for RDF data. Instead of breaking the document into title, body, etc., they use the RDF resource properties as fields. This version of BM25F uses field term frequency (number of times term appears in field), field length (number of terms in the field), and field weights as features. Pérez-Agüera et al. (2010) further tests BM25F for semantic data, and concludes that BM25F performs better when taking the document structure into account, however, it is unable to perform better in cases where the semantic information is in fields with less text.

Kim et al. (2009) introduced a different approach to set field weights called *probabilistic retrieval model for semi-structured data* (PRMS). PRMS do not use static field weights, but instead, use a dynamic method to determine the field weights on a term-by-term basis. They used Bayes' theorem to calculate the probability of mapping term $t$ to the given field $f$. Their approach shows improved retrieval efficiency compared to baseline methods. One important note about PRMS is that it assumes a homogeneous collection.

While there has been research on ranking models for semantic data, there is still a way to go to fully take advantage of the semantic information (Pérez-Agüera et al. 2010). Current common information retrieval methods do not improve the performance of semantic search systems. Pérez-Agüera et al. (2010) concludes that it is imperative that researchers consider how RDF and semantic data can be utilized to assist information retrieval functions, in order to advance semantic searches.

## 3.2 Semantically Enriched Ranking

The ranking models and methods described in the previous sections revolve around the term-based representation of entities, however, entities contain more information than just its term. As mentioned in section 2.2, entities consist of properties like attributes, types, and relationships, and using only term-based retrieval methods loses the advantages of this enriched semantic data. According to research, term-based models perform 25-40% worse on complex queries, like relationship queries or natural language questions (Balog 2018). For this reason, semantically enriched ranking is desired.

Semantically enriched ranking are methods that leverage the structured information in semantic data and can be added in addition to the term-based ranking to improve the retrieval methods. There are multiple ways of incorporating a semantically enriched ranking, and they often revolve around preserving and representing the semantic structure, as well as analyzing the query to create an enriched representation. The remainder of this section introduces various different methods for this.

### 3.2.1 Target Entity

One of the features of entities that can be leveraged in the ranking is the entity type, this can be utilized to improve search results. When a user is searching for something, they often have a specific type of entity or category in mind. As described in chapter 2.2, around 40% of queries are looking for a specific entity when searching online. For example, the query "*Author of Sult*" is most likely looking for the author Knut Hamsun. Identifying the target entity type makes it possible to narrow down the search results and provide more relevant information to the user.

Some search systems have the capability to specifically select the target type or category for the search, called faceted search (Tunkelang 2009). However, this type of information is explicitly given in this kind of system. When this information is not given, a possibility is to use the query or initial results to deduce what kind of entity type the search is looking for.

Some of the earliest research on this problem, Vallet and Zaragoza (2008), called this the entity type rank problem. Their approach retrieves passages instead of entire documents. In the information retrieval domain, a passage is a part of a document. They extract the 500 most relevant passages from their dataset before they find all entities that appear in these passages. They then find the relevant types by looking at the top-ranked entities and see which types they are a part of by using different weighting functions.

Balog and Neumayer (2012) introduced a new way to approach this problem, which they called hierarchical target type identification (HTTI). Their approach is based on the observation that entity types often are hierarchical. Their goal was to find the most specific type within the ontology that covers all relevant entities. Balog and Neumayer (2012) organize their entity types into hierarchical structures instead of flat list, which difference the approach from previous approaches. This leads to them needing more specific relationships between the relevant entities and the target type.

Both of these approaches are considered to be an entity-centric model. The entity-centric model is regarded as the most common approach for determining the target type of a query, according to Garigliotti et al. (2019). The idea behind the entity-centric model is straightforward. The first step is to rank the entities based on their relevance to the query, just like a normal information retrieval system would do. Then it uses the top-K ranked entities and aggregates the score for a given type, based on the relevance score of the entities with that type. Both Vallet and Zaragoza (2008) and Balog and Neumayer (2012) fall under this description, as they both extract the top-K ranked entities and are therefore considered to be entity-centric models.

An alternative to the entity-centric model is the type-centric model. The type-centric model's main goal is to analyze the query to find the relevant type or types. This approach builds a document containing descriptions of all entities that are labeled with that type. Once these documents are generated for each type, types can be ranked in a similar way as documents. This approach was introduced by Balog and Neumayer (2012). One of the results found in this experiment was that the type-centric model more often returns the correct type at the first rank.

Laclavik et al. (2015) introduced another type-centric approach to categorize the users intent with the query as well as a better representation of terms. Instead of operating on the level of single terms, they tried to capture richer information using term n-grams. They used the term n-grams to denote an order of words that commonly are used to describe a type. Their idea was that n-grams were more accurate to represent categories and types than single words. Words like "real estate" and "human rights" represent their categories well, but split into single words would remove the meaning. In addition to better representing types, n-grams can be used to detect the type directly from the query.

Table 4: n-gram example

| Book | |
|---|---|
| Book | 1.0 |
| E-book | 1.0 |
| Magazine | 0.364 |
| Comic book | 0.250 |

As shown in table 4, is an n-gram for the type *Book*. The seed concept, meaning the closest related terms, is given the highest score. Concepts derived from the neighbors of the seed concepts are getting a lower score. The score is calculated by using cosine similarity.

Once the target entity type has been identified, whether that is from an entity-centric model or a type-centric model, the next step is to use this target entity type to enhance the ranking accordingly. This can for instance be done by boosting every entity of the target entity type in the ranking. For instance, in the earlier example with the query "*Author of Sult*", one could boost every author entity in the ranking, giving the IR system a higher probability to find the relevant entity, *Knut Hamsun*. This is, however, based on the assumption that the target entity enrichment has managed to identify the right target entity type.

The difference between the entity-centric model and the type-centric model is in how the target entity type is identified. Where the type-centric model tends to return more specific types, the entity-centric returns general types. This was the expected result given the strategies of the models. Balog and Neumayer (2012), concluded that the entity-centric was the preferred model, but stated that an interesting approach to future work would be to combine features from both of them.

### 3.2.2 Graph-based Entity Search - Hybrid Approach

Another way to exploit semantic data is to take the graph structure and connections in the data into consideration in the ranking. Related entities in semantic data are often connected together, making it possible to find other related entities by traversing the graph from a given entity. For example in a bibliographic dataset, a series, its books, and its author will be connected together through various relationships (predicates), as visualized in figure 5. Tonon et al. (2012) proposes a simple hybrid ranking solution to leverage this graph structure to improve retrieval and ranking.

The hybrid solution consists of two steps, where the first steps use the standard term-based entity retrieval to retrieve entities and get an initial ranking. The second step uses the retrieved entities as seed entities and traverses the edges of these in the graph to identify related entities. To illustrate, the first step results in a list of retrieved entities.

$$Retr = \{e^1, e^2, e^3, e^4, ..., e^n\}$$

Top-N elements are then chosen as seed entities, for example, top-3 entities: $Seed\_entities = \{e^1, e^2, e^3\}$. The edges of these seed entities are then explored, resulting in a list of related entities. These related entities may already be in the list of retrieved entities, *Retr*, or be new entities added to the ranking. A new ranking of the entities can then be computed by the following formula (Balog 2018):

$$score(e; q) = \lambda_t * score_t(e; q) + \lambda_g * \sum_{e^{'} \varepsilon_{e^{'},e}} score_g(e; e^{'}, q)$$

The first component $score_t$ is the initial term-based ranking and the second component $score_g$ is a new ranking score based on the related entities. This means that related entities get an additional scoring for being close to a seed entity. For instance, in the graph example in figure 5, a search query for *"The Hunger Games"* can also return the entities of *Catching Fire*, *Mockingjay* and *Suzanne Collins* because of their closeness to *The Hunger Games* book or series.

The hybrid ranking method is highly relevant in many domains, as the users of search applications often do not explicitly mention the desired entities, but entities that are closely related to them

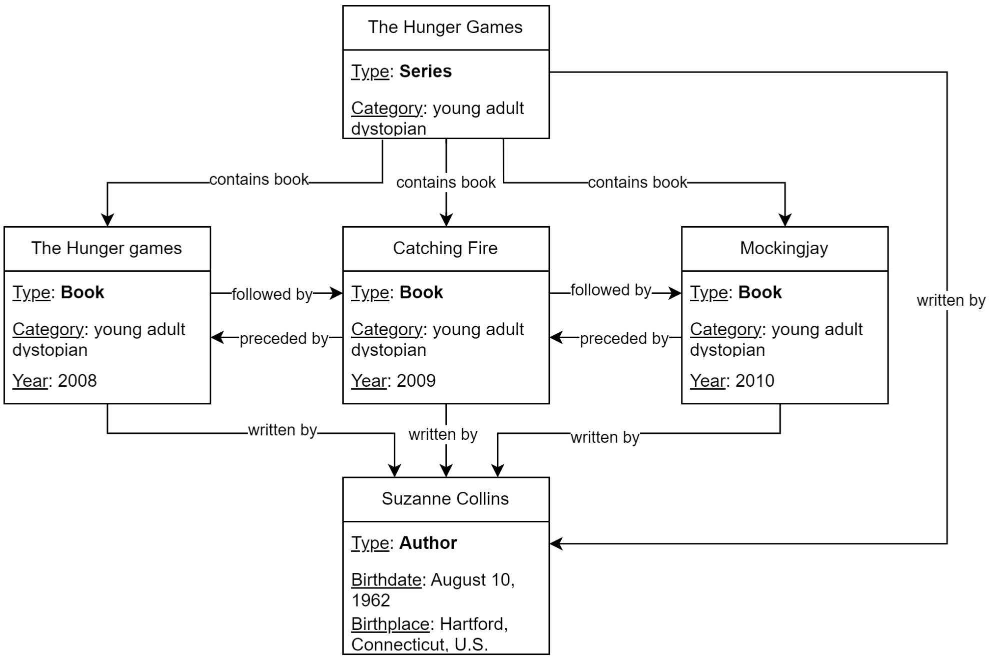


Figure 5: Graph of The Hunger Games series

(Tonon et al. 2012). In addition to this, the method also offers flexibility and the developers can choose which and how many predicates to traverse from the seed entities.

### 3.2.3 Entity Linking

Entity linking is another technique to achieve a semantically enriched ranking, which consists of recognizing and linking mentions of entities in the query to corresponding entities in the data (Hasibi et al. 2016). These corresponding entities can then be used to improve the entity retrieval and the final ranking. Entity linking is a technique utilized in various fields beyond just IR. It involves identifying and linking references of entities in a text with the corresponding entities in a dataset. It is for instance used for information extraction, by leveraging entity mentions in a text to extract more information on that particular object from a knowledge base. In IR it is also shown as an appropriate technique, as about 70% of queries in commercial web search engines contain mention of named entities (Guo et al. 2009). It is therefore highly relevant to use techniques to leverage these mentions and enhance the ranking accordingly.

There exists multiple research and techniques on entity linking in entity retrieval and how to leverage it. For simplicity, the task is often separated into two main tasks: the creation of entity annotation and the task of incorporating these in the retrieval and ranking.

To create entity annotations the system must analyze the query and try and detect entity mentions in the query. These entity mentions can then further be used to retrieve their corresponding entities from the dataset. One simple way to recognize entity mentions is to build a dictionary of all named entities in a dataset and compare the query terms to this dictionary. However, this approach has its limits, as building a dictionary for a large dataset can be too demanding. In addition to that, it can be useful to have a dictionary with different name variations, because of entity ambiguity, which again will make the task even more demanding. Another way to recognize entity mentions is to use named entity recognition tools, such as Stanford NER[7], OpenNLP[8], LingPipe[9] and TAGME[10], which helps to find the name of entities from a large pool of entity names in the query text. Using the mentions and linking them to corresponding entities can be a challenging

[7]https://nlp.stanford.edu/software/CRF-NER.shtml
[8]https://opennlp.apache.org
[9]http://www.alias-i.com/lingpipe/
[10]https://sobigdata.d4science.org/web/tagme/tagme-help

task, especially because of name variations and entity ambiguity. Still, there are various solutions publicly available to tackle this task.

One of the main approaches to retrieve these corresponding entities are named dictionary based techniques (Shen et al. 2014). These techniques entail building a dictionary containing various names and their mapping entities. For example, a bibliographic dictionary could be built by collecting every entity name and its mapping from a bibliographic knowledge graph, visualized in table 5. To further improve the dictionary it is possible to add name variations, abbreviations, spelling variations, nicknames, etc, of the different entities, to increase the chance of finding the relevant entity from an entity mention. Examples of this are visualized in table 6.

Table 5: Bibliographic name dictionary

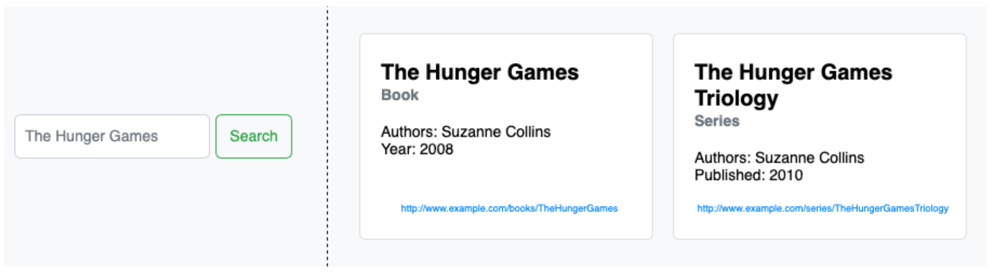
| Name | Entity |
|---|---|
| J.K Rowling | http://www.example.com/people/JKRowling |
| Suzanne Collins | http://www.example.com/people/SuzanneCollins |
| The Hunger Games | http://www.example.com/books/TheHungerGames |
| Catching Fire | http://www.example.com/books/CatchingFire |
| A Game of Thrones | http://www.example.com/books/GOT |

Table 6: Dictionary with added name variants

| Name | Entity |
|---|---|
| J.K Rowling | http://www.example.com/people/JKRowling |
| Joanne Rowling | http://www.example.com/people/JKRowling |
| A Game of Thrones | http://www.example.com/books/GOT |
| GOT | http://www.example.com/books/GOT |

Name dictionary is only one of the approaches that are possible. Wu et al. (2019) introduces a two-stage approach for zero-shot linking, with a bi-encoder and cross-encoder. Blanco, Ottaviano et al. (2015) proposes a new probabilistic model, which leverages information from query logs and anchor texts to automatically obtain aliases for the entities.

The first task in entity linking in entity retrieval results in a query, and one or more linked entities, visualized in figure 6. The second and next task consists of leveraging these linked entities in the ranking, i.e. incorporating the entity annotation in the retrieval and ranking. There are multiple ways of doing this. One way of doing this is to compare every entity in the retrieved ranking to the linked entities from the query. There are various aspects one can compare in the entities, for instance, it is possible to compare the term-similarity between the two (Hoffart et al. 2012). With a term-similarity comparison, one can for instance boost the entities that have similar terms with the linked entities further in the ranking, by giving the entities additional *points* in the ranking. Another approach would be to look at relationship and graph-structure, for example, finding similarity by looking at shared relationships (Jeh and Widom 2002), and then enhance the ranking accordingly. It is also possible to use similar techniques as described in the Graph-based entity search section. One can for example use Tonon's hybrid solution (Tonon et al. 2012), with linked entities as seed entities.



**The Hunger Games**
Book

Authors: Suzanne Collins
Year: 2008

http://www.example.com/books/TheHungerGames

**The Hunger Games Triology**
Series

Authors: Suzanne Collins
Published: 2010

http://www.example.com/series/TheHungerGamesTriology

Figure 6: Visualization of query and its entity links

Hasibi et al. (2016) describes another approach of incorporating entity linking into entity retrieval. They introduce a component called ELR (Entity Linking incorporated Retrieval), which can be applied on top of term-based retrieval models. ELR is based on the Markov Random Field (MRF) (Metzler and Croft 2005) model and they use the TAGME entity linking system to create the entity annotations. To use these entity annotations they, among other things, compare fields in the linked entities with fields in candidate entities to improve ranking.

To summarize there are many ways of finding entity links and incorporating these into the information retrieval. Entity linking makes it possible to utilize various entity attributes and relationships to improve the ranking, which is, as mentioned, one of the main goals in semantically enriched ranking. For instance, Hasibi et al. (2016)'s approach shows significant and consistent improvement to various state-of-the-art entity retrieval models.

### 3.2.4 Query Templates

The techniques target entity and entity linking can both fall under the category of query understanding/interpretation, as they can be dependent on interpreting the query to find target types or specific entities. One technique to recognize query intent, and perform rich query interpretation, is through query templates (Agarwal et al. 2010). Query templates are predefined patterns, formats, or structures that may exist in queries and may be used for interpreting them. The query templates are often stored in a dictionary or a taxonomy (Bortnikov et al. 2012). A simple example could be the query "writer of Game of Thrones" where the query template '*writer of*' could be used to infer that the user is looking for an author entity, which could be very useful when implementing target entity enrichment. Query templates could also consist of a sequence of terms and attributes, which enables the creation of specialized results based on the intent of the query (Agarwal et al. 2010).

Query templates can be crafted manually or extracted automatically. For example, Agarwal et al. (2010) proposes a way to automatically extract query templates from search logs. They analyzed a large-scale search log with 28000 queries and discovered that a high frequency of the queries followed structural patterns. They then developed the QueST framework, with probabilistic modeling and integrated inference, as a way to discover these patterns/templates and predict query intent. This approach avoids the limitation of hand-crafted templates, as the approach is possible to use on various datasets and domains. Through experiments, they showed that QueST gave accurate results and outperformed the baseline of Classify&Match. There are also numerous other algorithms for query templates discovery, for example in Natural Language Processing (NLP) (Dai and Callan 2019), Machine Learning (Beitzel et al. 2005), Collaborative filtering (Goldberg et al. 2001), etc.

Today, query templates are widely used in search engines and in IR technology. It had become a popular approach for presenting the user with easily readable results, often through specialized or vertical results. Figure 7 shows an example of Google's way to use a query template and give vertical results. Many other search engines also implement vertical search, like Yahoo[11] and Bing[12], as well as domain-specific IR technology, such as Amazon[13], Project Gutenberg[14], etc. In addition to being used for specialized results, query templates can be used to improve the ranking by offering query interpretations, for example for target entity type or entity linking.

### 3.2.5 Query-independent Ranking

The previous approaches have all been focused on ranking entities based on a query. There also exist ways to rank entities that are independent of the query. Query-independent ranking is helpful to determine the ranking of entities that have the same score for the query, as well as being able to rank entities entirely without a query. Balog (2018) describes the importance of query-independent ranking as the following:

[11]https://www.yahoo.com
[12]https://www.bing.com
[13]https://www.amazon.co.uk/
[14]https://www.gutenberg.org

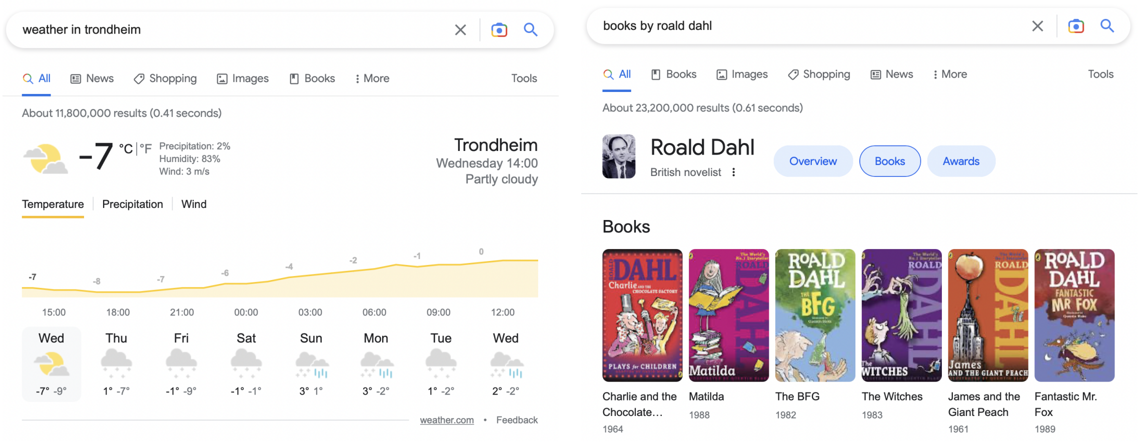


Figure 7: Google's vertical search

> Query-independent entity scores capture the notion of entity importance. That is, they reflect the extent to which a given entity is relevant to *any* query.

Popularity is a query-independent indicator that is popularly used in different entity retrieval methods. Metzger et al. (2017) uses Wikipedia's statistics for click frequency as a measure of its popularity. There are many possibilities for ranking entities after popularity in different domains. For a library, popularity may be measured in the number of times a book is rented, while the popularity of a movie can be ranked after the rating given by its viewers.

Popularity is not the only query-independent indicator that is of importance for semantic data and other graph data. Centrality is a feature often used in different retrieval models and is able to give the data a global rank, independent of the query. Entity centrality can be extracted from the graph structure and used to rank entities more accurately.

PageRank is one of the most popular centrality measures that calculate the importance of nodes. PageRank was introduced by Brin and Page (1998), in the same paper they introduce the world to the new search engine Google. There are multiple variations of the PageRank algorithm but this thesis will focus on algorithms for structured data. This algorithm takes advantage of the graph structure and uses nodes and connections between nodes to calculate a centrality score. Each node will get a score depending on its connections. Nodes with connections to nodes with a higher score will be more weighted than if their connections are more unimportant. According to Balog (2018), the formula for the PageRank of an entity is:

$$PR(e) = \frac{\alpha}{|\varepsilon|} + (1-\alpha)\sum_{i=1}^{n} \frac{PR(e_i)}{|\iota_{e_i}|} \tag{7}$$

In equation 7, $\alpha$, is used as a damping factor. This factor represents the probability of the user taking a jump to a random node. The score of all nodes is initialized with an equal score which is the damping factor divided by the number of outgoing links. Then all entities linked to entity e, divided by the number of outgoing links from $e_i$, are summed up and represent the PageRank score for entity e.

Since PageRank has been a popular centrality measure for multiple decades, the algorithm has been adapted to fit different kinds of fields and data. Some of these adjusted PageRank algorithms have been adjusted to fit semantic data and graph structures and are introduced in chapter 3.2.6 below.

### 3.2.6 PageRank Variants

As mentioned, several attempts have been made to adjust the PageRank algorithm to fit different types of data. When PageRank is explored and altered to fit different domains, it tends to acquire a new name. Pershina et al. (2015) presented a graph-based approach called Personalized PageRank that filters out incorrect candidates by only focusing on one entity at a time. Balmin et al. (2004) introduced ObjectRank which extends the original PageRank by sorting the database objects. The algorithm starts by finding the objects returned containing the keyword and will start the traversing of the graph from there. ObjectRank also differs from PageRank by weighting the edges and relationships of the graph. To do this, ObjectRank uses an Authority transfer schema, visualized in figure 8.

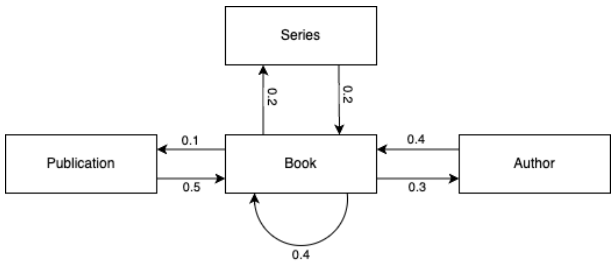


Figure 8: Authority transfer schema for ObjectRank

For example, let us say the damping factor is 15%, and the authority transfer rate between two nodes is 70%. This will give the following equation for the probability of the surfer traversing the edge.

$$p(traverse) = (100 - 15)\% * 70\% = 59.5\% \tag{8}$$

The damping factor $d$ can be adjusted. A high damping factor favors nodes adjacent to nodes returned by the query, while a lower damping factor will decrease the effects of the weights in the authority transfer schema. ObjectRank depends on manual tuning of the weights. This leads to the possibility that the entity returned by the system might not contain any keywords from the user query but is related to several entities that do. ObjectRank differs from PageRank since it is query-dependent and is calculated during query-time, as well as using weighting to give popular nodes a higher probability of being traversed.

Hogan et al. (2006) introduces ReConRank as an alternative to PageRank for structured RDF datasets. They discussed some of the disadvantages of a global rank, with some datasets being extensive and having to be recomputed when new data is indexed. The relevance of the global rank is also questioned, meaning the results returned should be the ones with local centrality, and entities with global centrality might not be the best results given the query. ReConRank avoids this problem by being a dynamic ranking system invoked at query-time ranking data returned by the query. The resources returned from the user query are returned as well as the surrounding resources, where the variable $n$ determines how broad the search result will be. Then the ranking is performed over these subgraphs. The final result is a list of resources prioritized according to their centrality in these subgraphs. The main disadvantage of this approach is that the ranking has to be performed during query-time, which can lead to slower and more resource-demanding retrieval.

PageRank is still commonly used in information retrieval systems and is the base for new approaches in different fields of study. In RDF data where nodes are connected by relationships and centrality is an important metric. The quality of the ranking can be improved by the context of the data. New approaches based on the original PageRank have been introduced, both with global ranking and local ranking calculated during query-time. One of the main reasons PageRank is still relevant is because of its global rank which is precomputed and does not use additional resources during the time the query is executed.

This chapter has introduced different approaches to ranking models of semantic data as well as different methods to extract metadata from the query. Both why semantically enriched ranking is important for improved entity retrieval and how it can be done were covered.

# 4 Search Engine Development

As mentioned in the introduction (chapter 1), this thesis aims to investigate various methods and techniques for information retrieval on semantic data. In order to answer the research questions, a test search engine was needed. The search engine was developed with the main objective of incorporating various techniques of semantically enriched rankings and visualizing the results they provide. The techniques implemented in the search engine are described in chapter 5.

In addition to the search engine, it was also necessary to have a test dataset. The search engine needed data to retrieve and rank. Therefore a test dataset was chosen and utilized.

This chapter describes the various tools used for the search engine, followed by a description of the dataset and database setup. Lastly, the chapter also contains an overview of the final implementation of the search engine.

## 4.1 Tools

To develop the search engine, various tools were needed. For instance, the search engine utilized a database system, a backend framework, and a user interface tool, these are all further described below.

### Neo4j

Neo4j[15] is an open-source Graph Data Platform. At the core of Neo4j is the Neo4j Graph Database which provides a schema-free graph database with a built-in web application called Neo4j Browser. Neo4j has its own query language called Cypher. Cypher creates and retrieves relations and data without the use of joins.

Neo4j provides official drivers for multiple popular programming languages which makes it possible to connect the Neo4j database to other applications. Neo4j provides multiple official plugins for different use cases. One of these plugins enables the use of RDF data and is called neosemantics (n10s). n10s simplifies importing both RDF data and ontologies to the Neo4j Graph Data Base.

Neo4j's Graph Data Science (GDS) is a plugin that provides different graph algorithms which can be used by calling Cypher procedures. GDS firstly projects a graph, called *named graph*, meaning it does not perform its algorithms directly on the data stored in the main graph. GDS provides algorithms concerning multiple fields. Some of the most popular algorithms that GDS has marked as production quality is:

- Centrality
  - PageRank (described in chapter 3.2.5)
  - Eigenvector centrality
- Path finding
  - Djikstra Shortest Path
  - Breadth First Search
  - Depth First Search

### Spring Boot

Spring Boot is a web application framework based on Java, an extension of the original Spring framework[16]. Spring Boot provides the ability to create standalone applications with minimal

[15]https://neo4j.com
[16]https://spring.io/

configuration. Spring Boot comes with embedded HTTP servers. The benefits of this tool are to reduce time in development and free up time to focus on the functionality to be implemented.

#### React

React[17] is a JavaScript library popularly used for building user interfaces. React is component-based, where each component has its own logic and controls. This makes it possible to create complex and dynamic web applications while still reusing the code. React also supports server-side rendering which makes it suitable for applications that are complex and interactive. Its large ecosystem of supporting libraries and tools makes it a popular framework in web development.

## 4.2 Database

### 4.2.1 Data

To be able to test and explore IR in semantic data, a large dataset with many different types of nodes and relationships was needed. Datahub[18], a free powerful data management platform, contains many free available datasets. For instance, Wikidata[19], Yago[20], British National Bibliography[21], ISFDB[22], etc. Wikidata and Yago are very large datasets with both more than 10 million entities belonging to different domains. A decision was made not to proceed with these options, as it would be difficult to restrict it to a single domain and there was an interest in exploring domain-specific IR. Rather, the datasets belonging to the bibliography domain were discovered to be an intriguing choice, given their numerous established properties and relationships. British National Bibliography, is a bibliography dataset containing information and descriptions of books published in the UK over the last 60 years. ISFDB, Internet Speculative Fiction Database, is another bibliography dataset containing information on works in Science Fiction, Fantasy and Horror. When comparing the two datasets, ISFDB was the most understandable and clean dataset. With these reasonings, the ISFDB was selected as the chosen dataset to investigate IR on semantic data.

The ISFDB is a comprehensive database that encompasses a wide range of information related to the world of fiction bibliography, including details on authors, publications, titles, awards, and other related data. It is openly distributed and available under Creative Commons license[23], and is created as a voluntary effort with users' contributions. The dataset is available in SQL format and can be downloaded as MySQL files. The dataset is divided into 7 different types of records: authors, publications, titles, series, publishers, publication series, and awards. An overview of the most important tables and connections is shown in figure 9.

From figure 9 you can see the seven tables that represent the different types of records mentioned earlier. Similarly, these seven records can be defined to be the various types of entities in the dataset, and these seven will be referred to as the entity types of the database from now on. To summarize, the dataset contains these entities: authors, publications, publishers, publication series, titles, series, and awards, which all have their own properties and relationships.

### 4.2.2 Mapping

As mentioned, the ISFDB comes as a SQL dataset, and a mapping is needed to transfer the data from SQL to RDF. To do this an R2RML[24] mapping approach was chosen, as that gave more control of the connections and vocabulary of the data compared to direct mapping.

[17]https://react.dev
[18]https://old.datahub.io
[19]https://www.wikidata.org/wiki/Wikidata
[20]https://yago-knowledge.org
[21]http://bnb.data.bl.uk
[22]https://isfdb.org
[23]https://en.wikipedia.org/wiki/Creative_Commons_license
[24]https://www.w3.org/TR/r2rml/

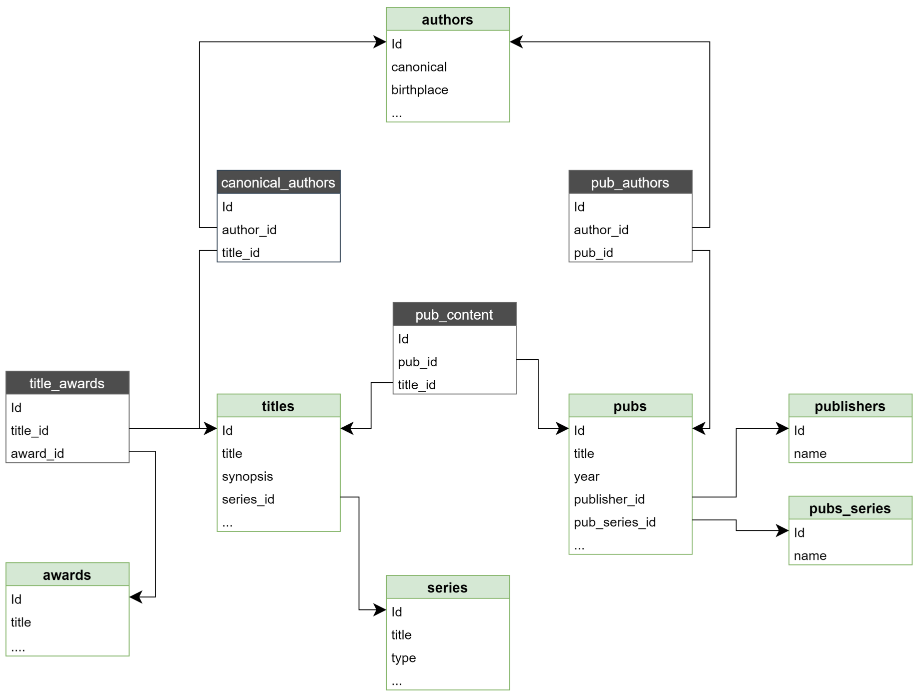


Figure 9: Simplified overview of the ISFDB dataset

For the vocabulary, RDA Registry[25] was chosen. RDA registry represents RDA entities, elements, and controlled terminologies in RDF, and is often used for bibliographic content.

The dataset was mapped to extract only the information that was deemed important, resulting in a reduction of ISFDB's overall content. The 7 entity types mentioned earlier were mapped using RDA registry classes that corresponded with each of these entities. Here is a list of each of these entities and their corresponding class.

Authors → rdac:C10004 (Person)

Publications → rdac:C10007 (Manifestation)

Publication series → rdac:C10007 (Manifestation)

Publishers → rdac:C10005 (Corporate body)

Titles → rdac:C10001 (Work)

Series → rdac:C10001 (Work)

Awards → rdac:C10003 (Item)

Each of these classes in RDA registry has corresponding properties, which were used actively to map the different columns from each table. Figure 10 shows the visualization of the RDF dataset that was generated from the mapping process.

### 4.2.3 Importing and Indexing

Once the mapping process was finished, the next step was to import the data into the database. The neosemantic plugin was utilized to import the RDF data into Neo4j.

When indexing the dataset, multiple considerations had to be taken into account. The RDF data consists of several entities with diverse properties and relationships, and the objective was to index

[25]https://www.rdaregistry.info/rgAbout/

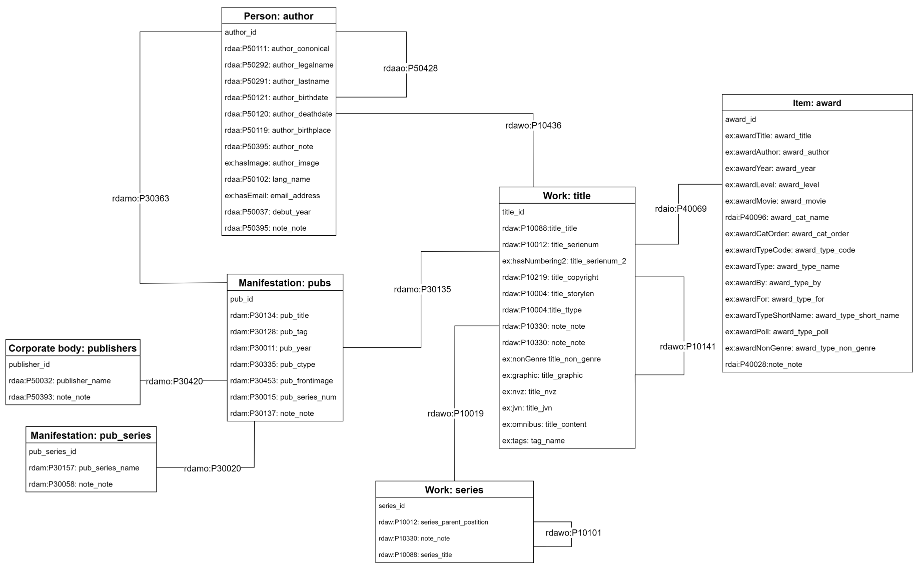

Person: author
author_id
rdaa:P50111: author_cononical
rdaa:P50292: author_legalname
rdaa:P50291: author_lastname
rdaa:P50121: author_birthdate
rdaa:P50120: author_deathdate
rdaa:P50119: author_birthplace
rdaa:P50395: author_note
ex:hasImage: author_image
rdaa:P50102: lang_name
ex:hasEmail: email_address
rdaa:P50037: debut_year
rdaa:P50395: note_note
rdaao:P50428
rdamo:P30363
rdawo:P10436
Item: award
award_id
ex:awardTitle: award_title
ex:awardAuthor: award_author
ex:awardYear: award_year
ex:awardLevel: award_level
ex:awardMovie: award_movie
rdai:P40096: award_cat_name
ex:awardCatOrder: award_cat_order
ex:awardTypeCode: award_type_code
ex:awardType: award_type_name
ex:awardBy: award_type_by
ex:awardFor: award_type_for
ex:awardTypeShortName: award_type_short_name
ex:awardPoll: award_type_poll
ex:awardNonGenre: award_type_non_genre
rdai:P40028:note_note
Work: title
title_id
rdaw:P10088:title_title
rdaw:P10012: title_serienum
ex:hasNumbering2: title_serienum_2
rdaw:P10219: title_copyright
rdaw:P10004: title_storylen
rdaw:P10004:title_ttype
rdaw:P10330: note_note
rdaw:P10330: note_note
ex:nonGenre title_non_genre
ex:graphic: title_graphic
ex:nvz: title_nvz
ex:jvn: title_jvn
ex:omnibus: title_content
ex:tags: tag_name
rdaio:P40069
rdawo:P10141
rdamo:P30135
Manifestation: pubs
pub_id
rdam:P30134: pub_title
rdam:P30128: pub_tag
rdam:P30011: pub_year
rdam:P30335: pub_ctype
rdam:P30453: pub_frontimage
rdam:P30015: pub_series_num
rdam:P30137: note_note
Corporate body: publishers
publisher_id
rdaa:P50032: publisher_name
rdaa:P50393: note_note
rdamo:P30420
Manifestation: pub_series
pub_series_id
rdam:P30157: pub_series_name
rdam:P30058: note_note
rdamo:P30020
rdawo:P10019
Work: series
series_id
rdaw:P10012: series_parent_postition
rdaw:P10330: note_note
rdaw:P10088: series_title
rdawo:P10101

Figure 10: ISFDB mapping graph

not just the titles of the entities but also their various properties so that they could be used in information retrieval. The built-in full-text indexer of Neo4j, Lucene full-text index, was therefore utilized to construct an index that could include all desired properties.

Additionally, term processing considerations were necessary. Given the dataset's numerous titles and proper names, it was deemed appropriate to avoid stemming and removing stop words. It was therefore decided to only remove punctuation. The full-text index was created with a Cypher query containing all desired properties to be indexed, the listing below shows the syntax for this.

```
CREATE FULLTEXT INDEX isfdbIndex FOR (n:Resource) ON EACH
[n.ns3__P50111, n.ns3__P50292, n.ns3__P50291, n.ns3__P50121, ...]
```

In addition to providing indexing, Apaches Lucene provides a variety of search features, including querying and ranking. By utilizing the full-text indexer, users can submit queries and obtain ranked results based on the frequency of terms. In other words, a term ranking has been created with the help of Neo4j's built-in library Lucene. Although this ranking produced suitable results according to the query's terms, it failed to utilize the benefits of semantic data. Thus, further proving the desire to investigate techniques that exploit the unique properties of entities in the data to achieve an improved ranking.

## 4.3 Web Application

A web application was developed to serve as the information retrieval system. As mentioned above, the objective of this search engine was to incorporate various techniques of semantically enriched ranking and visualize the result they provide. The functionality of the search engine was therefore to receive queries inserted from users, and return ranked datasets relevant to these queries.

### 4.3.1 Backend

The application's backend is a REST API implemented using Spring Boot. The api's task is simple, it receives queries from the frontend and returns ranked entities. To accomplish this, the API connects to the database using Neo4j's Java driver and leverages Apache Lucene's search feature to obtain term ranking for the query provided. Using the obtained ranking as a foundation, the API proceeds to utilize several semantically enriched techniques to further enhance it before sending it through to the frontend (read about the implementation of these techniques in chapter 5). A visualization of the application's architecture is shown in figure 11. Each semantically enriched technique results in an extra score that can be added to the initial ranking score, before it is returned to the user.

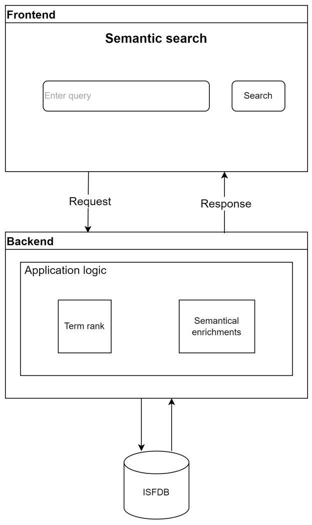
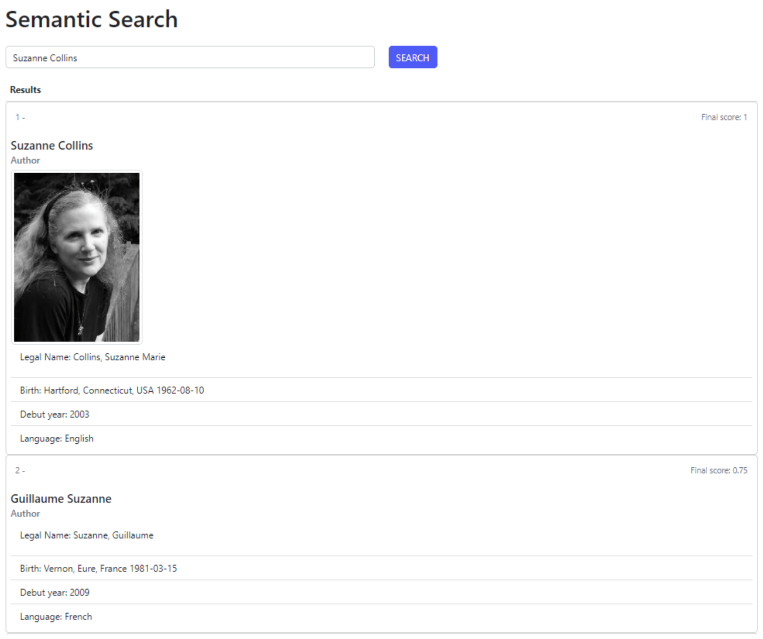


Figure 11: High-level application architecture

### 4.3.2 Frontend

The objective of the frontend was solely the design of the application. The frontend was implemented in React and comprised of a single page where users could input their queries and receive the ranked results. The main emphasis of the frontend was on creating a simple and user-friendly interface. An image of the design is shown in figure 12.



1 - Final score: 1

Suzanne Collins
Author

Legal Name: Collins, Suzanne Marie
Birth: Hartford, Connecticut, USA 1962-08-10
Debut year: 2003
Language: English

2 - Final score: 0.75

Guillaume Suzanne
Author

Legal Name: Suzanne, Guillaume
Birth: Vernon, Eure, France 1981-03-15
Debut year: 2009
Language: French

Figure 12: The application interface

# 5 Methodology

This chapter describes the various chosen semantically enriched ranking techniques and how they are implemented into the system. Four different fields are explored:

- Target entity
- Tonon's hybrid rank
- Entity linking
- Centrality

These fields yield six additional scores to the term-rank, which will later be assessed and validated in chapter 7.

## 5.1 Target Entity

The first enrichment implemented was target entity. As stated in chapter 3.2.1, this approach involves identifying the target types and leveraging them to enhance the ranking. Since the search system is not a version of faceted search, it was necessary to extract the target type from either the query or the initial ranked results. It was therefore desired to examine both entity-centric and type-centric techniques, mentioned in chapter 3.2.1, and implement two basic approaches to these.

**Entity-centric**
The entity-centric is regarded as the most simple and common approach for determining target types, which comprises using the initial ranked result to extract the target types. An approach to achieve this is by utilizing the top-K ranked entities obtained from the initial term rank and identifying the most frequent types within this subset. With this information, it becomes feasible to assign an additional score to every entity belonging to these frequent types. To demonstrate, the top-3 outcomes of a random query are examined.

J.K. Rowling *(Author)* → Score: 1.0

Harry Potter and the Philosophers Stone *(Title)* → Score: 0.84

Harry Potter and the Goblet of Fire *(Title)*→ Score: 0.80

In this example, there is one *author*-type and two *title*-types in the top-3 results, meaning that these two types can be viewed as the two target types of the ranking. With these two target types, it is possible to improve the original ranking by assigning a target type score to each of these types. One way of doing this could be to allocate a target entity score of *0.33* to author entities and *0.66* to title entities, as this corresponds to the distribution in the top-3 ranking. The resulting ranking score will therefore look like this.

For *Author*-entities → $score(e;q) = \lambda_t * score_t(e;q) + \lambda_{ec} * 0.33$

For *Title*-entities → $score(e;q) = \lambda_t * score_t(e;q) + \lambda_{ec} * 0.66$

For the rest → $score(e;q) = \lambda_t * score_t(e;q) + \lambda_{ec} * 0.00$

Using the distribution of the top-K ranking is a valid approach for assigning a score based on the frequency of target types, however, it does not consider the relative ranking of the different types. A straightforward method to address this issue is to incorporate the term-rank score when ranking the target types, which involves aggregating the scores associated with each type. For instance, with the previous example, the new scores would be *1.0* for the authors and *1.64 (0.84+0.80)* for titles. After normalization, the scores would then be *1.0* for titles and *0.61* for authors.

The implementation of this was fairly straightforward. A map is created to store each entity type and an initial score of *0.0*. Next, a loop is used to iterate through every entity in the top-K ranking, and the term score is added to the corresponding entity type in the map. After the map scores are computed, another loop is used to iterate through every entity and give each one an entity-centric score based on the normalization of the map score, which is calculated by dividing it by the max value. The code for this is shown below.

```
function entityCentricScore(k, rank) {
  // Creating a map for each entity type and initializing the scores to 0.0
  let typeMap = new Map();
  typeMap.set("authors", 0.0);
  typeMap.set("pubs", 0.0);
  typeMap.set("publishers", 0.0);
    ...

  // Iterating through the top-K entities to compute the scores for each entity
 type
  for (let i = 0; i < k; i++) {
    let type = rank[i]["type"];
    typeMap.set(type, typeMap.get(type) + rank[i]["term_score"]);
  }

  // Finding the maximum value from the map for normalization
  let max = Math.max(...typeMap.values());

  // Iterating through each entity to assign an entity-centric score
  for (let i = 0; i < rank.length; i++) {
    let type = rank[i]["type"];
    let type_score = typeMap.get(type);
    rank[i]["entity_centric_score"] = type_score / max;
  }

  return rank;
}
```

Each entity now has both a term rank score and an entity-centric score, which can be utilized in the entity ranking process.

$$score(e;q) = \lambda_t * score_t(e;q) + \lambda_{ec} * score_{ec}(e;q)$$

**Type-centric**

The second method for implementing target entity enrichment is known as the type-centric technique. This approach is considered slightly more complex than the entity-centric technique, as it involves interpreting queries to identify the target types. As described in chapter 3.2.1, the objective of the type-centric is to interpret the query to find the relevant target types. Once these relevant target types are identified, they can be utilized in a similar manner as in entity-centric ranking, thereby augmenting the ranking with an additional score.

There are multiple ways of interpreting the query to extract the relevant target types. One approach to this method, and the method used in this project, is by utilizing query templates which, as described in chapter 3.2.4, are predefined patterns, formats, or structures that might appear in the query. This is done by creating various templates for each entity type and then aggregating them in a map or dictionary, which can then be used to find type matches from the query.

Once a dictionary has been constructed, the query can be compared with the values in the dictionary to identify matches and thereby find the target types. For example, in the query "Who is the author of The Hunger Games?", a match can be found for the *author*-type if "*author of*" is a template for this type. After a target type has been found it is time to utilize this and score the entities accordingly.

Creating a type-centric score for the various types is a straightforward process. Typically, one would only find one match for a query and can therefore set the type-centric score for that type as *1.0* and the rest as *0.0*. However, to account for the queries with multiple matches one can divide it by the number of matches, giving the score *1 / (nr. of matches)* for each matching type. Figure 13 shows a visualization of this entire type-centric process.

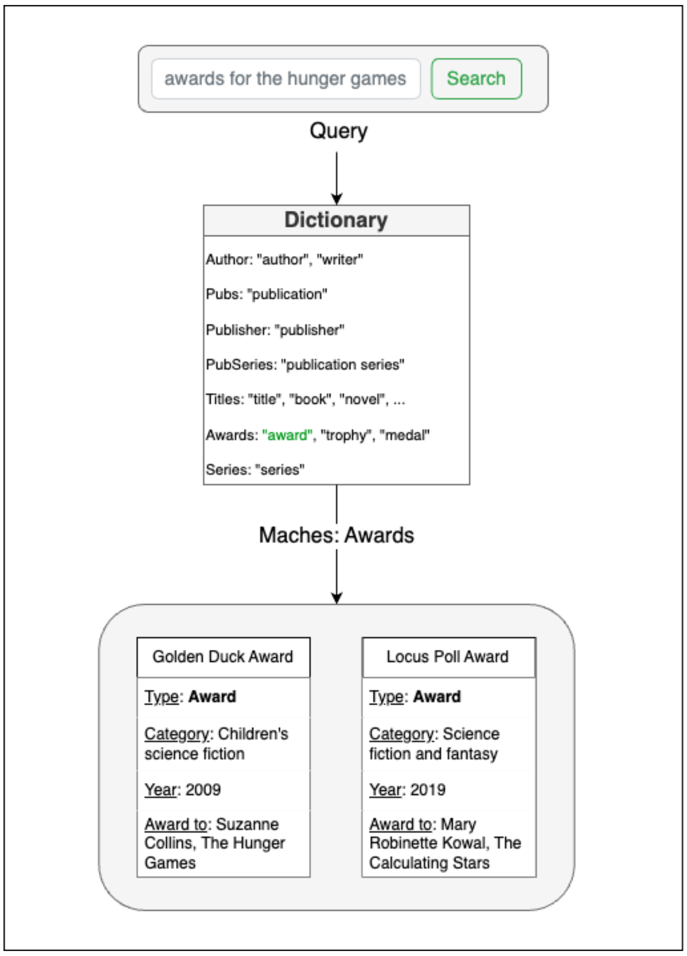

awards for the hunger games
Search
Query
Dictionary
Author: "author", "writer"
Pubs: "publication"
Publisher: "publisher"
PubSeries: "publication series"
Titles: "title", "book", "novel", ...
Awards: "award", "trophy", "medal"
Series: "series"
Maches: Awards
Golden Duck Award
Type: Award
Category: Children's science fiction
Year: 2009
Award to: Suzanne Collins, The Hunger Games
Locus Poll Award
Type: Award
Category: Science fiction and fantasy
Year: 2019
Award to: Mary Robinette Kowal, The Calculating Stars

Figure 13: An overview of the type-centric process

The implementation of this technique started with the construction of the query template dictionary. As mentioned earlier, the ISFDB contains 7 different entity types, and it was necessary to devise a range of query templates for each. Therefore, each entity type was analyzed separately and templates were developed for each one. This involved utilizing personal experience, as well as leveraging a thesaurus to identify synonyms and alternative names. While effective for this particular project, this approach may prove too cumbersome for larger and more complex datasets. Additionally, for bigger projects, a larger and more covering dictionary may be necessary, and the implementation of n-grams, as described in chapter 3.2.1, could be a potential solution. For example, Laclavik et al. (2015) offers a method for extracting n-grams from Wikipedia data. However, for the purposes of testing the type-centric model, the current approach was deemed sufficient. The resulting dictionary can be seen below.

Authors → [”author”, ”written by”, ”writer”, ”authors”]

Publication → [”publication”, ”publications”]

Publisher → [”publishers”, ”publisher”]

Publication series → [”publication series”, ”series”]

Titles → [”titles”, ”book”, ”novel”, ”bibliography”]

Series → [”series”, ”sequal”]

Awards → [”awards”, ”award”, ”medal”, ”trophy”]

After the construction of the dictionary, the rest of the implementation is straightforward and similar to the entity-centric approach. First, the query is checked against the values of the dictionary for any matches, and then a new score is generated for each entity based on the query matches.

```
function typeCentricScore(query, rank) {
        //Checking for matches in the dictionary
        ...

       // Iterating through each entity to assign a type-centric score
       for (let i = 0; i < rank.length; i++) {
         let type = rank[i]["type"];
         if(type = match) {
             rank[i]["type_centric_score"] = 1.0 / nr_of_matches;
         }
         else{
             rank[i]["type_centric_score"] = 0.0
         }
       }

  return rank;
}
```

As with the entity-centric approach, each entity now possesses a type-centric score, which can be utilized for ranking moving forward. This score is assumed to work well on *type-queries*, as it is based on extracting type information directly from the query.

$$score(e;q) = \lambda_t * score_t(e;q) + \lambda_{tc} * score_{tc}(e;q)$$

## 5.2 Tonon's Hybrid Rank

Tonon's hybrid rank enrichment differs from the target entity approaches in that it utilizes the graph structure of the ranked entities. Tonon's hybrid rank leverages the connections between entities in the ranking to enhance the overall ranking. To summarize the section from chapter 3.2.2, the objective of the hybrid solution is to improve the ranking by giving the neighbors of the top-ranked entities additional points.

The hybrid solution is deemed highly relevant in the context of IR on the ISFDB dataset due to the crucial role that connections play in the data graph. Suppose a user is searching for information on a book. In this case, it is highly likely that the user will find details about the book's author, publications, and series to be of interest. An example of the connections of a book can be seen in figure 14. These extra details will in many cases not appear in the initial ranking, as it is based solely on comparing the terms of the query with those in the indexed entities, and the user may therefore miss out on this valuable information. With the hybrid solutions, however, this information may be gathered from the connections of the top result and will therefore get a place in the ranking. Tonon's hybrid rank does therefore not only enhance the ranked entities in the initial ranking but may also add extra entities to the ranking, resulting in a new result list for the query. Thus, the hybrid solution may be highly beneficial in facilitating a more comprehensible and diverse set of results and ranking for the user that better addresses their information need.

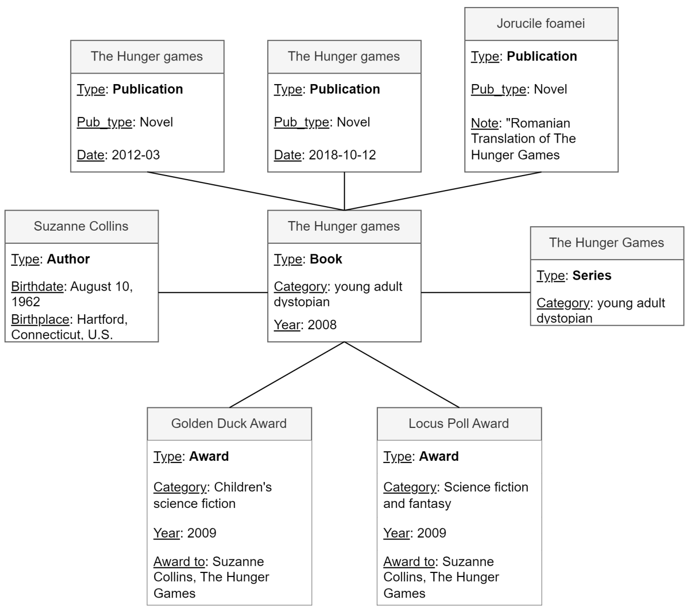


Figure 14: Visualizations of the various connection a book entity can have

As described earlier in section 3.2.2, the hybrid solution consists of two steps. The first step involves generating an initial ranking of entities based on the user's query. Then, in the second step, seed entities are identified from this initial ranking, and their related entities are determined by traversing their edges. The seed entities refer to the top-K entities from the initial ranking. Once the related entities have been identified, a new score for the ranked entities can be calculated, $score_{th}$, based on their *closeness* to the seed entities.

An example of this can be where the book *Alice's Adventures in Wonderland* appears as the seed entity. The hybrid solution will then traverse the edges of the entity in the graph, finding the title's closest connections. For example, the author entity *Lewis Carroll*, the series *Alice*, two award entities, and the various publications of the book are found as related entities. Some of these entities might appear in the initial ranking beforehand, such as publications that may share similar terms with the query, while some may be new and thereby added to the new result list. With this new result list, a score can be given to each entity connected to the seed entity, which will be used in the ranking of entities. For simplicity this score can be set as 1.0 for the related entities and 0.0 for the rest, giving every related entity an extra boost in the ranking.

If one were to expand this example even further one could add more seed entities. With multiple seed entities, each seed entity is traversed and more related entities might be added to the result list. Additionally, it is possible for two or more seed entities to share a related entity, which would further enhance the related entity's relevance. To take this into account the Tonon's hybrid score is based on their *closeness* to each seed entity and not 1.0 for each neighbor as in the previous example. This will ensure that an entity related to multiple seed entities is given a higher relevance score than an entity related to only one. In addition to that the score is based on the initial term rank for each connecting seed entity, meaning that one entity being connected to the first seed entity will score higher than one being connected to the last.

The implementation of this differs from the previous implementations because it was necessary to extract additional information from the database, as it was no longer possible to solely rely on the information provided by the initial ranking. The implementation, therefore, starts with identifying the seed entities and using their URIs to extract information on their neighbors from the database. To accomplish this, a Cypher query is employed. The query is as follows:

```
MATCH (e {uri: $uri})-[*1]-(p) RETURN p
```

For each seed entity, the query is utilized to retrieve information about its neighbors, *p*, and the retrieved information is stored in a list along with the term rank score of the seed entity. If multiple seed entities share a neighbor, their scores are added together. This results in the neighbor list containing information on each neighbor in addition to the sum of the connecting seed entities' scores. The neighbor list is utilized to calculate a new score for the ranking by performing two steps. Firstly, the entities in the list that are not present in the original result set are added to the ranking. Secondly, every entity in the ranking that appears on the list is assigned a hybrid solution score, which is calculated using the connecting seed entities' score divided by the sum of every seed entity's score, the division is done to normalize the score. For the entities that do not appear in the neighbors list the score is set to 0.0. Thereby, the formula for the hybrid score is as follows:

$$score_{th}(e;q) = \frac{\sum_{e' \varepsilon connected\_entities} score_t(e', q)}{\sum_{e' \varepsilon seed\_entities} score_t(e', q)}$$

Similarly to target entity each entity receives this score to enhance the ranking.

## 5.3 Entity Linking

Entity linking is another enrichment that can be used to leverage the graph structure of the data and take advantage of the importance the connections have. Entity linking involves identifying and linking references of entities in the query with corresponding entities in the dataset. The objective of the enrichment is therefore to leverage entity mentions in the query and enhance the ranking accordingly.

As described in chapter 3.2.3 there are multiple ways of performing entity linking on queries, and one of the most simple methods mentioned is name dictionary-based techniques. This involves building a dictionary containing the names of the entities and their corresponding mappings. The dictionary makes it possible to identify any references to entities in the query by comparing the terms in the query with those in the dictionary. If a match is found the mapping is used to retrieve the linked entity, or entities, which can further be used in the ranking. The use of dictionary-based techniques is just one of several methods available for conducting entity linking. However, for the purposes and objectives of this project, it is considered the most suitable approach. The choice of entity linking approach is not considered significant importance, as the primary focus is on utilizing the linked entities in the ranking. For more information regarding the various entity linking approaches, read the articles referenced in chapter 3.2.3.

Once the dictionary and query have been utilized to identify linked entities, the next step is to leverage them to enhance the ranking. Chapter 3.2.3 contains a description of multiple ways of doing this, where for instance entity comparison is mentioned. Entity comparison entails comparing the linked entities to the entities in the ranking. With this comparison, the entities that are similar to the linked entities can be viewed as more relevant and given an additional score in the ranking. Multiple factors can be used to compare two entities, for example, term similarity, relationships, specific properties, etc. Figure 15 shows an example of entity comparison based on term similarity, where the similar terms are marked in green.

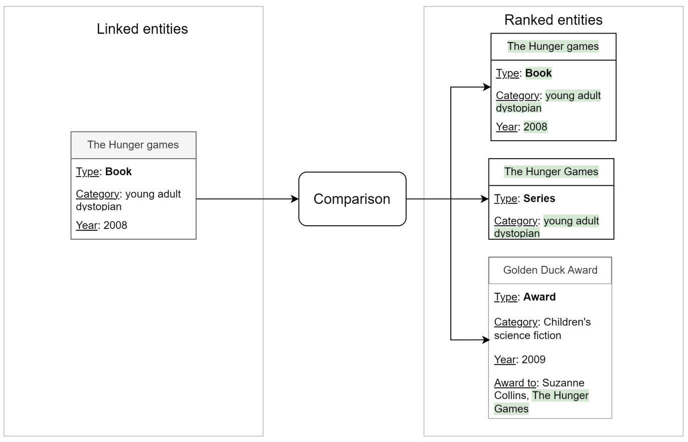


Figure 15: Illustration of entity comparison based on term similarity

When considering the ISFDB dataset it was challenging to identify scenarios where entity comparison could effectively improve the ranking and generate more desirable results for the user. For instance, if the entity *J.K. Rowling* is a linked entity, one could presume that the user is interested in information on the author and her numerous publications and books. However, entity comparison would not be useful in discovering this information, as the author entity has very little in common with the various books authored by her. Consequently, another approach was explored for utilizing the linked entities, which was deemed a better fit for the dataset.

The new approach resembles Tonon's hybrid solution, as the graph structure and connections from the linked entities are leveraged to improve the ranking. Therefore, instead of comparing the linked entities to the entities in the ranking, the connections of the linked entities were leveraged and the entities close to the linked entities were deemed more important. In the case of the linked entity *J.K. Rowling*, the books, publications, and awards connected to the author would with this solution get highlighted and get an extra boost in the ranking.

In summary, entity linking enrichment consisted of two steps: firstly, utilizing a named dictionary to identify linked entities, and secondly, utilizing the graph structure to identify the neighbors of the linked entities and assign them extra relevance in the ranking.

In order to implement this into the system, the named dictionary had to be constructed. This was done with the use of Chronicle Map[26]. Chronicle map is fast, in-memory, key-value stored designed for low latency. The creation of the dictionary involved first building a persistent dictionary map. Then, each entity in the dataset was iterated through, and for each one, the entity was added to the dictionary where the name was set as the key and the URI as the value.

```
static ChronicleMap<String, ArrayList> persistedDictionaryMap;

try{
    persistedDictionaryMap = ChronicleMap
            .of(String.class, ArrayList.class)
            .name("dictionary-map")
            .entries(5000000)
            .averageValueSize(100.0)
            .averageKeySize(100.0)
            .createOrRecoverPersistedTo(new File("/entity-details.dat"));
}
catch (IOException ie) {
    ie.printStackTrace();
}

entities.forEach(entity -> {
    String name = ((String) entity.get("name")).toLowerCase();
    String uri = (String) entity.get("uri");
    persistedDictionaryMap.put(name, uri);
});
```

[26]https://github.com/OpenHFT/Chronicle-Map

The next step after constructing the dictionary was to compare the query to the dictionary keys to find linked entities. Initially, the approach involved searching for any keys in the dictionary that appeared in the query. However, due to the dataset containing numerous 1-2 letter named entities, this method was deemed ineffective as it resulted in too many irrelevant linked entities being listed for each query. To address this, a new method was adopted whereby matches were only sought based on the query's whole terms. The query was therefore split into individual terms, and all potential combinations were generated. The resulting term combinations were then checked against the dictionary keys, and if any matches were found, the corresponding URIs were added to a list of linked entities. This approach avoided the matches on 1-2 letter named entities that occurred randomly within a query term.

After completing the lookup and finishing the list of linked entities, it is now time to use these to build an entity linking score for the ranking. Similarly to Tonon's hybrid solution, the entities' URIs were used to find every neighbor of these entities. One entity might be a neighbor with multiple linked entities. To address this, a map was created to track every neighbor and the number of linked entities it was connected to. This allowed for ranking entities that are neighbors to multiple linked entities higher than those that are only neighbors to one. A difference between the entity linking score and Tonon's hybrid score is that the linked entities also deserve an extra score in the ranking. In the hybrid solution, seed entities are selected from the top-K entities in the ranking, and the objective of the enrichment is to give extra relevance to their neighbors. However, in entity linking enrichment, the linked entities are obtained from the query and dictionary, and may not appear as high in the original ranking. Therefore, it is important to consider them when creating an entity linking score, as they deserve higher relevance for being matched with the query terms. As a consequence of this, it was determined that each linked entity should be assigned a certain number of *points* for being a linked entity, and each connection to these entities would receive *points* which were lower than the linked entities' *points*. These *points*, were later normalized and used as an entity linking score. The distribution of *points* was explored, and 10 *points* for each linked entity and 1 for each connection was deemed appropriate for the dataset.

## 5.4 Centrality

Centrality is a commonly used metric in information retrieval, which can be derived directly from the graph structure by analyzing links between the nodes. As described in chapter 3.2.5, one of the most popular algorithms for calculating centrality is PageRank. Since PageRank was first introduced as a method for ranking websites in Google search results, it has been adapted to many different graph problems. For instance, Twitter uses PageRank in order to give recommendations for people to follow and it is used in insurance fraud detection systems looking for anomalies.

PageRank assigns a score to every node, which takes into account the number of links a node has, as well as the quality of these links. Another approach that has been implemented is a local centrality measure, as an alternative to PageRank.

### PageRank

Given the popularity of the PageRank algorithm, there are several graph databases and plugins that provide this algorithm. Neo4j, which is the chosen graph database for this thesis, provides PageRank as well as other centrality metrics through an official plugin, as described in 4.1. Through this Neo4j plugin, which is called GDS, it was possible to apply PageRank to the data stored in Neo4j. There are some considerations to note when using the PageRank algorithm.

- When there are no relationships going out of a group of nodes, it is considered a spider trap. That can cause the PageRank score of nodes outside this group to vanish after a few iterations.
- The score can sink in cases where nodes are creating infinite cycles. This is partially avoided by the damping factor, as well as the algorithm will stop iterating when convergence is reached.

- Dead-ends occur when a node has zero outgoing relationships.

The PageRank algorithm is run by first creating a named graph in Neo4j. This is done by selecting the relationships and nodes from the graph database that should have a PageRank score. After the named graph is created, the PageRank algorithm can be run on the graph. PageRank is run by calling a cypher query like this on the named graph:

```
CALL gds.pageRank.write("named-graph", {
    dampingFactor: 0.85,
    writeProperty: 'pagerank',
    scaler: "MinMax"
}) YIELD nodePropertiesWritten, ranIterations
```

In the algorithm above, the damping factor is set to 0.85. As described in chapter 3.2.5, a damping factor of 0.85 gives the surfer a 15% probability of jumping to a random node, instead of traversing the edges of the node the surfer is currently in. The maximum number of iterations is set to the default of 20 iterations, which has been deemed sufficient for this dataset since the scores reach minimal tolerance and are deemed stable after 8 iterations. The scaler used is MinMax, which ensures the PageRank score is in the range of [0, 1]. The MinMax scaler uses the following equation to normalize the scores:

$$p_{scaled} = \frac{p - min(p)}{max(p) - min(p)} \tag{9}$$

In equation 9, $p$ denotes the vector containing all PageRank scores for the nodes in the named graph. The output property of the scaler will be a list of numbers, which are the scaled PageRank scores.

One of the problems with this metric is that it is a global score, which may not be the best fit for all datasets or give the best representation of centrality for all data, especially for semantic data. Every time new data is indexed the algorithm has to be rerun. Another flaw with PageRank is that one entity could have a large part of the total number of links. In the ISFDB dataset, all publications that have an unknown author, are linked to the same entity, *uncredited*, which has a total of 65742 outgoing and incoming links. This will lead other entities to get their scores diminished, without the *uncredited* author entity being as relevant as the PageRank score of 1.0 implies. This problem can be exemplified by comparing *uncredited* to the world-renowned author *George R. R. Martin.* Even though Martin has over 2000 titles, publications, and awards connected to him in the data graph, he only gets a PageRank score of 0.026. Another concern with using PageRank on the ISFDB dataset is that more relevant authors do not necessarily publish more books than less relevant authors. This means that an unpopular author can have a large number of books and can create his own spider trap. In order to get a more balanced score based on entities returned from the query, a local score was implemented to compete with the existing PageRank and give a more accurate view of the returned data.

#### Local Centrality

The local centrality score was implemented as an alternative to the original PageRank. The local centrality algorithm resembles ReConRank, which is introduced in chapter 3.2.6. Unlike PageRank, it is computed at query-time rather than beforehand. The score is based on the subgraph of the entities returned from the initial ranking and is calculated from the number of links each entity has in this subgraph. Although this increases the resources used during query-time, it provides a different view of centrality for the ISFDB dataset, compared to PageRank. Given the need for minimal resource usage during query time, an implementation was chosen that does not iterate through the subgraph the same way as PageRank and ReConRank do. By only calculating the number of incoming and outgoing links, this algorithm provides an accurate representation of the most influential nodes in the subgraph without unnecessarily consuming additional resources. In this algorithm, the quality of the links in the subgraph is not considered.

To implement this algorithm, every entity returned from the term ranking was collected and traversed to retrieve their neighbors. These entities and their neighbors were considered to be the subgraph for which the new centrality ranking was to be found. To gather the entities and their neighbors for the subgraph, the following query was used:

```
MATCH (p)-[*1]-(n)
WHERE p.uri IN [entity1.uri, entity2.uri, ...]
RETURN p, n
```

This query traverses every entity of the initially ranked entities and returns all neighboring entities that are linked to them. The returned entities and their neighbors are considered the subgraph for which the local score is calculated. To calculate the count of links for each entity in the subgraph, the following query was used:

```
WITH n match (n)- [] - (m)
WHERE EXISTS
    {MATCH (m) WHERE m.uri IN [entity1.uri, entity2.uri, ...] }
OR EXISTS
    {MATCH (p)-[*1]-(m) WHERE p.uri IN [entity1.uri, entity2.uri, ...] }
RETURN DISTINCT n.uri AS uri, count(m) AS count
ORDER BY count(m)
```

$n$ is each entity being iterated through from the subgraph.

With this list of entities and count of links, a score could now be computed. The score was computed by normalizing the count, which was done by dividing each count by the maximum count in the list. And this score was added as the new local centrality score in the ranking.

# 6 Evaluation Strategy

To evaluate the effectiveness of the different enrichment methods and answer the research questions outlined in chapter 1, a comprehensive evaluation strategy was needed that considered various factors. This chapter covers this project's evaluation strategy. It begins by presenting the overall goal and objective for the evaluation. Additionally, it further delves into the queries utilized to evaluate the implemented enrichment, as well as a description on the strategy used to gather relevance data from a user survey. This is furthered followed by a presentation of the result of the survey. Finally, the chapter ends by introducing the evaluation metrics used for the evaluation.

## 6.1 Strategy

The primary objective of the evaluation is to determine if each enrichment method enhanced the ranking and assess the specific effects each technique has on it. To understand the real effect, it is crucial to evaluate the enrichments on multiple searches and information needs, given that different search queries have varying information needs.

Several experiments were therefore designed to apply different enrichment methods to a set of search queries and measure the resulting changes in ranking. The evaluation comprised different types of queries, further described in section 6.1.1, and multiple techniques, such as precision@n, Discounted Cumulative Gain (DCG), and other relevant evaluation methods, could be implemented to measure the results from the different rankings. Section 6.1.5 below contains an introduction to the metrics employed in this thesis.

To conduct these evaluations, a form of relevance data for each entity returned from the various queries is required. Since the dataset lacked a standardized benchmark for testing, alternative approaches were necessary to obtain this. It was therefore decided to conduct a survey. The survey will give information of relevance for each entity and therefore enable an evaluation with various metrics.

Overall, the evaluation strategy was designed to provide a detailed and comprehensive understanding of the effectiveness of different enrichment methods across various contexts and scenarios

### 6.1.1 Queries

To gain insight into how each enrichment method contributes to the search system, a range of queries seeking diverse types of information were needed. To achieve this objective, queries were developed that cover diverse information needs. Furthermore, considering the vast number of entities in the ISFDB-dataset that may not be common knowledge, creating the queries required taking this aspect into account. This is because the queries and results would be used in the survey, and it was desired that the users of the survey had a basic understanding of the queries and entities in order to determine their relevance for each query. This would allow them to better understand the results and make more informed decisions.

The queries that were chosen are grouped into four different query categories; type queries, named entity queries, descriptive queries, and list queries. Type queries are designed to seek out a particular type of entity and typically include keywords such as *book*, *author*, or *award*. Named entity queries, on the other hand, are queries that seek out a specific entity by including its name in the query, such as a book title, author name, or publisher name. Descriptive queries, in contrast, are less specific and are aimed at finding entities that match a given description, making them useful when a user lacks knowledge of an entity's name but has information about it. For instance, a descriptive query could be a book description. Finally, list queries are used when a user seeks a list of multiple related entities, such as all the books in a particular series or by a particular author.

The queries used in the user survey and used for further evaluation of the metrics implemented can be seen in table 7.

Table 7: Queries used to evaluate search system

| Type | ID | Query |
|---|---|---|
| Type query | *Q1* | Author of Harry Potter |
|  | *Q2* | The Hunger Games awards |
| Named entity query | *Q3* | Harry Potter and the Philosopher's stone |
|  | *Q4* | Suzanne Collins |
| Descriptive query | *Q5* | Harry Potter third year at Hogwarts |
|  | *Q6* | Jurassic Park aimed for young adults |
| List Query | *Q7* | The Hunger Games books |
|  | *Q8* | Harry Potter books |

As mentioned above, the dataset did not have a standardized benchmark for testing and did therefore not include information on the relevance of entities for each query. It was, therefore, necessary to obtain that with a survey. However, for the list queries, acquiring relevance data from users was unnecessary. This is because list queries seek specific entities, making it obvious which entities are relevant. For example, for query *Q7*, which is "*The Hunger Games books*", it is evident that only three entities, *The Hunger Games* books, are relevant. Because of this, only queries *Q1-Q6* are evaluated in the survey.

### 6.1.2 Result Set

Before the survey was created, a result set was obtained to evaluate each query. Since the purpose of the evaluation was to assess each enrichment separately and determine its effect on different information needs, it was decided to look into the result set for each separate enrichment. To achieve this, every query was executed on the search system with each enrichment technique as the focus. Here is an overview of the 6 different implementations of enrichment techniques that were set in focus for the various result sets:

- Term rank
- Target entity
  - Type-centric approach
  - Entity-centric approach
- Tonon's hybrid rank
- Entity linking
- Centrality
  - PageRank
  - Local centrality

The weighting which all of the enrichments were run with, is viewed in equation 10. The only exception is the original term rank which does not require any weighting.

$$finalscore = 0.2 * termrank + 0.8 * enrichmentrank \tag{10}$$

To ensure that the survey was manageable for the participants, a limit was set on the number of entities to consider. The result set obtained from each search was deemed too large for the survey, therefore only the top 5 results from each result set were included. Since there are many entities that are returned from multiple result sets, the total entities in the survey are well below the theoretical maximum of 35 entities. However, for the list queries, which are not a part of the survey, a larger result set was obtained. For these queries, it was noted where the relevant entities appeared in the ranking.

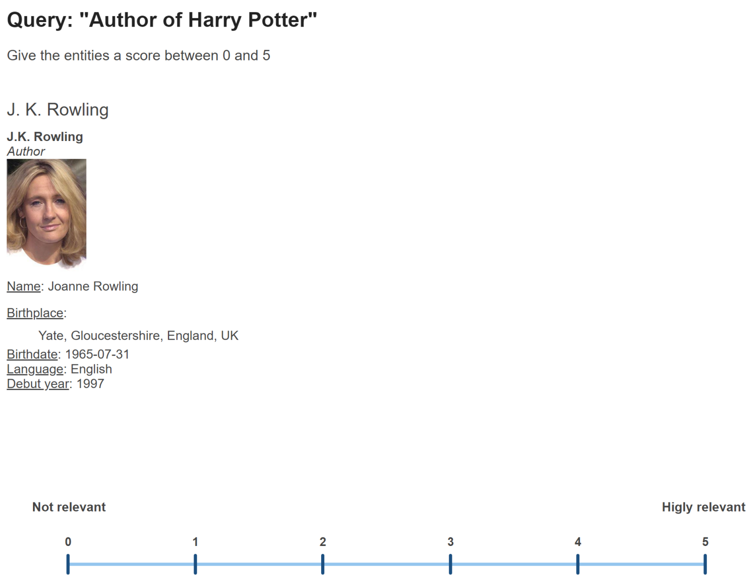
### 6.1.3 The Survey

With the 6 queries and various result sets, the objective of the survey is to label the relevance of the entities in the result sets for each query. The survey was implemented using Nettskjema[27], an online survey tool. Since the survey does not process any personal data, there was no requirement to notify the Norsk senter for forskningsdata (NSD).

When implementing the survey, various factors needed to be taken into account, and multiple methods were considered. The primary objective was to obtain relevance labels for the entities in the result set, although the methodology for achieving this was unclear. Multiple approaches were identified: One option was to allow the user to select their opinions on a ranking, for example, let them choose the top 5 relevant entities from the result set. However, this approach was found to be unsuitable for this project for several reasons. Firstly, the result set included various entities that were expected to be on the same level of relevance, making it challenging to rank one entity over the other. Secondly, it would have been difficult to compare the results, as a third-place entity in one result set might have a completely different relevance than a third-place entity in another result set, due to varying information needs and result sets.

Another approach possible involved assigning binary relevance labels (i.e., relevant or non-relevant) to each entity. This approach aimed to avoid incorrect results where one entity was ranked higher than another entity with the same relevance. However, this method does not take into consideration that some entities are of higher relevance than other entities. The first query, "*Author of Harry Potter*", can be an example of this. It is expected that the user will find the entity for *J.K. Rowling* to be of more relevance than the books of Harry Potter. And it was desired to have a survey that took this into consideration.

Therefore another approach was implemented, where the user had the possibility to rate the relevance of the returned entities by giving them a relevant score from 0 to 5, where 0 is not relevant at all and 5 is the most relevant. This solved the problem of having some entities being less relevant than others, but still not irrelevant. An example of the survey interface, where the user is given an opportunity to evaluate an entity is shown in figure 16.


**Query: "Author of Harry Potter"**

Give the entities a score between 0 and 5

J. K. Rowling

**J.K. Rowling**
*Author*

Name: Joanne Rowling

Birthplace:

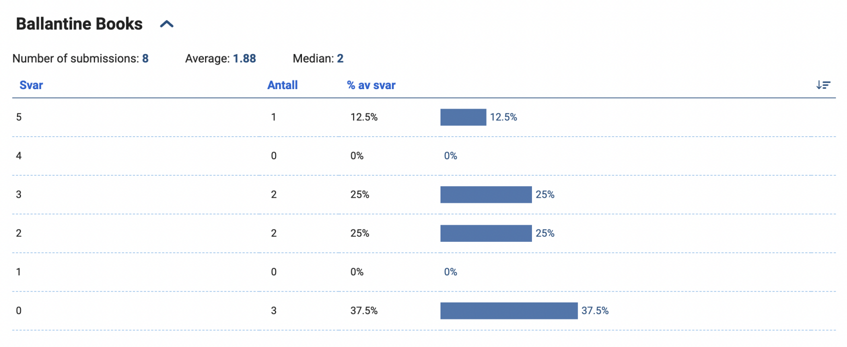
Yate, Gloucestershire, England, UK

Birthdate: 1965-07-31
Language: English
Debut year: 1997

Figure 16: Example from survey

When conducting the survey, another important factor to consider was the intent of each query. Instead of predetermining the purpose of each query, it was decided to allow respondents to rate the entities retrieved from the query and declare their own intentions. This approach was preferred

[27]https://nettskjema.no/



**Ballantine Books**

Number of submissions: **8** Average: **1.88** Median: **2**

| Svar | Antall | % av svar | |
|---|---|---|---|
| 5 | 1 | 12.5% | 12.5% |
| 4 | 0 | 0% | 0% |
| 3 | 2 | 25% | 25% |
| 2 | 2 | 25% | 25% |
| 1 | 0 | 0% | 0% |
| 0 | 3 | 37.5% | 37.5% |

Figure 17: Relevance result from query *Q6* for entity *Ballantine Books*

to avoid bias, as providing a description of the intended query could lead the surveyee to only focus on entities closely related to that intent. For example, with the query "*Suzanne Collins*", the obvious intent is the author entity, but individuals may differ in how they rank her books and other related entities. While some may find her books highly relevant to the query, others may only look for the specific author entity.

Lastly, the respondents of the survey had to be taken into consideration. It was necessary to select respondents who had experience using search systems. Additionally, having some basic knowledge of popular fantasy books and movies was desired, considering the nature of the dataset. However, possessing such knowledge was not mandatory, as respondents were allowed to search for additional information during the survey, for example, if they were unsure about the author of a particular book. It was important that the respondents made proper considerations when evaluating the results. With all these considerations in mind, the decision was made to not publicly share the survey online. Instead, a number of respondents were handpicked based on their experiences and knowledge of search systems and information retrieval. This ensured that the respondents had the right experience and made an effort when answering the survey.

### 6.1.4 Survey Results

In the previous chapter, the execution of the survey is explained. The total number of respondents was 8, which provides a good basis for using the data collected in further analyses. The relevance score for each entity was averaged and rounded to one decimal in order to retain readability and present the result set in an effective way. Another approach that was considered was to use the median instead of the average. This would give outliers less relevance. This was not the chosen approach since all answers and opinions were considered just as relevant.

The respondents have a high degree of consensus, but for some entities, the results are deviating. Especially for some types of entities, the relevance score is varying. For example for the query "*Jurassic Park aimed for young adults*", the results for the author and publisher of the books are spread out. While some believe the author of the books to be just as relevant as the Jurassic Park Junior Novelization books, others find the author to be almost irrelevant. An example of this can be seen in figure 17, the publisher *Ballantine Books*, which has published the *Jurassic Park* series.

In table 8 below, an extract of the top 5 entities ranked from the query *"Author of Harry Potter"* is shown.

Table 8: Top 5 results for query Q1

| *Query: "Author of Harry Potter"* | | | |
|---|---|---|---|
| **Score** | **Entity name** | **Type** | **SD** |
| 5.0 | J. K. Rowling | Author | 0.0 |
| 3.7 | Harry Potter | Series | 1.32 |
| 2.7 | Harry Potter and the Deathly Hallows | Book | 1.22 |
| 2.7 | Harry Potter and the Goblet of Fire | Book | 1.22 |
| 2.7 | Harry Potter and the Chamber of Secrets | Book | 1.22 |

Table 9 view the top 5 entities from the query *"Jurassic Park aimed for young adults"*. The author entity *Michael Crichton* is the author of both adult books and junior novelization books. This search is a descriptive query where the underlying goal is to match the free text in the properties of the entities. The junior novelization books got a description in the notes property, which claims that the books are "[...] aimed for young adults".

Table 9: Top 5 results for query Q5

| *Query: "Jurassic Park aimed for young adults"* | | | |
|---|---|---|---|
| **Score** | **Entity name** | **Type** | **SD** |
| 4.63 | Jurassic World: Special Edition Junior Novelization | Book | 0.70 |
| 4.63 | Jurassic World Dominion: The Deluxe Junior Novelization | Book | 0.70 |
| 4.64 | The Lost World: Jurassic Park: The Junior Novelization | Book | 0.70 |
| 4.38 | Jurassic Park | Series | 0.86 |
| 2.63 | Michael Crichton | Author | 1.73 |

The column containing standard deviation provides a measure of consensus. The standard deviation will measure the spread of scores around the mean. Entities with a high average relevance score and low standard deviation will indicate a high level of consensus among the respondents. Entities with a high standard deviation indicate disagreement among the respondents.

The main property in these tables is the relevance average, shown as the first column in the tables, *score*. This score will further be used to evaluate the various enrichment with different evaluation metrics. The evaluation metrics chosen for this evaluation are described below.

### 6.1.5 Evaluation Metrics

As mentioned earlier in this chapter, the reason for conducting the survey is to determine the relevance of the entities in the result set from various queries. With this information about relevance, it is possible to start evaluating the different enrichment techniques and determine their effect on different query types and information needs. There are several possible evaluation metrics to consider for this.

Since it is the rank that is evaluated in this thesis, a rank-aware evaluation metric is desired. While there exist several rank-aware evaluation metrics that would be interesting to use when evaluating the results, not all metrics take the level of relevance into account. For the last two queries, the list queries, this is not necessary as the relevance is binary and the entities are either relevant or not relevant. For these queries, the evaluation metrics of precision and recall have been chosen. This is because of the clear objective of the queries which is to retrieve every relevant entity and rank them high. Precision and recall will therefore give an adequate evaluation for evaluating the performance of these queries.

However, for the six other queries, the survey is set up to rank entities by level of relevance. Therefore a metric that takes this into account is desired for these. As described in chapter 2.3.3, Normalized Discounted Cumulative Gain (NDGC) has a similar goal to MAP, where both algorithms value relevant documents high on the recommended list. The main advantage this algorithm has over MAP and MMR is that it is able to distinguish between the level of relevance in the entities. As described in chapter 6.1.3, the survey was created with the goal of distinguishing between levels of relevance for the entities. This feature makes NDGC a good fit for the data since the relevant measures vary and are non-binary. By considering the importance of each relevant entity, NDCG provides a nuanced evaluation of the quality of the rankings, which will give a good evaluation of the overall performance of the system. In this thesis, the ideal ranking used to calculate the NDCG score is gathered from the results of the survey. More information about the calculation of NDCG can be seen in chapter 2.3.3.

The evaluation metrics employed in assessing the performance on the 6 queries are restricted to the top 5 returned entities because only the relevance of the top 5 entities retrieved by each of the six implemented metrics is labeled. Consequently, the evaluation is constrained by the limitation of labeled data. This may impact the accuracy and comprehensiveness of the evaluation. The reason for the restriction of rating only the top 5 entities in the survey was due to the practical constraints imposed to ensure the feasibility and manageability of the evaluation process for the survey participants. It is important to note that expanding the scope of the survey beyond ranking the top 5 retrieved entities could lead to a more comprehensive and reliable evaluation of the performance of the enrichments.

# 7 Results and Discussion

The previous chapters of this thesis have introduced and described the implementation of various enrichment methods for semantic data retrieval, including target entity, Tonon's hybrid rank, entity linking, and centrality. These methods have been integrated into a test search engine, enabling an evaluation of their effectiveness in improving search results. This chapter presents these search results, as well as a thorough evaluation and analysis of them. It starts by presenting the various results with a quick evaluation of this. The enrichment is then further evaluated and analyzed at the end of the chapter.

## 7.1 Enrichment Method's Result

This chapter presents the results of each enrichment and their evaluation, analyzing the strengths and limitations of each enrichment method. Specifically, the test search engine was used to conduct queries with different enrichment methods applied. The enrichments were applied one by one with a weighing of 20% term rank and 80% enrichment, to see the effects of each enrichment. The search result was then evaluated based on the relevance gathered from the survey result and used NDCG, relevance, and precision to further evaluate each result.

The objective of this evaluation was to assess the effectiveness of each enrichment method in improving search accuracy and to identify the effects each enrichment has on the search result. By evaluating and analyzing each enrichment result, valuable insight was gained into their strengths and limitation.

In summary, this chapter provides an overview of each enrichment's result with a critical evaluation of its performance, contributing to a deeper understanding of each enrichment and how they each can contribute to information retrieval for semantic data.

### 7.1.1 Term Rank

Before diving into any enrichment methods and evaluating their results, it is important to examine the original term rank and the outcomes it produced. The term rank utilized Neo4J's integrated full-text Lucene indexer, and the term rank in the search engine is implemented to retrieve 200 entities for each query. Below are the different outcomes generated by this ranking for the various queries.

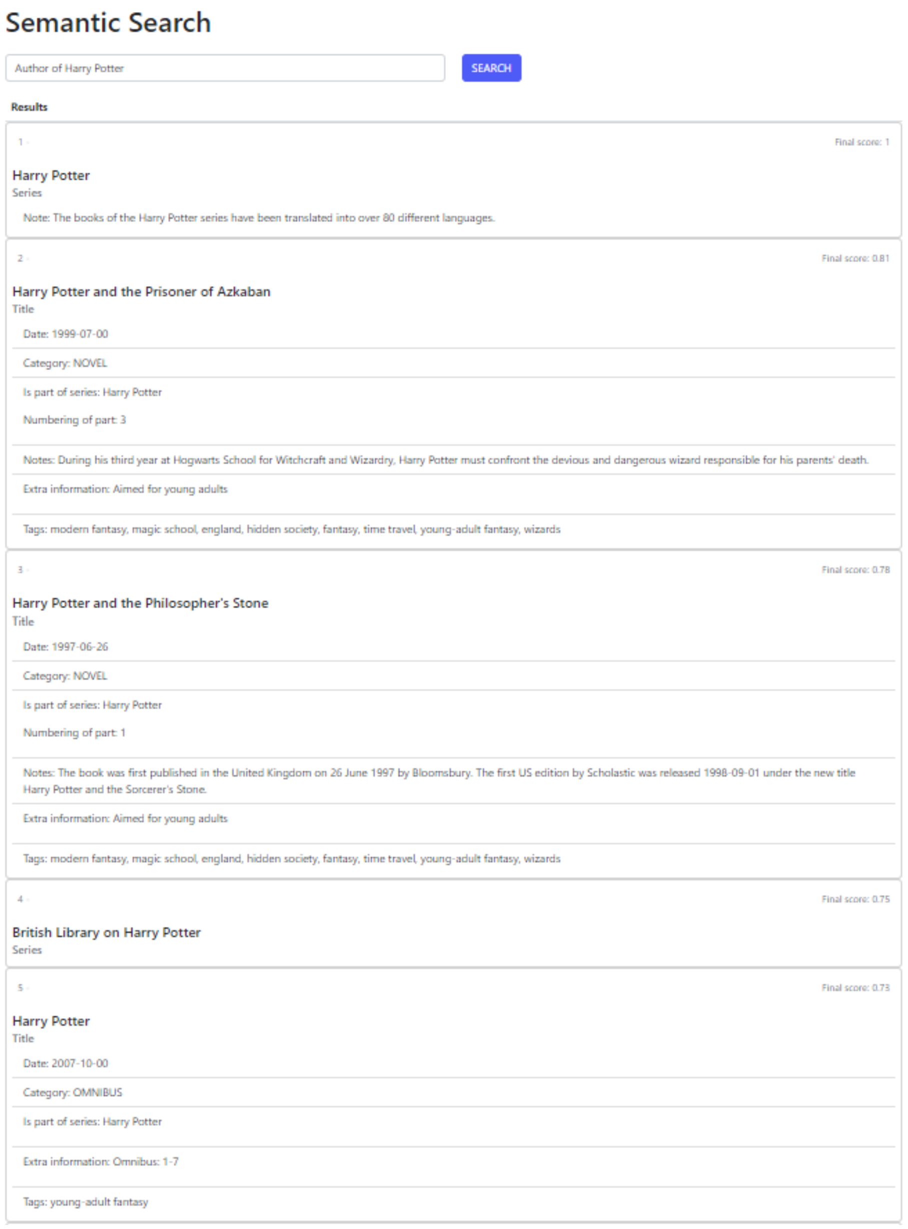
#### Type Queries

The first two queries used were type queries, which were designed to search for specific types of entities. The queries used were "*Author of Harry Potter*" and "*The Hunger Games awards*", referred to as *Q1* and *Q2*. These queries included the entity type they were seeking in the query term. This means the first query is searching for an author entity, while the second is searching for awards.

The first result to look at is from *Q1*. The top relevant entities gathered from the survey are shown in table 10 and the term rank result from the test search engine is shown in figure 18.

Table 10: Top relevance results for query *Q1*

| *Query:* "Author of Harry Potter" | | |
|---|---|---|
| **Score** | **Entity name** | **Type** |
| 5.0 | J. K. Rowling | Author |
| 2.63 | Harry Potter | Series |
| 2.0 | Harry Potter | Book (omnibus) |
| 2.0 | Harry Potter and the Deathly Hallows | Book |
| 2.0 | Harry Potter and the Goblet of Fire | Book |
| 2.0 | Harry Potter and the Chamber of Secrets | Book |
| 2.0 | Harry Potter and the Prisoner of Azkaban | Book |
| 2.0 | Harry Potter and the Cursed Child: Parts One and Two | Book |
| 1.86 | Harry Potter and the Philosophers Stone | Book |
| ... | ... | ... |
| 1.13 | British Library on Harry Potter | Series |



1 · Final score: 1

Harry Potter
Series

Note: The books of the Harry Potter series have been translated into over 80 different languages.

2 · Final score: 0.81

Harry Potter and the Prisoner of Azkaban
Title

Date: 1999-07-00

Category: NOVEL

Is part of series: Harry Potter

Numbering of part: 3

Notes: During his third year at Hogwarts School for Witchcraft and Wizardry, Harry Potter must confront the devious and dangerous wizard responsible for his parents' death.

Extra information: Aimed for young adults

Tags: modern fantasy, magic school, england, hidden society, fantasy, time travel, young-adult fantasy, wizards

3 · Final score: 0.78

Harry Potter and the Philosopher's Stone
Title

Date: 1997-06-26

Category: NOVEL

Is part of series: Harry Potter

Numbering of part: 1

Notes: The book was first published in the United Kingdom on 26 June 1997 by Bloomsbury. The first US edition by Scholastic was released 1998-09-01 under the new title Harry Potter and the Sorcerer's Stone.

Extra information: Aimed for young adults

Tags: modern fantasy, magic school, england, hidden society, fantasy, time travel, young-adult fantasy, wizards

4 · Final score: 0.75

British Library on Harry Potter
Series

5 · Final score: 0.73

Harry Potter
Title

Date: 2007-10-00

Category: OMNIBUS

Is part of series: Harry Potter

Extra information: Omnibus: 1-7

Tags: young-adult fantasy

Figure 18: Term rank result for *Q1*

Table 10 indicates that the *J.K Rowling* entity is the most pertinent, aligning with the objective of the entity type search, which aims to identify an author-type entity. The *Harry Potter* series and several of its books follow, as noted in the survey. However, these entities are more relevant to a term search focused on Harry Potter rather than the entity type being sought. The term ranking results in figure 18 support this observation, as they return the *Harry Potter* series and a couple of the books appearing in the top relevance result for *Q1*. This makes sense as the term rank is only focused on the terms of the queries and therefore retrieves results based on term relevance with "*Harry Potter*". However, it is worth noting that the term rank does not consider the entity type being sought, which explains why the most relevant entity is not being returned.

In order to evaluate the results from *Q1* using NDCG@5, as described in chapter 6.1.5, the ideal result set has to be calculated. For *Q1*, the equation of the ideal result set can be seen in table 10, with the top 5 results being 5, 2.63, 2, 2, 2. Put these numbers into the logarithmic reduction factor, which penalizes according to the position of the results, the top 5 results become 5, 1.66, 1, 0.86, 0.77. That sums up to 9.29, which is the IDCG@5.

The top 5 term rank results for *Q1*, got the following relevance scores: 2.63, 2.0, 1.68, 1.13, and 2.0. After the logarithmic reduction factor is applied, the scores will be 2.63, 1.26, 0.84, 0.49, and 0.77. The DCG@5, which is the sum of these numbers, totals 5.99. The NDCG@5 for *Q1* is calculated in equation 11 below.

$$NDCG_5 Q1 = \frac{DCG_5}{IDCG_5} = \frac{5.99}{9.29} = 0.65 \tag{11}$$

The entity considered the most relevant from the survey, *J. K. Rowling*, is not returned in the top 5 entities using the basic term rank. This was anticipated since the Lucene full-text search is looking for term matches from the query. The term rank performs well in retrieving the Harry Potter books but does not score well in the evaluation metric since these entities are not considered as relevant. A comparison of the NDCG@5 scores for every query can be seen in figure 19 at the end of the chapter.

Next up is the results for the second type search query: *Q2*. As previously stated, this is a type query seeking award entities. Table 11 displays the top relevant result from the survey. Similar to *Q1*, this relevance table aligns with the objective of the type search, with the top 4 relevant results being of the requested entity type, awards.

Table 11: Top relevance results for query *Q2*

| *Query: "The Hunger Games awards"* | | |
|---|---|---|
| **Score** | **Entity name** | **Type** |
| 4.63 | Golden Duck Award | Award |
| 4.63 | Locus Poll Award | Award |
| 4.63 | Grand Prix de I'Imaginaire (2010) | Award |
| 4.63 | Grand Prix de I'Imaginaire (2012) | Award |
| 2.88 | The Hunger Games | Series |
| 2.75 | The Hunger Games | Publication |
| 2.75 | Suzanne Collins | Author |
| 2.38 | The Hunger Games | Book |
| ... | ... | ... |
| 2.13 | The Hunger Games | Publication |
| 1.25 | The Hunger Games Tribute Guide | Book |

Table 12 presents the results from the term rank for *Q2*. As with the previous example, this is another instance where the term rank returns entities based on the terms of the query, resulting in entities with term relevance to "*The Hunger Games*" being returned. This illustrates that the term rank is not suitable for type queries where the objective is to find the specific entity type being sought, rather than just simple term matches.

Table 12: Top 5 term rank result for *Q2*

| *Query: "The Hunger Games awards"* | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Publication |
| 2 | The Hunger Games | Series |
| 3 | The Hunger Games | Book |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games Tribute Guide | Book |

The ideal result set can be seen in table 11. After the top 5 scores are reduced with the logarithmic reduction, the scores are 4.63, 2.92, 2.32, 1.99, and 1.07. The results from the term rank got the following scores, after the logarithmic reduction: 2.75, 1.82, 1.19, 0.92, and 0.48.

$$NDCG_5Q2 = \frac{DCG_5}{IDCG_5} = \frac{7.16}{12.93} = 0.55 \tag{12}$$

As seen in equation 12, the term rank does not score well for this query either. Since both *Q1* and *Q2* are type queries, this was the expected result as they look for entities not directly matched by terms in the query.

**Named Entity Queries**

Moving on to the next two queries, *Q3* and *Q4*, these are named entity queries that aim to identify specific entities by including their name in the query. *Q3* is the query "*Harry Potter and the Philosopher's Stone*", while *Q4* is "*Suzanne Collins*". The relevance result for these queries is presented in table 13 and table 14.

Table 13: Top relevance results for query *Q3*

| *Query:* "Harry Potter and the Philosopher's Stone" | | |
|---|---|---|
| **Score** | **Entity name** | **Type** |
| 5 | Harry Potter and the Philosopher's Stone | Book |
| 4.63 | Harry Potter and the Philosopher's Stone | Publication |
| 4.50 | Harry Potter and the Philosopher's Stone | Publication |
| 4.38 | Harry Potter and the Philosopher's Stone | Publication |
| 3.63 | Harry Potter and the Philosopher's Stone / Harry Potter and the Chamber of Secrets | Book (Omnibus) |
| 3.13 | Harry Potter | Series |
| 3.0 | J.K. Rowling | Author |
| ... | ... | ... |
| 1.75 | Harry Potter and the Prisoner of Azkaban | Book |

Table 14: Top relevance results for query *Q4*

| *Query:* "Suzanne Collins" | | |
|---|---|---|
| **Score** | **Entity name** | **Type** |
| 4.88 | Suzanne Collins | Author |
| 3.50 | Suzanne Collins | Book |
| 3.50 | Suzanne Collins | Publication |
| 2.50 | The Hunger Games | Book |
| 2.38 | Catching Fire | Book |
| 2.38 | Catching Fire (The Second Book of the Hunger Games) | Publication |
| 2.25 | The Hunger Games | Book |
| 2.25 | The Hunger Games | Publication |
| ... | ... | ... |
| 0.50 | Meghan Collins | Author |
| 0.13 | Guillaume Suzanne | Author |

Table 15 presents the results for *Q3*, which will now be examined in greater detail. When comparing the top 5 relevant entities with the top 5 results from the term rank, it becomes evident that the term rank performs better for named entity queries than type queries. Term rank returns several of the most relevant entities by matching the terms in the query with those in the names and properties of the entities. This approach is particularly effective in cases where the query and entities share common terms. As such, it makes sense that term rank would yield better results in this context.

Table 15: Top 5 term rank result for *Q3*

| *Query:* "Harry Potter and the Philosopher's Stone" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Series |
| 2 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter and the Philosopher's Stone | Publication |
| 4 | Harry Potter and the Prisoner of Azkaban | Book |
| 5 | Harry Potter and the Philosopher's Stone / Harry Potter and the Chamber of Secrets | Book |

For *Q3*, which is searching for one specific *Harry Potter* book, the IDCG@5 has a total score of 13.46. The DCG@5 from equation 4 is then calculated, and has a total score of 10.87. NDCG@5 for *Q3* gives the following calculation.

$$NDCG_5Q3 = \frac{DCG_5}{IDCG_5} = \frac{10.87}{13.46} = 0.81 \tag{13}$$

Next up, is *Q4*, which only includes the name of the author being sought, *Suzanne Collins.* The term rank for this query includes several relevant entities, but also a couple of irrelevant ones within the top five. It is likely that this occurs due to the limited number of search terms in the query. Entities with matches on one term, such as *Guillaume Suzanne* and *Meghan Collins*, receive a high score because they contain one of two terms in the query which gives a 50% match. As a result, *Q4* may not provide as accurate results as *Q3* with the term rank. The specific term rank results of *Q4* can be seen in table 16.

Table 16: Top 5 term rank result for *Q4*

| *Query: "Suzanne Collins"* | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Suzanne Collins | Author |
| 2 | Guillaume Suzanne | Author |
| 3 | Suzanne Collins | Book |
| 4 | Meghan Collins | Author |
| 5 | Suzanne Collins | Publication |

For the query *Susanne Collins*, the ideal result set according to the survey results have a reduced total of 10.84. That combined with the scores from the term rank, which are 4.88, 0.08, 1.75, 0.22, 1.35, gives the following NDCG calculation for *Q4*:

$$NDCG_5Q4 = \frac{DCG_5}{IDCG_5} = \frac{8.28}{10.84} = 0.76 \tag{14}$$

As expected, the term rank scores are better for named entity queries than for type queries. This is mostly due to the nature of the term rank, which looks for matches of terms in the queries and the indexed entities. The reason term rank scores better for these queries is mainly due to the most relevant entities being a named match with the queries.

**Descriptive Queries**

As described in chapter 6, descriptive queries are aimed at finding entities that match a given description. This evaluation consists of two of these: *Q5* and *Q6*. *Q5* is the query "*Harry Potter's third year at Hogwarts*" and *Q6* is the query "*Jurassic Park aimed for young adults*". Query *Q5* describes the third book in the Harry Potter series, while *Q6* describes the junior novelization of the Jurassic Park books as it is aimed at young adults. Below table 19 and table 20 show the relevance score of the most relevant entities for these queries.

Table 17: Top relevance results for query *Q5*

| *Query:* "Harry Potter's third year at Hogwarts" | | |
|---|---|---|
| **Score** | **Entity name** | **Type** |
| 5 | Harry Potter and the Prisoner of Azkaban | Book |
| 3.38 | Harry Potter | Series |
| 2.38 | Harry Potter | Book (Omnibus) |
| 2.38 | J.K. Rowling | Author |
| 2.13 | Harry Potter #1 - #5 | Book (Omnibus) |

Table 18: Top relevance results for query *Q6*

| *Query:* "Jurassic Park aimed for young adults" | | |
|---|---|---|
| **Score** | **Entity name** | **Type** |
| 4.63 | Jurassic World: Special Edition Junior Novelization | Book |
| 4.63 | Jurassic World Dominion: The Deluxe Junior Novelization | Book |
| 4.50 | The Lost World: Jurassic Park: The Junior Novelization | Book |
| 4.38 | Jurassic Park | Series |
| 3.13 | Jurassic Park | Publication |
| 2.88 | Jurassic Park | Book |
| 2.63 | Michael Crichton | Author |

Furthermore, the top 5 results for *Q5* and *Q6* with the term rank are presented in table 19 and table 20.

Table 19: Top 5 term rank result for *Q5*

| *Query:* "Harry Potter's third year at Hogwarts" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter and the Prisoner of Azkaban | Book |
| 2 | Harry Potter | Series |
| 3 | Harry Potter and the Philosopher's Stone | Book |
| 4 | Harry Potter | Book |
| 5 | British Library on Harry Potter | Series |

Table 20: Top 5 term rank result for *Q6*

| *Query:* "Jurassic Park aimed for young adults" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Filipino Fiction for Young Adults | Series |
| 2 | Jurassic Park | Publication |
| 3 | Jurassic Park | Series |
| 4 | Jurassic Park | Book |
| 5 | Jurassic World Dominion: The Deluxe Junior Novelization | Book |

First, the results for *Q5* are evaluated by examining the top 5 ranking of entities. A quick observation reveals that the ranking appears to be reasonably accurate, with three of the most relevant entities appearing within the top 5. These entities include *Harry Potter and the Prisoner of Azkaban*, *Harry Potter* (series), and *Harry Potter* (book).

To obtain a more detailed understanding of the results, NDCG@5 can be employed to evaluate the ranking for *Q5* further. The scores from table 17 are used to calculate the Ideal Discounted Cumulative Gain (IDCG). With scores of 5, 2.13, 1.19, 1.03, and 0.82, the total IDCG@5 for *Q5* totals 10.17. The top 5 term rank results for *Q5*, which are viewed in table 19, have a total score of 9.05 after the logarithmic reduction factor has penalized proportionally to the position of the result. Based on the results of the IDCG@5 and the DCG@5, the NDCG@5 is calculated for *Q5*.

$$NDCG_5 Q5 = \frac{DCG_5}{IDCG_5} = \frac{9.05}{10.17} = 0.89 \tag{15}$$

Based on the observation and the NDCG@5 score, it can be concluded that the ranking for query *Q5* yields appropriate results. To further provide an evaluation of term rank for descriptive queries, it is necessary to analyze the results for the other query as well; *Q6*.

Similarly to the results obtained for *Q5*, the top 5 ranking for *Q6* includes several highly relevant entities For instance, two of the entities returned in the ranking, *Jurassic Park* (series) and *Jurassic World Dominion: The Deluxe Junior Novelization*, have been assigned relevance scores above 4 in the survey.

Given the relevance results from table 18, the IDCG@5 is calculated to be 12.9. The term rank results viewed in table 20 give a total score of 7.45 after the reduction factor is used. The NDCG@5 for *Q6* is calculated in equation 16.

$$NDCG_5 Q6 = \frac{DCG_5}{IDCG_5} = \frac{7.45}{12.9} = 0.58 \tag{16}$$

This result indicates that the term rank does not perform at the same level for *Q6* as it did for *Q5*. The primary reason for this is that the top-ranked entity is deemed irrelevant, and the fifth-ranked result is shown to be the most pertinent among the five results. Still by observing the top

5 results without weighing the positions of the ranking, one could still say that the ranking yielded appropriate results given that four of the five entities were among the most relevant for the query.

The evaluation of term ranking for descriptive queries based on the results of *Q5* and *Q6* has yielded positive outcomes. This is consistent with the nature of descriptive queries, which aim to retrieve entities that match a given description. Since term rank relies solely on term matches between the query and entities, and the description of relevant entities often contains terms that appear in their properties, it can be expected to perform well in this context.

**List Queries**

The final type of query that requires evaluation is the list query, which aims to retrieve a series of related entities. These queries differ from the others in terms of how their relevance is evaluated, since the relevance of the entities is not based on the preceding survey. This is due to the specific objective of list queries, which offers a chance to have binary relevance. For example, consider *Q7*, which asks for a list of all the Hunger Games books. In this case, the relevance of each book is predetermined, and any entity that is not a Hunger Games book is considered irrelevant. This approach was adopted for the sake of simplicity, and also to differentiate it from the type queries. Although both *Q7* and *Q8* contains the name of an entity type, they were categorized as list queries to also assess this type of user intention. This decision was made to ensure that several query types are thoroughly evaluated, including those that might have characteristics of multiple types.

First, take a look at *Q7* which is seeking the Hunger Games books. There are three books total in the series and the name of these are: *The Hunger Games*, *Catching Fire*, and *Mockingjay*. Table 21 presents the top entities being retrieved by the term rank. This ranking only includes one of the three books in the top 5, which could be considered a poor result as the list query is intended to retrieve all the books. The last two books are not a name match of the query and are retrieved further down on the list with ranks 60 and 61.

Table 21: Term rank result for *Q7*

| *Query:* ”The Hunger Games books” | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Publication |
| 2 | The Hunger Games | Series |
| 3 | The Hunger Games | Book |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games | Publication |
| ... | ... | ... |
| 60 | Catching Fire | Book |
| 61 | Mockingjay | Book |

Query *Q8* are looking for the 7 books in the Harry Potter series, these are: *Harry Potter and the Philosopher's Stone*, *Harry Potter and the Chamber of Secret*, *Harry Potter and the Prisoner of Azkaban*, *Harry Potter and the Goblet of Fire*, *Harry Potter and the Order of the Phoenix*, *Harry Potter and the Half-blood Prince* and *Harry Potter and the Deathly Hallows*. Query *Q8* generates a similar outcome as *Q7*, with only two out of seven books appearing within the top 5 ranking (see result in table 22). It performs better than *Q7* since it retrieves all seven books in the first 26 entities. This is mainly due to all books having the terms ”*Harry Potter*” in their name.

Table 22: Term rank result for *Q8*

| *Query:* "Harry Potter books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Series |
| 2 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter and the Prisoner of Azkaban | Book |
| 4 | Harry Potter | Book |
| 5 | British Library on Harry Potter | Series |
| ... | ... | ... |
| 16 | Harry Potter and the Deathly Hallows | Book |
| ... | ... | ... |
| 19 | Harry Potter and the Chamber of Secrets | Book |
| 20 | Harry Potter and the Half-Blood Prince | Book |
| ... | ... | ... |
| 22 | Harry Potter and the Goblet of Fire | Book |
| ... | ... | ... |
| 26 | Harry Potter and the Order of the Phoenix | Book |

In order to evaluate the term rank for list queries, precision and recall were chosen as the primary evaluation metrics. Since list queries have a binary result set, NDCG is not the best fit. The relevance for the returned entities was selected to be the books from each respective query, and false for all other entities. The calculation of precision and recall were chosen to be @5, @10, and @15 to get a good overview of how the results behaved both at the first few returned entities, as well as further down.

The precision and recall for *Q7* and *Q8* can be seen in table 23 below.

Table 23: Precision and recall for *Q7* and *Q8*

| **Query** | **P@5** | **P@10** | **P@15** | **R@5** | **R@10** | **R@15** |
|---|---|---|---|---|---|---|
| *Q7* | 0.2 | 0.1 | 0.067 | 0.33 | 0.33 | 0.33 |
| *Q8* | 0.4 | 0.2 | 0.13 | 0.29 | 0.29 | 0.29 |

The results obtained from evaluating the term rank on the list queries, *Q7* and *Q8*, are unsatisfactory as they fail to retrieve most of the entities the queries are seeking within the first 15.

## Summary

As visualized in figure 19, term rank had varying results. Type queries had consistently low results for both queries, which was expected due to the nature of the information need. The expectations were higher for named entity queries, which are looking for exact name matches, and the term rank performed better than for type queries. Descriptive queries, which searched for information and descriptions in the metadata of the entity, had inconsistent results. The results for *Q5*, "*Harry Potter's third year at Hogwarts*", were close to retrieving the top 5 most relevant entities. *Q6* on the other hand, had the lowest NDCG@5 score of all the six queries. For the two list queries, term rank performed poorly. Term rank retrieved all relevant entities for *Q7* and *Q8*, but few of the entities were retrieved with an acceptable rank. This shows how inaccurate the term rank could be, and further explains the motivation behind implementing different enrichments to improve both the search results and the consistency itself.

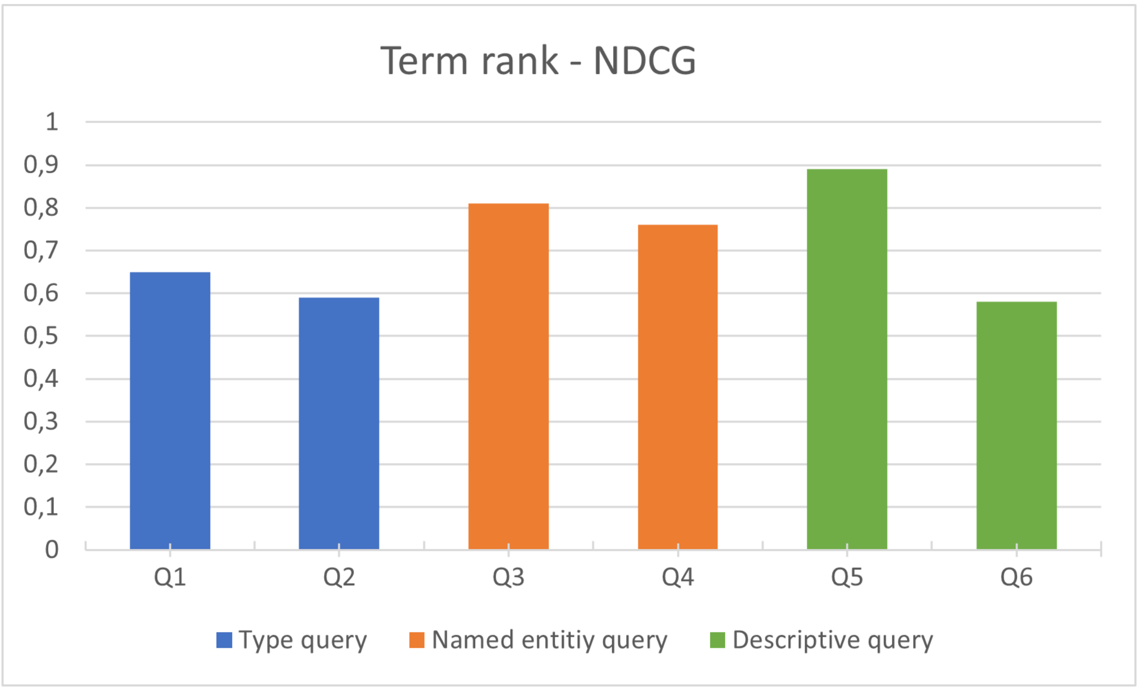


Figure 19: The NDCG scores for term rank

### 7.1.2 Target Entity - Entity-centric

The first enrichment evaluated is target entity. As described in chapter 3.2.1, target entity is a popular approach that involves identifying a specific entity type within a dataset and using it as a central point for ranking the related information. To evaluate the effectiveness of the target entity method in improving semantic data retrieval, the 8 queries, mentioned in 6.1.1, were used to conduct a series of tests using the test search engine. Since there are two types of target entity implementations, the evaluation was divided, and each implementation was evaluated separately.

Entity-centric was the first target entity approach implemented, and it consisted of using the original term rank to identify the most frequent types within the top results.

#### Type Queries

The first two queries, *Q1* and *Q2*, are the first to be examined. These queries are designed to search for specific types of entities and are classified as type queries. The objective of target entity enrichment suggests that this enrichment method would be effective for type queries since it aims to enhance the ranking of specific types of entities. However, there are two ways of implementing target entity enrichment, and they differ in how they extract these target entity types. The entity-centric approach, which is currently evaluated, utilizes the original term rank and boosts the most frequent types of entities within the top results. It will now be investigated whether this approach performs well on the selected type queries.

*Q1* contains the term ”author” and is seeking author-type entities. With *Q1*, the top 10 term rank results yield the normalized entity type scores below:

- Book: 1.0
- Series: 0.48
- Publication: 0.13
- Author: 0.0

- Publisher: 0.0
- Pubseries: 0.0
- Awards: 0.0

The computation of these scores involves adding up all the scores for the same entity type and then normalizing the sum by dividing it by the maximum value, as explained in chapter 5.1. However, the results show that the entity-centric approach is not able to identify the most relevant entity type for the query. Despite *Q1* seeking an author entity, the entity-centric score is only elevating the scores of books, series, and publications. This is because the term rank does not consider the entity type being sought, and instead retrieves results based on term relevance with *"Harry Potter"*, which is shown above in the term rank evaluation. Based on these scores, the entity-centric approach returns the entities listed in Table 24, which displays the top 5 results.

Table 24: Top 5 entity-centric result for *Q1*

| *Query:* "Author of Harry Potter" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter and the Prisoner of Azkaban | Book |
| 2 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter | Book |
| 4 | Harry Potter and the Cursed Child: Parts One and Two | Book |
| 5 | Harry Potter (excerpt) | Book |

The top 5 results consist of Harry Potter content, which is all of the entity type *book*. This entity type is the most boosted by the entity-centric enrichment, which explains why it appears as the most relevant entity type for *Q1*. When comparing these results to the top relevant entities from Table 10, it can be observed that most of the entities are among the top 3 highest relevance scores. However, it is important to note that although they are among the top 3 scores, they differ significantly from the top relevant entity, which is *J.K. Rowling*. Equation 17 illustrates the NDCG@5 result based on the top 5 entity-centric results and their relevance to the query. However, this score is even lower than the term rank result, indicating that the entity-centric approach does not improve the ranking for this particular query.

$$NDCG_5Q1 = \frac{DCG_5}{IDCG_5} = \frac{5.32}{9.29} = 0.57 \tag{17}$$

*Q2* is another query seeking a specific entity type, awards, which is mentioned in the query terms. As seen below, the entity-centric approach does not find the most relevant type once again, as it again only elevates the score for books, publications, and series. Again giving the highest boost to the book type.

- Book: 1.0
- Publication: 0.86
- Series: 0.45
- Awards: 0.0
- ...

With these type scores, the top ranking with entity-centric enrichment is shown in table 25. Similar to the result for *Q1*, this result also only consists of one entity type, book, which is the type considered the most relevant according to the entity-centric enrichment. When comparing this result to the relevance score in table 11 it is shown again that this enrichment does not improve the ranking. Equation 18 further proves this and shows that the NDCG@5 score is even lower than the original term rank score.

Table 25: Top 5 entity-centric result for *Q2*

| *Query:* "The Hunger Games awards" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Book |
| 2 | The Hunger Games Tribute Guide | Book |
| 3 | The Hunger Games Trilogy (box set) | Book |
| 4 | The Hunger Games 4-Book Set | Book |
| 5 | The Hunger Games: Official Illustrated Movie Companion | Book |

$$NDCG_5Q2 = \frac{DCG_5}{IDCG_5} = \frac{5.51}{12.93} = 0.43 \tag{18}$$

Based on these results, one could conclude that the entity-centric enrichment approach does not significantly improve the ranking for these particular queries, as it does not succeed in identifying the most relevant entity type.

### Named Entity Queries

Next, an evaluation was conducted to determine whether entity-centric enrichment could enhance the performance of the following two queries: *Q3* and *Q4*, both of which are categorized as named entity queries. In named entity queries, the search term includes the name of a particular entity.

For *Q3*, which is "*Harry Potter and the Philosopher's Stone*", the enrichment process identified a majority of entities related to book types in the top 10 search results. Consequently, this type of entity was given extra weight in the ranking. The top 5 results of the entity-centric ranking therefore exclusively comprised of book entities, as illustrated in Table 26.

Table 26: Top 5 entity-centric result for *Q3*

| *Query:* "Harry Potter and the Philosopher's Stone" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter and the Philosopher's Stone | Book |
| 2 | Harry Potter and the Prisoner of Azkaban | Book |
| 3 | Harry Potter and the Cursed Child: Parts One and Two | Book |
| 4 | Harry Potter and the Sorcerer's Stone (excerpt) | Book |
| 5 | Harry Potter and the Philosopher's Stone / Harry Potter and the Chamber of Secrets | Book |

Within the top 5 results, one can see several of the Harry Potter books or excerpts, which is also considered as a book type in this dataset. Three of these books are considered some of the most relevant entities from the survey, see table 13, and are highly related to the first Harry Potter book which the query is referencing. In addition to that, there are two other Harry Potter books, which are not rated as relevant. Still, the NDCG@5 result presented in equation 19 shows that the entity-centric enrichment gives appropriate results. However, the NDCG@5 score is lower than the result from the original term rank.

$$NDCG_5Q3 = \frac{DCG_5}{IDCG_5} = \frac{10.10}{13.46} = 0.75 \tag{19}$$

The second named entity query *Q4*, "*Suzanne Collins*", is the next query to be examined. In this scenario, the entity-centric enrichment process has identified *author* as the most relevant entity, given it has the highest frequency among the original top 10 search results. Similar to the previous query, *Q3*, the top 5 results of the entity-centric ranking, as presented in Table 27, exclusively consist of the most relevant entity type, author entities.

Table 27: Top 5 entity-centric result for *Q4*

| *Query:* "Suzanne Collins" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Suzanne Collins | Author |
| 2 | Guillaume Suzanne | Author |
| 3 | Meghan Collins | Author |
| 4 | Christophe Collins | Author |
| 5 | Mabel Collins | Author |

Nevertheless, there is a stark contrast between the results of *Q4* and *Q3*, since only one entity is deemed relevant within the top 5 ranking for the former. According to the survey, *Suzanne Collins* is the only author deemed relevant for *Q4*, with the remaining authors returned by the entity-centric ranking considered irrelevant to the query. This ranking, therefore, yields a poor outcome, as confirmed by the NDCG@5 score of *0.52* displayed in equation 19. However, it is worth mentioning that the entity-centric ranking does prioritize the highest relevant entity by ranking it as number 1. This implies that if a user is solely interested in the most pertinent result, then the outcome could be deemed satisfactory.

$$NDCG_5Q4 = \frac{DCG_5}{IDCG_5} = \frac{5.62}{10.84} = 0.52 \tag{20}$$

Based on the NDCG scores obtained for the two named entity queries, it is evident that the entity-centric enrichment does not enhance the ranking performance compared to the term ranking. The scores are lower for both queries, indicating that the entity-centric approach is not more effective than the term-based approach. Nevertheless, it is worth noting that the entity-centric ranking does retrieve the highest relevant entity for both queries and ranks it as number 1, which could be considered satisfactory for certain users.

### Descriptive Queries

The next type of query to evaluate the performance for the entity-centric enrichment are descriptive queries. This evaluation includes two descriptive queries: *Q5*, which searches for "*Harry Potter's third year at Hogwarts*" and *Q6*, which searches for "*Jurassic Park aimed for young adults*".

Similar to other queries, entity-centric enrichment creates scores for each entity type based on their appearance in the top 10 ranking. These scores for *Q5* are shown below.

- Book: 1.0
- Series: 0.41
- Publication: 0.0
- ...

Only two types receive a score above 0: books and series. Since books are considered more relevant, the top 5 ranking results from entity-centric consist only of books, shown in table 28. The NDCG@5 score, calculated in equation 21, for *Q5* is 0.75, indicating a good result. This makes sense as the highest entity in the ranking is rated the most relevant entity from the survey. In addition to the omnibus being returned as number 3, which is also deemed as one of the most relevant entities for the query. However, the score is still lower than the original term rank score of 0.89.

Table 28: Top 5 entity-centric result for *Q5*

| *Query:* "Harry Potter's third year at Hogwarts" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter and the Prisoner of Azkaban | Book |
| 2 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter | Book |
| 4 | Harry Potter (excerpt) | Book |
| 5 | Harry Potter and the Cursed Child: Parts One and Two | Book |

$$NDCG_5Q5 = \frac{DCG_5}{IDCG_5} = \frac{7.59}{10.17} = 0.75 \tag{21}$$

In contrast, the NDCG score for *Q6* is 0.79, calculated in equation 22, which is higher than the previous term rank score for this query. The result for query *Q6* is represented in table 29, where you can see that the top 5 consists of only books. With the reasoning that the enrichment considers books to be the most relevant entity for the query. This is appropriate for this particular query as the three most relevant entities are all books, and the entity-centric algorithm are retrieving all of them within the first five.

$$NDCG_5Q6 = \frac{DCG_5}{IDCG_5} = \frac{10.24}{12.9} = 0.79 \tag{22}$$

Table 29: Top 5 entity-centric result for *Q6*

| *Query:* "Jurassic Park aimed for young adult" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Jurassic Park | Book |
| 2 | Jurassic World Dominion: The Deluxe Junior Novelization | Book |
| 3 | Jurassic World: Special Edition Junior Novelization | Book |
| 4 | Horror: Filipino Fiction for Young Adults | Book |
| 5 | The Lost World: Jurassic Park: The Junior Novelization | Book |

Overall, the NDCG scores show that entity-centric enrichment performs worse for *Q5*, but significantly improves the result for *Q6*. This evaluation highlights the strengths and weaknesses of entity-centric enrichment for descriptive queries, as one can see that the enrichment fails to identify an appropriate target entity type for one query, while it is successful for the other.

**List Queries**

The last evaluation for the entity-centric enrichment is for query *Q7* and *Q8*, which are both list queries.

First up is query *Q7*, which is seeking the three Hunger Games books. As seen in the above section for the term rank evaluation, the term rank did not give a good ranking for this list query, as it does not retrieve every book of the series until rank 61. The entity-centric enrichment, however, significantly improves this by returning every book in the series within the top 30, shown in table 30.

Table 30: Entity-centric result for *Q7*

| *Query:* "The Hunger Games books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Book |
| 2 | The Hunger Games Tribute Guide | Book |
| 3 | The Hunger Games Trilogy (box set) | Book |
| 4 | The Hunger Games 4-Book Set | Book |
| 5 | The Hunger Games: Official Illustrated Movie Companion | Book |
| ... | ... | ... |
| 26 | Catching Fire | Book |
| 27 | Mockingjay | Book |

Similar to every previous query, the entity-centric ranking for this query only retrieves one type of entity within its top results, which in this case is book. This is another case where the targeted entity is highly appropriate for the query as the list query is only seeking book entities from the Hunger Games series, which explains why the result is improved from the original term rank.

The same can be seen for query *Q8*, which is only seeking book entities that are targeted by the enrichment. Here as well, the enrichment gives improved results. The result from *Q8* is presented in table 31.

Table 31: Entity-centric result for *Q8*

| *Query:* "Harry Potter books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter and the Philosopher's Stone | Book |
| 2 | Harry Potter and the Prisoner of Azkaban | Book |
| 3 | Harry Potter | Book |
| 4 | Harry Potter (excerpt) | Book |
| 5 | Harry Potter and the Cursed Child: Parts One and Two | Book |
| ... | ... | ... |
| 11 | Harry Potter and the Deathly Hallows | Book |
| 12 | Harry Potter and the Chamber of Secrets | Book |
| 13 | Harry Potter and the Half-Blood Prince | Book |
| ... | ... | ... |
| 15 | Harry Potter and the Goblet of Fire | Book |
| ... | ... | ... |
| 17 | Harry Potter and the Order of the Phoenix | Book |

Table 32: Precision and recall for *Q7* and *Q8*

| **Query** | **P@5** | **P@10** | **P@15** | **R@5** | **R@10** | **R@15** |
|---|---|---|---|---|---|---|
| *Q7* | 0.2 | 0.1 | 0.13 | 0.33 | 0.33 | 0.33 |
| *Q8* | 0.4 | 0.2 | 0.4 | 0.29 | 0.29 | 0.86 |

Overall, the two results and the precision and recall scores from table 32 show that the entity-centric enrichment is a good improvement from the original term rank, and shows the positives of having a targeted entity type for certain types of queries.

**Summary**

The evaluation of the entity-centric result above clearly shows that the enrichment has both strengths and weaknesses.

As described earlier, the objective of target entity is to identify target entity types and give these an extra elevation in the ranking. If the enrichment cannot find the appropriate target entity type, this can lead to a negative result for the ranking. This can be seen for the queries *Q1*, *Q2*, and *Q4*, where the target type from the enrichment does not match the real target type for the query, or the query simply does not have one specific target type.

On the other hand, there are positives of the enrichment when the targeted entities are appropriate for the queries. This is the case for the queries *Q6*, *Q7*, and *Q8*, which all give improved results compared to the original term ranking.

Figure 20 shows the NDCG results for each query and query type. Even though the descriptive queries seem to give the highest result, it is difficult to determine which query type the enrichment fits the most, as it varies from query to query whether the targeted entity types are appropriate or not. However, it is clear that the enrichment gives good results when identifying the correct target types for queries seeking a specific query type or queries where the most relevant entities are of this targeted type.

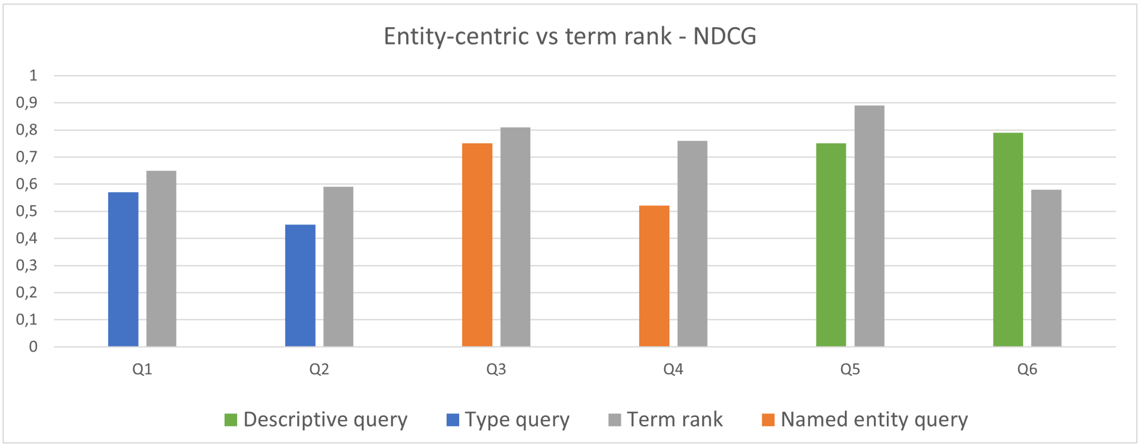


Figure 20: The NDCG scores for target entity: entity-centric

### 7.1.3 Target Entity - Type-centric

The other approach for target entity enrichment is called type-centric. Unlike the entity-centric technique that uses the most frequent types from the ranking, type-centric interprets the query terms to determine the target entity type.

#### Type Queries

To start the type queries *Q1* and *Q2* are evaluated. For *Q1*, the query ”*Author of Harry Potter*” is clearly searching for an author-entity. The type-centric enrichment automatically identifies a type-match in the query and therefore enhances the entities of this type further in the ranking.

The top 5 returned entities from *Q1* are shown in table 33. All returned entities are of the author-type. As described in equation 10, the weighting of this search is 20% original term rank and 80% of the type-centric algorithm. This leads to the top 5 results for this query effectively being the top 5 returned authors from the term rank. However, as the term rank of this system only retrieves entities that have term matches with the query, it fails to return the most relevant entity, *J.K. Rowling*. This shortcoming results in poor performance for the type-centric approach, as evidenced by the low NDCG@5 score calculated in equation 23.

Table 33: Top 5 type-centric result for *Q1*

| *Query:* "Author of Harry Potter" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Dennis Potter | Author |
| 2 | Robert Potter | Author |
| 3 | Deborah Potter | Author |
| 4 | Alexander Potter | Author |
| 5 | Stephen Potter | Author |

$$NDCG_5Q1 = \frac{DCG_5}{IDCG_5} = \frac{1.47}{9.29} = 0.16 \tag{23}$$

For *Q2*, the type-centric algorithm performs in a similar way as for *Q1*. It matches the term "awards" in the query to the entity type in the data set. All awards returned from the term rank get a higher score and the top 5 results turn out to be like in table 34. The four first awards were all rewarded to *The Hunger Games*-books. The results for this query were satisfactory, and in contrast to *Q1*, it managed to retrieve the entities with high relevance. Other entities that the survey subjects found partly relevant to this query were the books and authors of the series. This is why the NDCG@5 score did not reach all the way to the maximum of 1. The most relevant results for this query can be seen in table 11.

Table 34: Top 5 type-centric result for *Q2*

| *Query:* "The Hunger Games awards" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Grand Prix de l'Imaginaire (2010) | Award |
| 2 | Grand Prix de l'Imaginaire (2012) | Award |
| 3 | Golden Duck Award | Award |
| 4 | Locus Poll Award | Award |
| 5 | The Kitschies | Award |

$$NDCG_5Q2 = \frac{DCG_5}{IDCG_5} = \frac{11.86}{12.93} = 0.92 \tag{24}$$

The type-centric target entity approach works well for type queries, especially in this case where the query contains the name of the type that is sought or has a term match with the query. However, as shown in *Q1* the enrichment is dependent on the term rank retrieving the relevant entities in order to perform well. A solution for this problem could be to combine with another enrichment that retrieves extra relevant entities for the ranking. This will be further tested below in section 7.2.

### Named Entity Queries

For *Q3* and *Q4*, the type-centric algorithm is not able to interpret which type the query is searching for. This is due to the nature of the queries, which are looking for matches on the entity names. Neither of the queries has any terms that trigger the algorithm. This leads to the scores being an exact match of the term rank. In order for this approach to work for this kind of query, the user has to specify which type the named entity should be of, for example by adding "book" or "publication" at the end of the query.

Since these searches effectively are term rank, the results can be seen in table 15 for *Q3* and in table 16 for *Q4*, in chapter 7.1.1.

$$NDCG_5 Q3 = \frac{DCG_5}{IDCG_5} = \frac{10.87}{13.46} = 0.81 \tag{25}$$

$$NDCG_5 Q4 = \frac{DCG_5}{IDCG_5} = \frac{8.28}{10.84} = 0.76 \tag{26}$$

*Q3* for term rank got an NDCG@5 score of 0.81, which will also be the score for the type-centric metric. *Q4* got an NDCG@5 score of 0.76, which can be seen in equation 26.

### Descriptive Queries

Similar to the two named entity queries, the descriptive queries do not have any terms that trigger the type-centric algorithm. This means only the term rank would provide the entire score for *Q5* and *Q6*. The scores for the two descriptive queries are shown in equations 27 and 28, and are the same as the scores for the term rank.

$$NDCG_5 Q5 = \frac{DCG_5}{IDCG_5} = \frac{9.05}{10.17} = 0.89 \tag{27}$$

$$NDCG_5 Q6 = \frac{DCG_5}{IDCG_5} = \frac{7.45}{12.9} = 0.58 \tag{28}$$

### List Queries

Next, are the two list queries *Q7* and *Q8*. *Q7* is the first list query, which are searching for *The Hunger Games*-books. Table 35 presents the returned entities after the type-centric method has been applied. The results from *Q8*, which are searching for the *Harry Potter*-books, are presented in table 36. Both searches trigger the type-centric algorithm with the word "book", which leads to all book entities getting a higher score. The top 5 returned entities returned for both queries are all books, which shows the effects of this enrichment.

Table 35: Type-centric result for *Q7*

| *Query:* "The Hunger Games books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Book |
| 2 | The Hunger Games Tribute Guide | Book |
| 3 | The Hunger Games 4-Book Set | Book |
| 4 | The Hunger Games Trilogy (box set) | Book |
| 5 | The Hunger Games: Official Illustrated Movie Companion | Book |
| ... | ... | ... |
| 25 | Catching Fire | Book |
| 26 | Mockingjay | Book |

Table 36: Type-centric result for *Q8*

| *Query:* "Harry Potter books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter and the Philosopher's Stone | Book |
| 2 | Harry Potter and the Prisoner of Azkaban | Book |
| 3 | Harry Potter | Book |
| 4 | Harry Potter (excerpt) | Book |
| 5 | Harry Potter and the Cursed Child: Parts One and Two | Book |
| ... | ... | ... |
| 11 | Harry Potter and the Deathly Hallows | Book |
| 12 | Harry Potter and the Chamber of Secrets | Book |
| 13 | Harry Potter and the Half-Blood Prince | Book |
| ... | ... | ... |
| 15 | Harry Potter and the Goblet of Fire | Book |
| ... | ... | ... |
| 17 | Harry Potter and the Order of the Phoenix | Book |

The results after calculating precision and recall for the two list queries can be seen in table 37. For *Q7*, the results have significantly improved from the original term rank. All three books are ranked within the top 26, while the term rank does not return all books until rank 61.

For *Q8*, the performance of the implemented enrichment is similar to *Q7*, with an improvement from the original term rank. In this case, the last relevant book is found at rank 17, while the term rank does not return it until rank 26.

Table 37: Precision and recall for *Q7* and *Q8*

| **Query** | **P@5** | **P@10** | **P@15** | **R@5** | **R@10** | **R@15** |
|---|---|---|---|---|---|---|
| *Q7* | 0.2 | 0.1 | 0.07 | 0.33 | 0.33 | 0.33 |
| *Q8* | 0.4 | 0.2 | 0.4 | 0.29 | 0.29 | 0.86 |

Overall, the type-centric approach proves to be a valuable enrichment for list queries, as it successfully retrieves all relevant books before it is done in the term rank for both queries.

**Summary**

By analyzing the results from the type-centric algorithm, it becomes evident that it has both strengths and weaknesses. Since the objective of the metric is to find the intended entity type based on the terms in the query, the results were expected to vary from each query type. For the type queries, the results were mixed. Both *Q1* and *Q2* identified the correct target entity type. However, as the term rank had not succeeded to retrieve the most relevant entity, the enrichment did not perform well for *Q1*. This will, however, be explored further in chapter 7.2, where another enrichment will be added to the ranking to retrieve the relevant entities.

For the named entity queries and the descriptive queries, type-centric lacked the information needed to identify a target entity type. For this reason, the queries were solely dependent on the term rank in order to retrieve the most relevant entities, which was covered in chapter 7.1.1. In figure 21, the scores from NDCG@5 are visualized. The score for type queries is not perfect, which is explained by the poor result for *Q1*. Another thing to have in mind is that the query *Q1* is considered to only have one highly relevant entity, which means that even if the enrichment successfully ranked this entity first it might not get a great score as the 4 other entities in top 5 are potentially irrelevant.

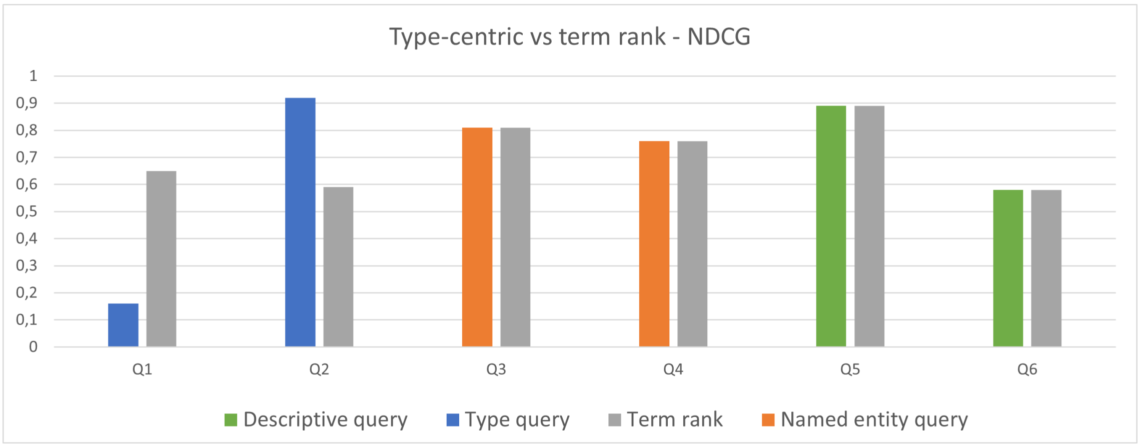


Figure 21: The NDCG scores for target entity: type-centric

For list queries, the performance of the type-centric enrichment is positive. It returns close to all relevant entities in the first 15, which is a significant improvement from the original term rank.

### 7.1.4 Tonon's Hybrid Rank

The next approach to be evaluated is Tonon's hybrid rank, which utilizes the graph structure of the ranked entities and leverages the connections between entities to enhance the overall ranking. To assess the effectiveness of this approach, the evaluation employed the same tests as those used for the target entity method.

#### Type Queries

First, let us start by examining the two type queries: *Q1* and *Q2*. As described in section 5.2, Tonon's hybrid rank retrieves the top-ranked entities from the original term ranking and leverages the connections they have to enhance the overall ranking. In other words, the objective of Tonon's hybrid rank is to enhance the ranking by awarding additional points to the neighbors of the top-ranked entities. For the evaluation, we select the top five entities from the term rank for both *Q1* and *Q2*, which are listed in table 38.

Table 38: Top 5 term rank result for *Q1* and *Q2*

| Query: "*Author of Harry Potter*" | | |
|---|---|---|
| 1 | Harry Potter | Series |
| 2 | Harry Potter and the Prisoner of Azkaban | Book |
| 3 | Harry Potter and the Philosopher's Stone | Book |
| 4 | British Library On Harry Potter | Series |
| 5 | Harry Potter | Book |

| Query: "*The Hunger Games books*" | | |
|---|---|---|
| 1 | The Hunger Games | Publication |
| 2 | The Hunger Games | Series |
| 3 | The Hunger Games | Book |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games Tribute Guide | Book |

Tonon's hybrid rank differs from previous enrichment methods in that it not only boosts the entities retrieved from the initial ranking but also adds more entities. For example, for *Q1*, the previous enrichment method only used the retrieved entities from the term rank, which does not include the author *J.K. Rowling*. However, the hybrid enrichment method retrieves *J.K. Rowling* because it is a neighbor of multiple entities in the top five results. This can be seen in table 39 where the top results for *Q1* are presented, and the *J.K. Rowling* entity is ranked as number 6.

Table 39: Top 6 Tonon's hybrid rank result for *Q1*

| *Query:* "Author of Harry Potter" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Series |
| 2 | Harry Potter | Publication |
| 3 | Harry Potter | Publication |
| 4 | Harry Potter | Publication |
| 5 | Harry Potter | Publication |
| 6 | J.K. Rowling | Author |

Table 40 presents similar results for *Q2*, where *Suzanne Collins*, the author of the Hunger Games series, appears as one of the top-ranked entities despite not appearing in the ranking before.

Table 40: Top 5 Tonon's hybrid rank result for *Q2*

| *Query:* "The Hunger Games awards" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Suzanne Collins | Author |
| 2 | The Hunger Games | Book |
| 3 | The Hunger Games | Book |
| 4 | The Hunger Games | Series |
| 5 | The Hunger Games | Publication |

Furthermore, the NDCG is calculated to further evaluate and analyze the results. Equations 29 and 30 show the calculations for *Q1* and *Q2*, with *Q1* receiving a score of 0.70 and *Q2* receiving 0.49. When compared to the original term rank and its NDCG score for the two queries, one can see that the enrichment method achieves a higher score for *Q1* but a worse result for *Q2*.

$$NDCG_5Q1 = \frac{DCG_5}{IDCG_5} = \frac{6.52}{9.29} = 0.70 \tag{29}$$

$$NDCG_5Q2 = \frac{DCG_5}{IDCG_5} = \frac{5.91}{12.93} = 0.46 \tag{30}$$

Based solely on the scores, it is impossible to determine whether Tonon's hybrid rank improves the ranking or not, as one score increases while the other decreases. However, based on observations, one can see that the enrichment method provides an advantage as it retrieves additional entities that were not previously ranked. This is particularly evident in *Q1*, where the most relevant entity is now being retrieved and ranked for the first time.

### Named Entity Queries

The next type of queries to be evaluated is the named entity queries: *Q3* and *Q4*. Similar to the last two queries, Tonon's hybrid enrichment utilizes the top 5 entities from the term rank and awards the neighbors of these. The top 5 results from term rank can be seen above in the term rank evaluation in table 15 and table 16.

For *Q3* which is searching with the name of the first Harry Potter book, the term rank includes the *Harry Potter* series, a publication, and various Harry Potter books within its top 5 results. Tonon's hybrid rank then utilizes these entities to create a new ranking, the result of this is shown in table 41. Here one can see that the enrichment once again, similar to the result of *Q1*, returns the author of the *Harry Potter* series, *J.K. Rowling.* In addition to the author, it also returns the series, the first book, and a couple of publications.

Table 41: Top 5 Tonon's hybrid rank result for *Q3*

| *Query:* "Harry Potter and the Philosopher's Stone" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Series |
| 2 | J. K. Rowling | Author |
| 3 | Harry Potter and the Philosopher's Stone | Book |
| 4 | Harry Potter | Publication |
| 5 | Harry Potter and the Philosopher's Stone | Publication |

Furthermore, the outcome can be assessed by comparing it with the top relevant entities for the query displayed in table 13. From a quick observation, one can see that the enrichment produces relevant results, as the top two relevant entities which are the first Harry Potter book and its publication are also ranked highly by the enrichment. In addition to that the series and *J.K Rowling* which is ranked first and second, are also deemed relevant. These observations can be confirmed by the NDCG score, which also indicates a good outcome for this query. The calculations are presented below in equation 31.

$$NDCG_5Q3 = \frac{DCG_5}{IDCG_5} = \frac{11.13}{13.46} = 0.84 \tag{31}$$

Next, the outcome for query *Q4* is examined. The top 5 results from Tonon's hybrid enrichment are displayed in table 42. The enrichment returns *Kerily Sapet*, who is the author of Suzanne Collins's biography, *the Hunger Games* publication, and the series' three books. This aligns with the objective of the enrichment as they all have some connections to the person named in the query, *Suzanne Collins.* However, the enrichment performs badly when considering the relevant entities from the survey and calculating the NDCG results. The NDCG calculation yields a score as low as 0.43, as shown in equation 32. This is because, when examining the relevant entities from the query in table 14, it is apparent that the survey participants deem the author and the author's biography as the most relevant. Although the author's books are connected to the author, they are not deemed as relevant.

Table 42: Top 5 Tonon's hybrid rank result for *Q4*

| *Query:* "Suzanne Collins" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Kerily Sapet | Author |
| 2 | The Hunger Games | Publication |
| 3 | Catching Fire | Book |
| 4 | Mockingjay | Book |
| 5 | The Hunger Games | Book |

$$NDCG_5Q4 = \frac{DCG_5}{IDCG_5} = \frac{4.62}{10.84} = 0.43 \tag{32}$$

Similar to the evaluation of the type queries, the results for these queries exhibit a mixed performance, where one query outperforms the term rank while the other falls short. In the case of *Q4*, the enrichment retrieves three entities that were not previously ranked, but were considered moderately relevant by the survey participants with a relevance score above 2.3. While this indicates that the enrichment has done a good job in retrieving more relevant entities, it falls short because these entities are given too much weight in the ranking. This can be easily fixed by adjusting the weighting between the term rank and the enrichment, which are currently at 0.2 and 0.8, respectively. Changing this to a different weighting where the term rank is given higher weight could have produced a different result, where the author and their biography ranked highly by the term rank are weighted higher than their neighbors.

### Descriptive Queries

Next up is queries *Q5* and *Q6*, which are descriptive queries. The enrichment result for *Q5* is presented in table 43 and the result for *Q6* is presented in table 44.

The query "*Harry Potter's third year at Hogwarts*" in *Q5* describes the third book in the *Harry Potter* series, and the enrichment performs well by returning the third book and its publications, as well as the series and its author within the top 5 results.

Similarly, *Q6* describes the junior novelization of the Jurassic Park books as being aimed at young adults. The enrichment returns two such junior novelizations, as well as the *Jurassic Park* series, its author, and the first book. This can be considered a good result.

Table 43: Top 5 Tonon's hybrid rank result for *Q5*

| *Query:* "Harry Potter's third year at Hogwarts" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | J. K. Rowling | Author |
| 2 | Hogwarts Library | Book |
| 3 | Harry Potter | Series |
| 4 | Harry Potter and the Prisoner of Azkaban | Book |
| 5 | Harry Potter and the Prisoner of Azkaban | Publication |

Table 44: Top 5 Tonon's hybrid rank result for *Q6*

| *Query:* "Jurassic Park aimed for young adult" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Jurassic Park | Series |
| 2 | Michael Crichton | Author |
| 3 | Jurassic Park | Book |
| 4 | Jurassic World Dominion: The Deluxe Junior Novelization | Book |
| 5 | Jurassic World: Special Edition Junior Novelization | Book |

Overall, the results from Tonon's hybrid enrichment for both queries, *Q5* and *Q6* shows good result. The top 5 results for each query include relevant entities such as the book titles, the series, and the authors, indicating that the enrichment is able to capture the main themes and contexts of the queries. In fact, a comparison of the retrieved entities with the most relevant ones from the survey confirms this observation. Moreover, the NDCG@5 scores for both queries are high (see equations 33 and 34). However, there is still an instance where the enrichment performs worse than the initial ranking score. As previously mentioned in the evaluation of named entity queries, the enrichment is able to retrieve relevant entities that were not captured by the initial rank. In these cases, it is a matter of adjusting the weight of the metrics used to rank the results in order to obtain a better score, while still benefiting from the improved relevance provided by the enrichment.

$$NDCG_5 Q5 = \frac{DCG_5}{IDCG_5} = \frac{8.40}{10.66} = 0.79 \tag{33}$$

$$NDCG_5 Q6 = \frac{DCG_5}{IDCG_5} = \frac{11.26}{12.9} = 0.87 \tag{34}$$

### List Queries

The last two queries are the list queries: *Q7* and *Q8*. These two queries did not have a good result from the initial term rank, as they did not manage to retrieve all the relevant entities within its top result. For *Q7*, all the Hunger Games books were not retrieved until rank 61, while for *Q8* the last Harry Potter book was retrieved at rank 26.

The result for Tonon's hybrid enrichment is presented in table 45 and table 46. And here one can see that the enrichment gives a mixed performance. For query *Q7* the enrichment shows promising results, as it now retrieves all three Hunger Games books within rank 15. For the other query *Q8*, however, the last Harry Potter book is not retrieved until rank 45, which is a worse performance than the initial term rank. With this query, it is shown that the enrichment considers the series and various Harry Potter publications to be most relevant instead of the desired books.

Table 45: Tonon's hybrid rank result for *Q7*

| *Query:* "The Hunger Games books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Suzanne Collins | Author |
| 3 | The Hunger Games | Book |
| 4 | The Hunger Games | Series |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games Tribute Guide | Book |
| ... | ... | ... |
| 14 | Catching Fire | Book |
| 15 | Mockingjay | Book |

Table 46: Tonon's hybrid rank result for *Q8*

| *Query:* "Harry Potter books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Series |
| 2 | Harry Potter | Publication |
| 3 | Harry Potter | Publication |
| 4 | Harry Potter | Publication |
| 5 | Harry Potter | Publication |
| ... | ... | ... |
| 30 | Harry Potter and the Philosopher's Stone | Book |
| 31 | Harry Potter and the Prisoner of Azkaban | Book |
| ... | ... | ... |
| 39 | Harry Potter and the Deathly Hallows | Book |
| 40 | Harry Potter and the Chamber of Secrets | Book |
| 41 | Harry Potter and the Half-Blood Prince | Book |
| ... | ... | ... |
| 43 | Harry Potter and the Goblet of Fire | Book |
| ... | ... | ... |
| 45 | Harry Potter and the Order of the Phoenix | Book |

These results give the precision and recall presented in table 47, which further shows that the enrichment gives enhanced results for query *Q7* and worse the result for query *Q8*.

Table 47: Precision and recall for *Q7* and *Q8*

| **Query** | **P@5** | **P@10** | **P@15** | **R@5** | **R@10** | **R@15** |
|---|---|---|---|---|---|---|
| *Q7* | 0.2 | 0.1 | 0.2 | 0.33 | 0.33 | 1 |
| *Q8* | 0 | 0 | 0 | 0 | 0 | 0 |

Based on these results it is impossible to determine whether Tonon's hybrid enrichment is a good fit for list queries or not. For one query, *Q7*, it perceives the desired book as relevant as they are neighbors to the series and publication which are ranked highest in the term rank. On the other query, *Q8*, It also perceives the desired books as relevant as it is neighbor to the Harry Potter series. However, in this case, it perceives the various publications as even more relevant as they have more connections to the top-ranked entities from the term rank.

### Summary

Tonon's hybrid rank enrichment gives both positive and negative outcomes compared to the original term rank. However, as mentioned in the evaluation above, despite the enrichment sometimes giving worse precision scores it still provides valuable enrichment as it retrieves more relevant entities to the ranking. This suggests that despite its low result, the technique can still be useful in improving the search result, it is just a matter of adjusting the weighing of the enrichment compared to the term rank.

In summary, the evaluation indicates that Tonon's hybrid rank enrichment technique is a valuable tool for improving search results, and further research is needed to optimize its effectiveness. It is not advisable to weigh the enrichment higher than the initial term rank, as doing so caused the ranking to consider the neighbors of the top-ranked entities more highly than the entities themselves. This was observed in queries *Q4*, *Q5*, and *Q8*. An adjustment of the weighting is therefore tested and evaluated further, which can be found below in chapter 7.2.

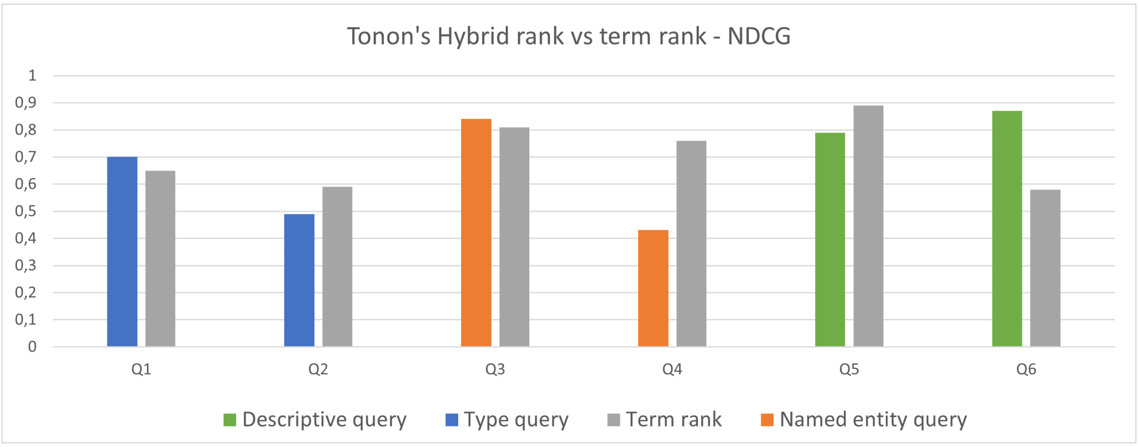


Figure 22: The NDCG scores for Tonon's hybrid rank

### 7.1.5 Entity Linking

Next up on the evaluation is the entity linking enrichment. Entity linking is similar to Tonon's hybrid rank as both utilize the graph structure of the dataset for the enriched ranking. However, the objective of entity linking in this thesis is to identify and leverage entity mentions in queries. Entity linking finds and retrieves all entities with a match of any word combination in the query, as well as the connections to these entities. This leads to new entities being introduced in the rankings, which might not be picked up by the term rank.

### Type Queries

The first type of query to evaluate is the two type queries, *Q1* and *Q2*. Similar to Tonon's hybrid enrichment, entity linking is another approach that can retrieve extra entities to the final result set. It creates the possibility for entities with a score of 0 in the term rank to still be ranked.

Since the objective of entity linking is to find and use entity mentions in the query to improve the ranking, one can start to look at which mentions are identified. For *Q1* there are identified

24 linked entities, which come from matches on the terms: "*author*", "*harry*" and "*harry potter*". This means that the system does not only link entities with the name *Harry Potter*, but it also links entities named *author* and *Harry*, which might be irrelevant to the query, and *Harry Potter* series. Still the result from *Q1* with the entity linking enrichment shows to give relevant result, as every entity returned within top 5 results are deemed relevant to the query, these are shown in table 48. This is due to the fact that the match on *Harry Potter* outweighs the two others, by having more linked entities.

In addition to the linked entities, the enrichment also looks at their connections. For this particular query, the enrichment finds 94 connecting entities and boosts these entities based on the number of connections they have to the linked entities. This becomes evident in the result for *Q1*, as the most relevant entity, *J.K. Rowling*, is returned. This entity is returned due to it being connected to multiple entities that are linked to the terms "*Harry Potter*".

Table 48: Top 5 entity linking result for *Q1*

| *Query:* "Author of Harry Potter" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Book |
| 2 | J. K. Rowling | Author |
| 3 | Harry Potter | Publication |
| 4 | Bloomsbury | Publisher |
| 5 | Harry Potter and the Prisoner of Azkaban | Book |

Entity linking proves to give good results for *Q1*, this is also shown by the high NDCG score calculated in equation 35.

$$NDCG_5 Q1 = \frac{DCG_5}{IDCG_5} = \frac{7.42}{9.29} = 0.80 \tag{35}$$

However, the enrichment does not perform as well for the second type query *Q2*. For this query, the enrichment has identified linked entities on the terms: "*the*", "*hunger*", "*games*", "*awards*", "*the hunger*", "*hunger games*" and "*the hunger games*". With these matches, the enrichment identifies 311 different linked entities, which clearly shows the disadvantage of having such a large and diverse dataset. Most of the term matches are not related to *The Hunger Games* series which the query is asking about, instead, they are related to random other entities which share the same terms as the series. Luckily there is another case where most of the linked entities are coming from the relevant term match, *the Hunger Games*, which makes most of the top-ranked entities relevant. The top 5 results set from the enrichment are presented in table 49. Here one can see that four out of five entities are related to *the Hunger Games* series, where 3 of them are identified as linked entities. However, this similar to term rank fails to rank the awards of the series, which is what the query is seeking. This is emphasized by a mediocre NDCG@5 score of 0.53 (see equation 36).

Table 49: Top 5 entity linking result for *Q2*

| *Query:* "The Hunger Games awards" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Book |
| 2 | Suzanne Collins | Author |
| 3 | The Hunger | Book |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games | Series |

$$NDCG_5 Q2 = \frac{DCG_5}{IDCG_5} = \frac{6.48}{12.93} = 0.50 \tag{36}$$

To summarize entity linking for type queries, the results are deviating. Only one of the queries shows improved results from the term rank. For *Q2* does neither the enrichment nor the initial term rank identify the top relevant result, which are the awards for the series. This is due to the books and author entity having more total connections than the award entities.

### Named Entity Queries

The next query type to be evaluated is named entity queries. These two queries, *Q3* and *Q4*, are exact term matches of entities in the data set. Because of this, the enrichment is expected to perform well for these queries. The enrichment should identify linked entities with the exact name as the query for the two queries, and boost them as well as their connecting neighbors.

However, for *Q3*, the query does not perform as well as expected. In addition to identifying matching entities on the whole query, it also identifies partial matches on the query like: *"harry"*, *"and"*, *"the"*, *"philosopher"*, *"stone"*, *"harry potter"* and *"the philosopher"*, resulting in as many as 71 linked entities. Due to this, the focus is skewed from being on the first Harry Potter book to being on every linked entity and their connections, resulting in the result set viewed in table 50. This causes the enrichment to perform worse than the original term rank, with now an NDCG score of only 0.61 (see equation 37).

Table 50: Top 5 entity linking result for *Q3*

| *Query:* "Harry Potter and the Philosopher's Stone" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Book |
| 2 | J. K. Rowling | Author |
| 3 | Harry Potter | Publication |
| 4 | Harry Potter and the Philosopher's Stone | Book |
| 5 | Bloomsbury | Publisher |

$$NDCG_5Q3 = \frac{DCG_5}{IDCG_5} = \frac{8.23}{13.46} = 0.61 \tag{37}$$

For *Q4*, the results of the entity linking are more as expected and are presented in table 51. The data set contains three entities that are an exact name match of the query, which are all returned top 3. The last two entities returned are from a partial query match and from a connecting entity. *Kerrily Sapet* is retrieved as a connecting entity, as it is the author of Suzanne Collins biography, however, it is not deemed relevant according to the user survey. Still, the result is deemed positive, as the NDCG@5 score is marginally better than the term rank's score (see equation 38). The reason the enrichment proves to be more valuable for this query is due to the dataset only containing 8 entities that are considered a match for the entity linking enrichment. This smaller pool of linked entities, in comparison to the 71 linked entities in *Q3*, makes it simpler to enhance the correct entities in the ranking.

Table 51: Top 5 entity linking result for *Q4*

| *Query:* "Suzanne Collins" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Suzanne Collins | Book |
| 2 | Suzanne Collins | Publication |
| 3 | Suzanne Collins | Author |
| 4 | Suzanne | Book |
| 5 | Kerrily Sapet | Author |

$$NDCG_5Q4 = \frac{DCG_5}{IDCG_5} = \frac{8.31}{10.84} = 0.77 \tag{38}$$

Entity linking shows both strengths and weaknesses for named entity queries. The main takeaway from these two queries is that entity linking works well to find connected entities to the named entities, but that the scoring may fail to rank the most relevant entities first. Both searches manage to retrieve the most relevant entity, but the enrichment fails in the task to rank them first for query *Q3* as it has too many irrelevant linked entities.

### Descriptive Queries

The two next queries, *Q5* and *Q6*, are descriptive queries. The results from these queries are presented in tables 52 and 53. These queries are looking for term matches in the metadata of the entities, like *Q5* which is searching for the third year Harry Potter went to Hogwarts. *Q6* is searching for the junior editions of the Jurassic Park books.

Table 52: Top 5 entity linking result for *Q5*

| *Query:* "Harry Potter's third year at Hogwarts" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Book |
| 2 | J. K. Rowling | Author |
| 3 | Harry Potter and the Order of the Phoenix | Book |
| 4 | Harry Potter and the Prisoner of Azkaban | Book |
| 5 | Harry Potter | Publication |

Table 53: Top 5 entity linking result for *Q6*

| *Query:* "Jurassic Park aimed for young adult" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Jurassic Park | Book |
| 2 | Michael Crichton | Author |
| 3 | Jurassic Park | Publication |
| 4 | Ballantine Books | Publisher |
| 5 | Jurassic Park | Publication |

Overall the results from entity linking for descriptive queries work decently. One of the main issues with this enrichment is that the linked entities and connected entities that are connected to multiple linked entities get a high score and outweigh the initial term rank. Whereas, for this particular query type, the term rank is more efficient at ranking the most relevant entities. This is because descriptive queries aim to identify entities that match a particular description, and these descriptions are often found in the metadata of the entities. As entity linking enrichment is primarily based on the name of the entities, it can be challenging to identify the most suitable relevant entities for descriptive queries.

For *Q5*, all the results in the top 5 rankings are relevant, but the most relevant entity is ranked 4th which leads to a mediocre NDCG@5 score of 0.74 (see equation 39). *Q6* does not return any of the most relevant entities, but since all entities returned are connected to the highest relevance entities, and therefore are partly relevant, the score is better than expected. The calculation for the NDCG@5 score for *Q6* is presented in equation 40.

$$NDCG_5Q5 = \frac{DCG_5}{IDCG_5} = \frac{7.52}{10.17} = 0.74 \tag{39}$$

$$NDCG_5Q6 = \frac{DCG_5}{IDCG_5} = \frac{8.13}{12.90} = 0.63 \tag{40}$$

Even though the NDCG@5 scores for the descriptive queries are not impressive, and are collectively outperformed by term rank, there are still positive effects to highlight. The algorithm manages to retrieve many relevant entities the term rank is unable to collect, due to the lack of term matches in the entities.

### List Queries

The last query type to evaluate the entity linking enrichment against is list queries. These queries did not perform well in the initial term rank. The results from the two searches can be seen in tables 54 and 55. The results had deviating results between the two queries, where *Q7* performed poorly and *Q8* had a better performance.

*Q7* had a poor performance whereas the first book of the series was ranked first, but the two others did not appear before the last retrieved entities in rank 995. This shares similarity with the second type query, *Q2*, as the enrichment retrieves a large number of linked entities. For *Q7* the enrichment retrieves as many as 317 linked entities, which explains why the two last books in the series lose their importance when they do not have a direct term match from the query.

For *Q8*, the results were better. All books were retrieved within the top 11 entities. With this query, the enrichment identified 32 linked entities, where the books in the Harry Potter series were connected to most of them. Due to this, the books got a good score, making them rank high in the resulting search result.

Table 54: Entity linking result for *Q7*

| *Query:* ”The Hunger Games books” | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Book |
| 2 | Suzanne Collins | Author |
| 3 | The Hunger | Book |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games | Series |
| ... | ... | ... |
| 994 | Catching Fire | Book |
| 995 | Mockingjay | Book |

Table 55: Entity linking result for *Q8*

| *Query:* ”Harry Potter books” | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Book |
| 2 | J. K. Rowling | Author |
| 3 | Harry Potter | Publication |
| 4 | Bloomsbury | Publisher |
| 5 | Harry Potter and the Philosopher’s Stone | Book |
| 6 | Harry Potter and the Prisoner of Azkaban | Book |
| 7 | Harry Potter and the Deathly Hallows | Book |
| 8 | Harry Potter and the Chamber of Secrets | Book |
| 9 | Harry Potter and the Half-Blood Prince | Book |
| 10 | Harry Potter and the Goblet of Fire | Book |
| 11 | Harry Potter and the Order of the Phoenix | Book |

Table 56: Precision and recall for *Q7* and *Q8*

| **Query** | **P@5** | **P@10** | **P@15** | **R@5** | **R@10** | **R@15** |
|---|---|---|---|---|---|---|
| *Q7* | 0.2 | 0.1 | 0.07 | 0.33 | 0.33 | 0.33 |
| *Q8* | 0.2 | 0.6 | 0.47 | 0.14 | 0.86 | 1 |

As shown in table 56, entity linking for list queries had inconsistent outcomes. It further proves that entity linking enrichment highly depends on the query formulation and the dataset used. The queries which generate a lot of linked entities tend to perform worse than the queries with few linked entities.

### Summary

The entity linking algorithm provided both positive and negative results compared to term rank. In cases where the performance is worse than for term rank, there are positive effects to consider. For instance that the entity linking enrichment retrieves more relevant entities, which might not be retrieved by the original term ranking.

In some cases, the linked entities and their connections receive a too high score, so entities with a high term rank score get a lower total score. This could be solved by altering the weighting of the scores, which could lead to a more well-balanced result set. Entity linking could also be combined with other enrichments in order to get a better ranking.

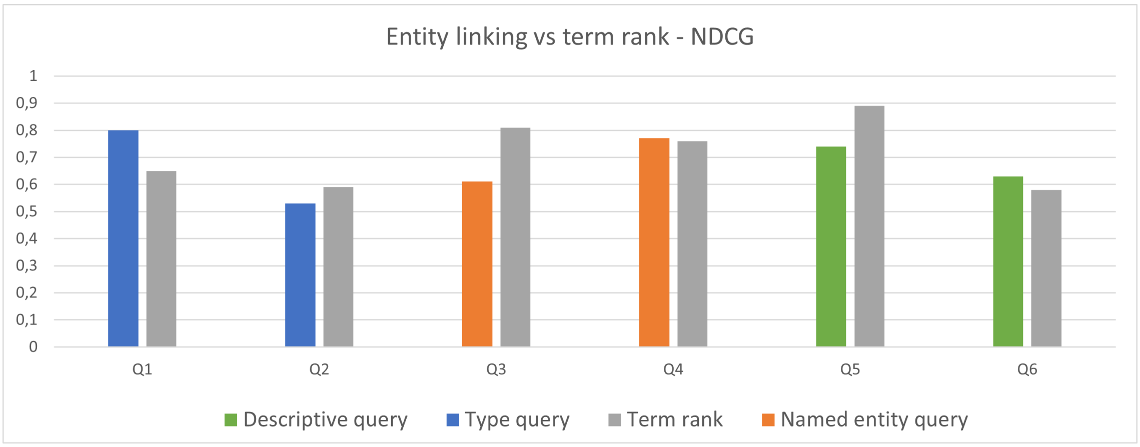


Figure 23: The NDCG scores for entity linking

### 7.1.6 Centrality - PageRank

The last enrichment implemented is the two types of centrality, which both are evaluated separately.

PageRank is the most popular algorithm for calculating centrality, which is a centrality score that accounts for the number of links a node has as well as the quality of these links.

### Type Queries

First up is the evaluation of the two type queries. The objective of PageRank is to find the entities with the highest amount of links and the links with the highest quality and rank them thereafter. The results were expected to be fairly similar to term rank since most of the entities retrieved in the term rank search have a similar amount of links and are linked to the same entities. The top 5 results are an exact match for term rank and PageRank for both *Q1* and *Q2*, and the two results can be seen in figure 18 and table 12.

$$NDCG_5Q1 = \frac{DCG_5}{IDCG_5} = \frac{5.99}{9.29} = 0.65 \tag{41}$$

$$NDCG_5Q2 = \frac{DCG_5}{IDCG_5} = \frac{7.16}{12.93} = 0.55 \tag{42}$$

Upon analyzing the PageRank scores, it became apparent that most entities had significantly low scores. This is because the entity with the highest score, which is *uncredited* (assigned as the author entity for every uncredited book), has considerably more connections than the next highest entity. Consequently, all other entities receive substantially lower scores, which results in only marginal effects on the search results.

### Named Entity Queries

As for type queries, named entity queries did not differ a lot compared to the term rank. For *Q3*, the results varied some, but 4 of the 5 entities exist on both lists, see top 5 results in table 57. The two first entities are similar. Since one of the most relevant entities is placed further down on the list, the NDCG@5 score for PageRank is worse for *Q3*, compared to term rank.

Table 57: Top 5 PageRank result for *Q3*

| *Query:* "Harry Potter and the Philosopher's Stone" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Series |
| 2 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter and the Prisoner of Azkaban | Book |
| 4 | Harry Potter and the Philosopher's Stone | Publication |
| 5 | Harry Potter and the Cursed Child: Parts One and Two | Book |

$$NDCG_5Q3 = \frac{DCG_5}{IDCG_5} = \frac{9.69}{13.46} = 0.72 \tag{43}$$

For *Q4*, the results are precisely the same as for term rank. The only entity with a relatively high PageRank score in the top 5 list is the *Suzanne Collins* author-entity, while the rest is mostly dependent on the term rank score in order to place high on the ranking.

$$NDCG_5Q4 = \frac{DCG_5}{IDCG_5} = \frac{8.28}{10.84} = 0.76 \tag{44}$$

In total, the named entity queries did not vary much from the term rank. *Q3*, which was the one query that provided different results compared to term rank, had a worse score.

### Descriptive Queries

PageRank continues to have little impact, and the enrichment does not affect the result for any of the two descriptive queries. For *Q5*, the only entity that has a relatively high PageRank score is the *Harry Potter* series. This was expected since this entity is connected to all Harry Potter books and publications, as well as the author and publisher. Due to every entity being similar to the term rank, the NDCG@5 score is also the same. The score for *Q5* and *Q6* can be seen in equations 45 and 46.

$$NDCG_5Q5 = \frac{DCG_5}{IDCG_5} = \frac{9.05}{10.17} = 0.89 \tag{45}$$

$$NDCG_5Q6 = \frac{DCG_5}{IDCG_5} = \frac{7.45}{12.90} = 0.58 \tag{46}$$

**List Queries**

The same problems as for the previous queries exist for list queries as well. The PageRank score for the returned entities is too small to make a big impact on the ranking. The results for *Q7* and *Q8* can be seen in tables 58 and 59.

For *Q7*, the only difference from term rank is that the last book of the series is ranked two places better.

Table 58: PageRank result for *Q7*

| *Query:* "The Hunger Games books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Publication |
| 2 | The Hunger Games | Series |
| 3 | The Hunger Games | Book |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games Tribute Guide | Book |
| ... | ... | ... |
| 59 | Mockingjay | Book |
| 60 | Catching Fire | Book |

The PageRank scores for *Q8* are marginally better than for term rank. Neither of the relevant entities manages to get into the top 15 ranking, except the two that are ranked similarly to the term rank results.

Table 59: PageRank result for *Q8*

| *Query:* "Harry Potter books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Harry Potter | Series |
| 2 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter and the Prisoner of Azkaban | Book |
| 4 | Harry Potter | Book |
| 5 | British Library on Harry Potter | Series |
| ... | ... | ... |
| 16 | Harry Potter and the Deathly Hallows | Book |
| ... | ... | ... |
| 18 | Harry Potter and the Chamber of Secrets | Book |
| 19 | Harry Potter and the Goblet of Fire | Book |
| 20 | Harry Potter and the Half-Blood Prince | Book |
| ... | ... | ... |
| 26 | Harry Potter and the Order of the Phoenix | Book |

As seen in table 60, the PageRank has similar precision and recall scores as for term rank, with a marginal improvement.

Table 60: Precision and recall for *Q7* and *Q8*

| **Query** | **P@5** | **P@10** | **P@15** | **R@5** | **R@10** | **R@15** |
|---|---|---|---|---|---|---|
| *Q7* | 0.2 | 0.1 | 0.07 | 0.33 | 0.33 | 0.33 |
| *Q8* | 0.4 | 0.2 | 0.13 | 0.29 | 0.29 | 0.29 |

### Summary

The PageRank results were in total underwhelming. Some of the problems faced with this enrichment are that it is a global score, and a few entities get a high score while it ruins the score for the rest of the entities. This leads to the PageRank score having less impact with the chosen evaluation formula of 20% term rank score and 80% enrichment score. Another reason the PageRank has little impact on the result set could be that the entities have similar connections, which makes the PageRank scores indistinguishable. A comparison with the original term rank can be seen in figure 24.

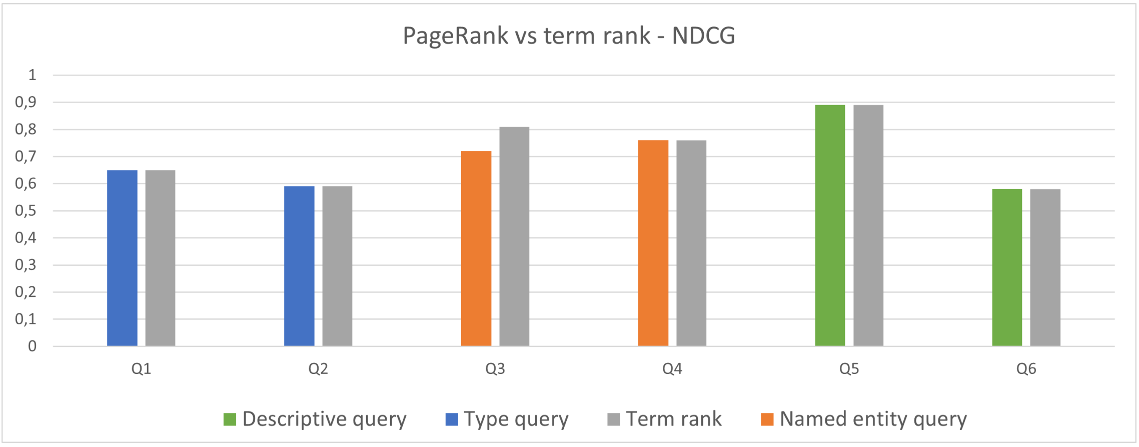


Figure 24: The NDCG scores for PageRank

### 7.1.7 Centrality - Local Centrality

As it was found that the global PageRank score may not be the best fit for providing relevant results to a query, a local centrality score was implemented as an alternative. This score is calculated at the time of the query and is a centrality score of the subgraph formed by the returned entities.

#### Type Queries

To evaluate the local centrality score, the two type queries are evaluated first: *Q1* and *Q2*. The aim of the local centrality score is to calculate a centrality score for the subgraph that is created by the entities returned by the initial term rank. The purpose of this score is to boost the entities with a high centrality score in the ranking. This means that the entities with many connecting neighbors in the subgraph should be given a higher ranking to distinguish them from the entities without relevant neighbors. Furthermore, the local centrality score has a similar function to Tonon's hybrid rank and entity linking, as it can add additional entities to the ranking that were not present in the initial term rank.

The objective of query *Q1* is to retrieve *J.K. Rowling* as the author of Harry Potter, but the term rank evaluation demonstrated that the term rank is insufficient to identify this entity as it has no common term with the query. On the other hand, the local centrality method generates a subgraph by taking every entity in the original term rank and creating connections with their closest neighbors. As a result, the subgraph contains more entities than the initially returned entities, which allows for the addition of more relevant entities. For example, in the case of query *Q1*, the enrichment includes the entity *J.K. Rowling* in the subgraph, as this entity is connected to multiple previously retrieved entities. Moreover, since the author entity is connected to numerous other entities in the subgraph, it also receives a high score in the resulting ranking. This is demonstrated in Table 61, which shows the final top 5 results.

Table 61: Top 5 local centrality result for *Q1*

| *Query:* "Author of Harry Potter" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | J. K. Rowling | Author |
| 2 | Harry Potter and the Prisoner of Azkaban | Book |
| 3 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter | Series |
| 4 | Harry Potter and the Chamber of Secrets | Book |

The results indicate that the enrichment has significantly improved the ranking from the original term rank by returning many of the most relevant entities for the query. This is also demonstrated by the NDCG score, which is as high as 0.98, representing a substantial increase from the term rank score of 0.65. The calculation for this score is illustrated in equation 47.

$$NDCG_5 Q1 = \frac{DCG_5}{IDCG_5} = \frac{9.09}{9.29} = 0.98 \tag{47}$$

However, for query *Q2*, the enrichment appears to be less effective. The result for this query is presented in Table 62, and the resulting NDCG score is only 0.49 (see equation 48), which is considerably worse than the previous query. As mentioned previously, query *Q2* is looking for award entities related to *the Hunger Games* series, which neither the term rank nor the enrichment has identified. Similar to the previous query *Q1*, the enrichment instead focuses on the author and various books in the series, which is not deemed as relevant for this particular query.

Table 62: Top 5 local centrality result for *Q2*

| *Query:* "The Hunger Games awards" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Suzanne Collins | Author |
| 2 | The Hunger Games | Book |
| 3 | Nebula Awards | Series |
| 4 | Mockingjay | Book |
| 5 | Catching Fire | Book |

$$NDCG_5 Q2 = \frac{DCG_5}{IDCG_5} = \frac{6.39}{12.93} = 0.49 \tag{48}$$

Overall, the local centrality enrichment score was found to be highly effective for one of the type queries but produced a slightly worse result for the other query.

### Named Entity Queries

Moving on, the next type of query to evaluate is named entity queries. These are two queries that contain the name of the entity that is sought. However, the local centrality enrichment did not prove to be effective for these queries as exact term matches were considered more important than the centrality of entities. The results of the enrichment for these queries can be seen in Table 63 and Table 64, where it is evident that the most relevant entities were ranked lower than the original term rank.

Table 63: Top 5 local centrality result for *Q3*

| *Query:* "Harry Potter and the Philosopher's Stone" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | J. K. Rowling | Author |
| 2 | Harry Potter and the Philosopher's Stone | Book |
| 3 | Harry Potter and the Prisoner of Azkaban | Book |
| 4 | Rex Stone | Author |
| 5 | Harry Potter and the Chamber of Secrets | Book |

Table 64: Top 5 local centrality result for *Q4*

| *Query:* "Suzanne Collins" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Paul Collins | Author |
| 2 | Frederic Collins | Author |
| 3 | Nancy A. Collins | Author |
| 4 | Suzanne Collins | Author |
| 5 | Wilkie Collins | Author |

The NDCG scores for these queries were 0.57 for *Q3* (see equation 49) and 0.27 for *Q4* (see equation 50), which further demonstrates that the local centrality enrichment is not appropriate for the named entity queries. It is reasonable to assume that the term rank, which solely focuses on term matches, would be more sufficient for providing an appropriate result.

$$NDCG_5 Q3 = \frac{DCG_5}{IDCG_5} = \frac{7.72}{13.46} = 0.57 \tag{49}$$

$$NDCG_5 Q4 = \frac{DCG_5}{IDCG_5} = \frac{2.88}{10.84} = 0.27 \tag{50}$$

**Descriptive Queries**

Next is the evaluation of the enrichment for the descriptive queries. Similar to named entity queries this is another type of query one can assume that term matches between the query and entities play a big part. This is also shown earlier in the evaluation of the term rank, which receives two scores 0.89 and 0.58 for the two descriptive queries. Because of this, the enrichments once again does not give effective result in improving the ranking.

For query *Q5*, the enrichment does not provide a terrible result. The top 5 results for this query are presented in table 65. The enrichment returns the author of the *Harry Potter* series first in the ranking, as that is the entity that is connected to most of the entities returned in the initial ranking. In addition to that the enrichment return several of the Harry Potter books, where the most relevant entity is returned as rank 2. As mentioned this result is not considered too bad, but it is still below the original term rank which returns more entities that are considered more relevant. This is further supported by the NDCG calculations, which show that the enrichment gives a decrease in result from the term rank from 0.89 to 0.69 (see equation 51).

Table 65: Top 5 local centrality result for *Q5*

| *Query:* "Harry Potter's third year at Hogwarts" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | J. K. Rowling | Author |
| 2 | Harry Potter and the Prisoner of Azkaban | Book |
| 3 | Harry Potter and the Philosopher's Stone | Book |
| 4 | Harry Potter and the Chamber of Secrets | Book |
| 5 | Harry Potter and the Goblet of Fire | Book |

$$NDCG_5Q5 = \frac{DCG_5}{IDCG_5} = \frac{7.02}{10.17} = 0.69 \tag{51}$$

However, the results for query *Q6* were considered bad and irrelevant. The results can be seen in table 66, which showed that the enrichment retrieved only one moderately relevant entity within top 5, with the rest being considered irrelevant. The NDCG score for query *Q6* is therefore as low as 0.16 (see equation 52), which further shows the bad effects the enrichment has on this particular query. The enrichment returns two authors at the top, which share term matches with the query, but has by the participant of the survey been deemed irrelevant. The reason the enrichment has enhanced this in the ranking, however, is because of their large amount of connections, making them the most central entities of the subgraph from the retrieved entities.

Table 66: Top 5 local centrality result for *Q6*

| *Query:* "Jurassic Park aimed for young adults" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | Paul Park | Author |
| 2 | Ella Young | Author |
| 3 | Jurassic Park | Book |
| 4 | Dream Park | Book |
| 5 | Chronicles of the Nephilim for Young Adults | Series |

$$NDCG_5Q6 = \frac{DCG_5}{IDCG_5} = \frac{2.06}{12.90} = 0.16 \tag{52}$$

Overall, the evaluation of the enrichment for descriptive queries showed that term matches between the query and entities play a significant role and the centrality of the entities where irrelevant. Therefore, the enrichment did not provide significant improvements in the ranking.

**List Queries**

The last evaluation for the local centrality score is for the list queries, which are seeking specific types of related entities. *Q7* is as mentioned earlier seeking the Hunger Games books, while *Q8* is seeking all the books from the Harry Potter series. Further, how local centrality can affect the retrieval of these books is investigated.

For query *Q7* the final result with the local centrality is presented in table 67. As seen by these results, the ranking returns all three books within its top 12 results, which is a huge improvement from the initial term rank which does not retrieve the two last books until rank 61. This result shows that local centrality can play a part in finding the related entities. This is because of the book's connection with the other entities in the subgraph. For example, the Hunger Games books will be connected to the author, the series, each other, and various publication which appears in the subgraph, making it receive a high centrality score.

Table 67: Local centrality result for *Q7*

| *Query:* "The Hunger Games books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | The Hunger Games | Book |
| 2 | Suzanne Collins | Author |
| 3 | The Hunger Games | Series |
| 4 | The Hunger Games | Publication |
| 5 | The Hunger Games Trilogy (box set) | Book |
| ... | ... | ... |
| 9 | Mockingjay | Book |
| ... | ... | ... |
| 12 | Catching Fire | Book |

The enrichment performs similarly with the other list query *Q8*, it identifies the Harry Potter book to be highly relevant as it also receives a high centrality score in the ranking. In fact, the enrichment performs even better for this query as it retrieves all Harry Potter books within the top 8 in the ranking, as shown in result table 68.

Table 68: Local centrality result for *Q8*

| *Query:* "Harry Potter books" | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | J. K. Rowling | Author |
| 2 | Harry Potter and the Prisoner of Azkaban | Book |
| 3 | Harry Potter and the Philosopher's Stone | Book |
| 4 | Harry Potter and the Chamber of Secrets | Book |
| 5 | Harry Potter and the Goblet of Fire | Book |
| 6 | Harry Potter and the Order of the Phoenix | Book |
| 7 | Harry Potter and the Deathly Hallows | Book |
| 8 | Harry Potter and the Half-Blood Prince | Book |

The effectiveness of the enrichment can also be proven by the precision and recall calculations that are presented in table 69. These calculations make it clear that the recall of both queries is 1 on the top 15 results, indicating that all relevant entities were successfully retrieved within the top 15 ranking. In addition, the precision scores for both queries increased significantly from the term rank results.

Table 69: Precision and recall for *Q7* and *Q8*

| **Query** | **P@5** | **P@10** | **P@15** | **R@5** | **R@10** | **R@15** |
|---|---|---|---|---|---|---|
| *Q7* | 0.2 | 0.2 | 0.2 | 0.33 | 0.67 | 1 |
| *Q8* | 0.8 | 0.7 | 0.47 | 0.57 | 1 | 1 |

With this result, one can conclude that the enrichment effectively improves the ranking for list queries significantly.

**Summary**

The result above once again shows that enrichments can give both positive and negative outcomes on different types of queries, this is also shown to be the case for the local centrality score. Local centrality has proven to be an advantageous aspect in many cases, as shown for queries *Q1*, *Q7*, and *Q8*. However, it has also proven to be irrelevant and can skew the result in the wrong direction for other queries, from the evaluation this was seen to be the case for both the named entity queries and the descriptive queries. This does make sense when one thinks of the objective of the two types of queries, which both depend on identifying entities that have term matches or are similar to the query terms. In such cases, adding centrality enhancement has no advantage, as the terms of the entities are more important than their connections. In addition to that, similar to Tonon's hybrid rank and entity linking ranking the enrichment as high as has been done here has proved not to be the best choice, it is therefore not recommended to have as high weighting for the centrality enrichment. A lower weighting would result in the enrichment only being used to differentiate between similarly ranked entities, which is essentially the point of the centrality score.

In summary, the local centrality technique has been proven to be a valuable tool for certain queries and has in multiple evaluations significantly improved the result from the original term ranking.

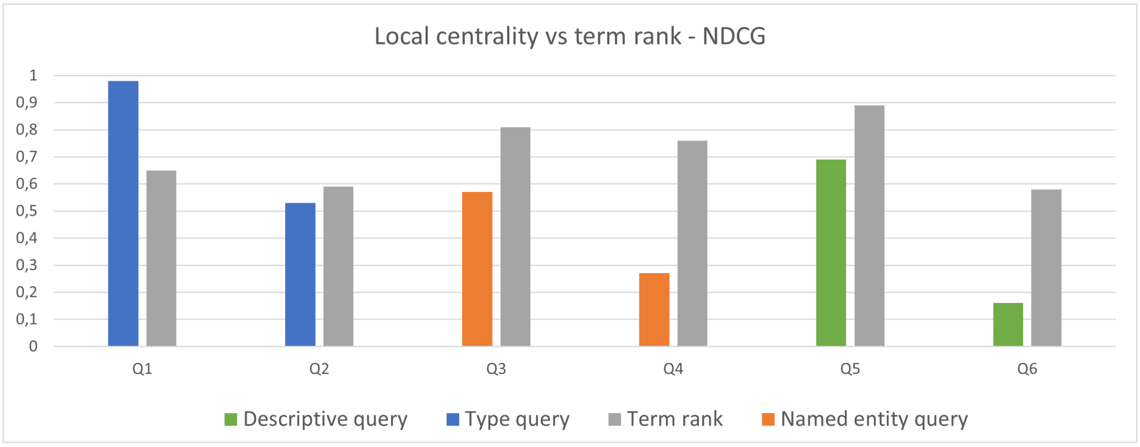


Figure 25: The NDCG scores for Local centrality

## 7.2 Discussion

After evaluating the enrichment techniques in the previous section, this section presents a discussion of the results obtained. The main objective of this section is to analyze and compare the effectiveness of the different enrichment techniques in improving the retrieval of semantic data. In addition to that the sections aim to further evaluate the enrichment, as they until now only have been evaluated on their own and with a set weighting to understand each of their effects. In particular, the chapter aims to answer the research questions raised in the introduction, including what impact the different enrichment techniques have on the semantic data retrieval performance, and whether the use of the enrichment techniques can lead to more relevant search results compared to traditional methods. The chapter also includes a discussion of the strengths and weaknesses of each technique and provides recommendations for future research in the field of semantic data retrieval.

### 7.2.1 Target Entity

First, start by looking back at the first evaluated enrichment, the target entity enrichment. These enrichments are meant to identify target entity types for the query and enhance the ranking based on these target types. As shown in sections 7.1.2 and 7.1.3, the enrichment had mixed effects on various types of queries. Target entity enrichment could be an appropriate enrichment for the queries *Q1*, *Q2*, *Q6*, *Q7*, and *Q8*, as they all have a specific type of entity on the top of their relevance list.

Table 70: Target entity for each query vs the identified target entities

| | Q1 | Q2 | Q6 | Q7 | Q8 |
|---|---|---|---|---|---|
| Target entity | Author | Award | Book | Book | Book |
| Entity-centric identified target entity | Book | Book | Book | Book | Book |
| Type-centric identified target entity | Author | Award | - | Book | Book |

As shown in table 70 both enrichment have a fairly high precision for identifying the right target type for the various queries. The type-centric enrichment performed slightly better by identifying four out of five target types compared to the entity-centric enrichment, which identified three correctly. The difference can be explained by the fact that the queries *Q1*, *Q2*, *Q7*, and *Q8* all have the targeted type mentioned in the query, while *Q6* does not explicitly mention its target type.

However, even with the enrichment identifying the correct target type for over half of the queries, it does not effectively improve the ranking for every query. For instance, the type-centric enrichment does not give an improvement on the ranking for query *Q1* due to the term rank not returning the most relevant entity, *J.K. Rowling*, making it impossible for the target entity enrichment to rank this entity. This demonstrates that the term rank and target entity enrichment are insufficient for this query. Nevertheless, the target entity enrichment can be useful when combined with another enrichment, such as Tonon's hybrid rank, entity linking, or local centrality enrichment, that can successfully retrieve the most relevant entity. When tested again, the type-centric enrichment with additional entities retrieved from Tonon's hybrid rank showed improved results with the *J.K. Rowling* entity finally appearing on the top (see the results in table 71). The new test was performed with the weighting 0.2 for term rank, 0.7 for target entity, and 0.1 for Tonon's hybrid rank.

Table 71: Top 5 type-centric and Tonon's hybrid result for *Q1*

| *Query: "Author of Harry Potter"* | | |
|---|---|---|
| **Rank** | **Entity name** | **Type** |
| 1 | J.K. Rowling | Author |
| 2 | Dennis Potter | Author |
| 3 | Robert Potter | Author |
| 4 | Deborah Potter | Author |
| 5 | Alexander Potter | Author |

Overall, target entity enrichment can significantly improve the ranking, particularly for queries seeking specific types of entities or where the most relevant entities are of the same entity type. However, it can skew the result in the wrong direction for queries seeking multiple entities of different entity types. In this case, the type-centric approach avoids this issue by only enhancing the ranking for specific types mentioned in the query, whereas the entity-centric approach identifies target entities based on the top 10 results from the term ranking, potentially leading to worse results for certain queries, as seen in *Q3*, *Q4*, and *Q5*.

In conclusion, the target entity enrichment is a good enrichment that can significantly improve the ranking for many queries. In this dataset and with the particular queries chosen for this evaluation, the type-centric approach gives slightly better results than the entity-centric approach due to its ability to pick up mentioned types in the query and not skew the result for queries without a targeted entity type. However, it should be noted that the entity-centric approach also has its strengths, as demonstrated in query *Q6* where the target type is not explicitly mentioned. In conclusion, a combination of both entity-centric and type-centric enrichment can lead to further improvements in ranking. However, to avoid skewing the results significantly from the original term rank, it is advisable not to weigh the enrichments as heavily as was done in the evaluation. The high weighting was used solely to demonstrate the individual effects of each enrichment method.

### 7.2.2 Tonon's Hybrid Rank

The next enrichment to be further examined is Tonon's hybrid rank. As discussed earlier in this paper, Tonon's hybrid rank has the advantage of retrieving more relevant entities for the final result set, which can be particularly useful for queries like *Q1*, where the initial term rank fails to retrieve the most relevant entity.

Tonon's hybrid rank enhances the ranking by giving an extra boost to the neighbors of the top-ranked entities. However, as discussed in Tonon's hybrid rank evaluation summary (refer to section 7.1.4), weighting this enrichment higher than the initial term rank may not be advisable, as it can decrease the relevance for several queries. This is evident in the case of queries *Q4*, *Q5*, and *Q8*. Tonon's hybrid rank is meant to boost the relevant neighbors of the top entities, not replace the top entities with the neighbors, which is what happens with the previous weighting.

As a result, additional tests were conducted for Tonon's hybrid rank with a different weighting. In this case, the term rank weighting was set to 60% and Tonon's hybrid enrichment weighting was set to 40%. The results of this new weighting are presented in table figure 26.

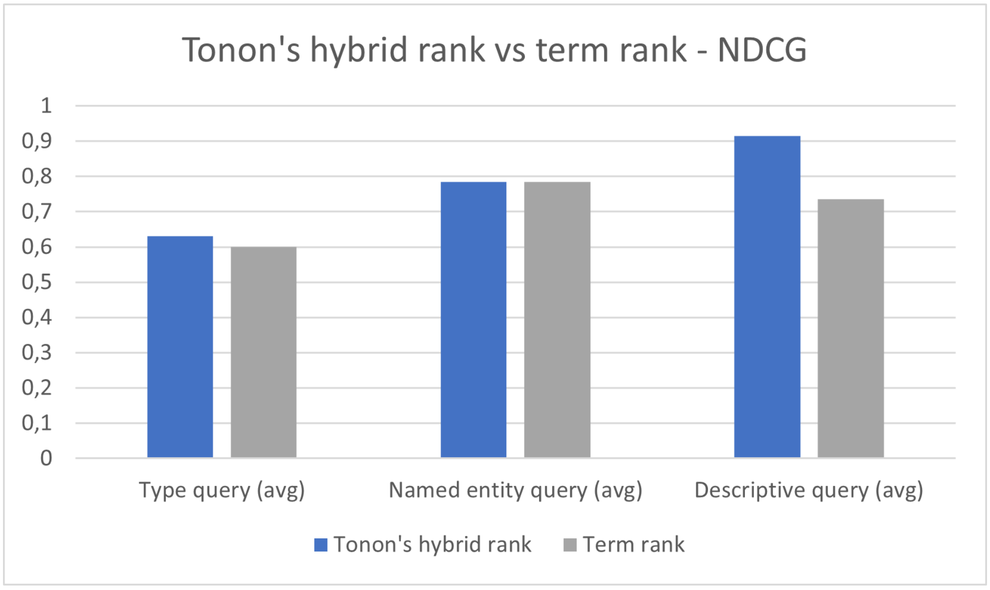


Figure 26: The NDCG scores for Tonon's hybrid rank compared to term rank

The new results provide a clearer demonstration of the advantages of incorporating Tonon's hybrid solution into the ranking process, while also providing a more realistic picture of how the enrichment works due to the inclusion of a more representative weighting scheme.

In conclusion, Tonon's hybrid rank enrichment provides a significant advantage for entity ranking by retrieving more relevant entities and enhancing the ranking by giving the neighbors of the top-ranked entities an extra boost. However, weighting the enrichment higher than the original term rank could lead to a decrease in value for some queries. Therefore, it is important to find a balanced weighting for the enrichment to ensure optimal results. The additional tests conducted in this section with a different weighting demonstrate the advantages of adding Tonon's hybrid solution to the ranking and provide a more realistic picture of how the enrichment works. And with these additional results, it can be concluded that Tonon's hybrid enrichment is a good enrichment to improve the ranking.

### 7.2.3 Entity Linking

Next up is the entity linking approach, which identifies entities that exactly match the query or parts of the query, and utilizes these to enhance the ranking. For instance in query *Q3*, "*Harry*

*Potter and the Philosopher's Stone*", the enrichment finds linked entities that are named *harry*, *and*, *the*, *philosopher*, *stone*, *harry potter*, *the philosopher* and *Harry Potter and the Philosopher's Stone*.

Before starting the evaluation, the entity linking enrichment was expected to perform well for several query types. Especially for named entity queries, where the query is specifying which entity they are looking for by containing their name. The enrichment was then expected to pick up these mentions and enhance the specified entities in the ranking, as well as their close neighbors. However, after observing the result, it was evident that the enrichment did not perform as expected. For instance as shown above for query *Q3*, the enrichment picked up 8 entity mentions for a query originally written to only refer to one, which resulted in the enrichment having many more linked entities than the relevant ones. This was also a case for many other queries, as the enrichment continued to pick up every entity mention in the query whether it was relevant or not. This is due to the large size of the dataset, which increases the chance of having entity mentions from parts of the query.

One suggestion to improve this is to change the enrichment to also consider the number of term matches from the query and entity name. Thereby enhancing entities that share more terms with the query more than entities that only share one or two terms. This would for instance make the enrichment consider the matches for the first book of Harry Potter more relevant than the other entity mentions for *Q3*, which is what the query is seeking.

In addition to that, another thing observed from the result is that the enrichment was considered to have a too strong weighting in the ranking. Similar to Tonon's hybrid evaluation some connected entities received higher scores than entities with high term rank scores. In most cases, this is not desired. However, in a few cases, such as for *Q1*, which searches for an entity without term matches with the query, this approach was particularly effective. The enrichment ended up retrieving the desired entity and ranked it high in the ranking, despite it not appearing in the initial term ranking. Because of this, it is therefore difficult to determine a perfect weighting for the enrichment. This also applies to most of the other enrichments, as they perform differently for each type of query, making it hard to perfect the weighting for every query.

In conclusion, entity linking enrichment did not perform up to expectation for these particular queries on this dataset. Still, entity linking proved to be valuable for a few queries like *Q1*, *Q4*, and *Q8*. In addition to that the enrichment is assumed to perform better with further improvements in the prioritization of the linked entities, where the amount of term matches plays a part. This would assumably give better results for *Q3* and *Q7* as well. Making the enrichment fit well for named entity queries and list queries.

### 7.2.4 Centrality

The last enrichments that were evaluated were the two centrality enrichments: PageRank and the local centrality score. PageRank is as mentioned a global centrality score, which calculated the centrality of all entities before query time. This is an effective approach and is in this case for information retrieval in semantic data used to enhance the ranking by boosting the entities with many important connections. As an alternative to the PageRank score, the local centrality enrichment was implemented. This is a centrality score that instead of calculating the centrality before query time, calculates the centrality for the subgraph of the returned relevant entities. The aim of this approach was to have a score that only focused on the relevant connections for the query and not a global centrality score that does not consider the query. In addition to this, the local centrality score also affects the final result set, as it retrieves the neighbors of the retrieved entities.

Upon evaluating the PageRank enrichment, it was observed that it had minimal impact on the ranking. This was attributed to the extensive dataset utilized, which contained a diverse range of connections between entities. The PageRank scores were normalized to align with the other enrichments, which were confined to a range of 0 to 1. However, a study of the PageRank scores for the dataset revealed that the highest scoring entity, an *uncredited* author entity, received a perfect score of 1.0, while the second highest scoring entity only received a score of 0.2. Consequently, all other entities were relegated to extremely low scores, as the top-scoring entity possessed a significantly greater number of connections than the rest. Given the low PageRank scores of most entities, the enrichment had minimal impact compared to the other enrichments.

Because of this, a new evaluation was performed to further check the real effects of the PageRank algorithm, this time using another formula in order to give the PageRank score more weight. The new formula used for this evaluation can be seen in equation 53.

$$finalscore = (0.2 * termrank) * (0.8 * enrichmentrank) \tag{53}$$

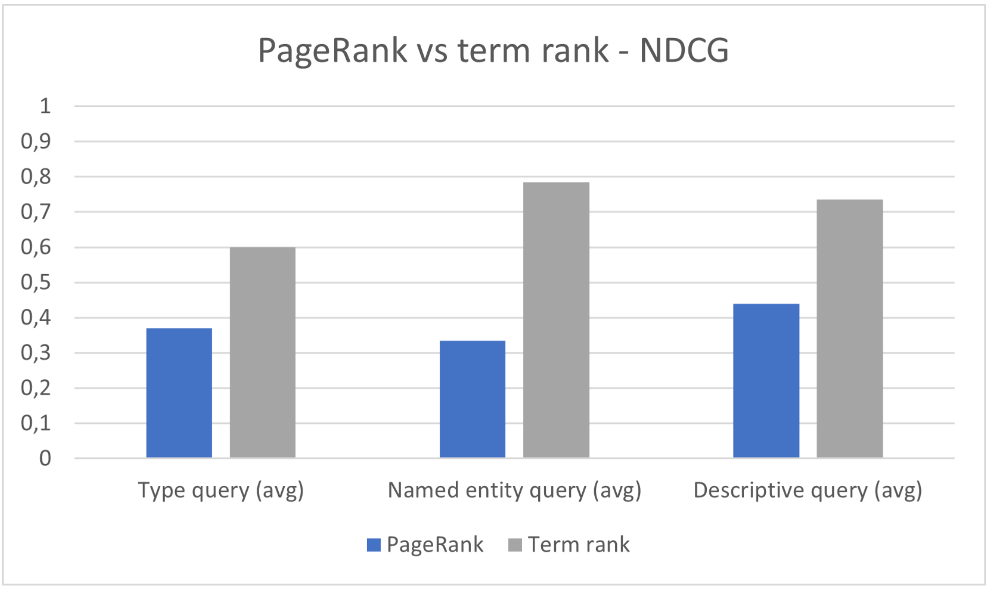


Figure 27: The NDCG scores for PageRank compared to term rank

With these new results, which give a more realistic view of how the PageRank algorithm can affect ranking, one can compare the two centrality algorithms. As seen by the two centrality results, the enrichments both prove to yield overall negative effects on the ranking. This can mostly be explained by the unrealistic high weighting of the centrality enrichment. As centrality enrichment is only meant to slightly affect the ranking between similar ranked entities. The idea of centrality enrichments is that an entity with a large number of connections is of more importance than entities with fewer connections. For instance, in the case where two entities are fairly similar one could separate them based on their connections. However, as proved by the enrichments results, this is not always the case. This is especially clear in this dataset where presumable irrelevant entities receive a much higher centrality score than the relevant entities. For instance, in the Harry Potter query, entities like *Hogwarts Library* received a much larger PageRank score than the *Harry Potter* series and books, which in most realistic cases is deemed undesired.

However, as mentioned the local centrality algorithm differs highly from the PageRank algorithm in that it additionally retrieves extra entities. This is for instance shown to have a positive effect on *Q1*, where the local centrality algorithm receives a near-perfect relevance score. This is also evident in queries *Q5*, *Q7*, and *Q8* where related entities that were previously unranked are now among the top results. To further give a more realistic view of the local centrality enrichment, it is run again with a more realistic weighting, where term rank is weighted 0.8 and local centrality 0.2 in the ranking. The result for this is presented in figure 28.

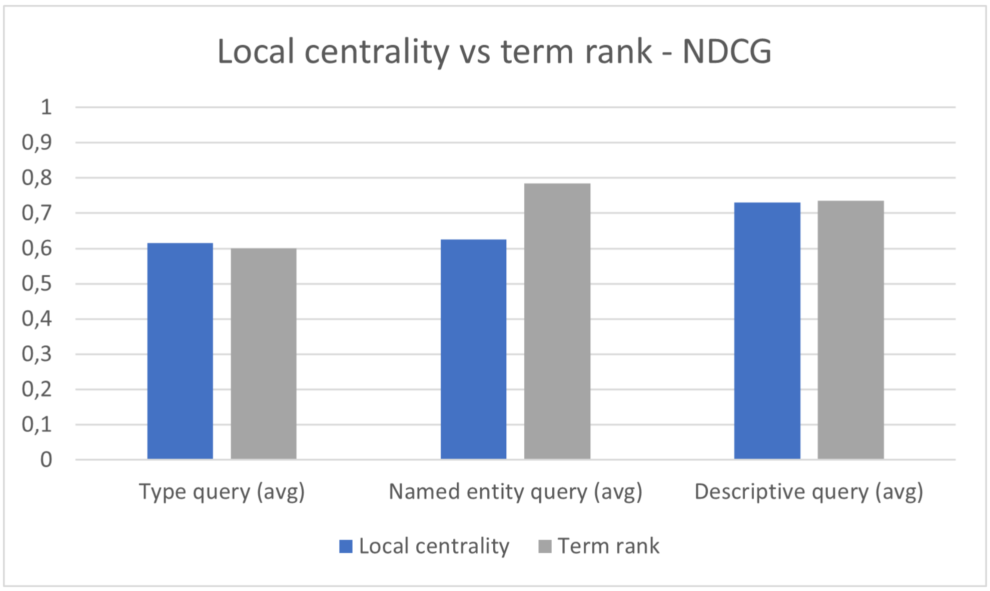


Figure 28: The NDCG scores for local centrality compared to term rank

This result gives a more realistic view of how the local centrality enrichment will perform in a search engine. Here one can see that the enrichment gives slightly improved results for queries *Q1*, *Q2*. This new weighting also improves results for queries *Q7*, and *Q8*. While it negatively affects the ranking for *Q3* and *Q4*. Overall, the enrichment can be seen to be a valuable enrichment, as it improves half of the queries in the evaluation.

In conclusion, the evaluation indicates that the local centrality performs slightly better than the PageRank as a centrality score for these queries and the dataset. This is due to the weakness of PageRank being affected by the irrelevant entities to the query. The dataset contains entities that possess a significantly greater number of connections than other entities, resulting in the PageRank enrichment having a minimal impact on most rankings as most entities have a low PageRank score. In addition to that, the local centrality also has the advantage of including additional entities in the ranking, which can be highly relevant in some instances. However, the local centrality score's negative impact on the system's performance makes it difficult to determine whether the additional entities are worth the trade-off.

#### 7.2.5 Summary

After evaluating every enrichment by themselves it has become evident that each enrichment has a valuable effect on information retrieval on semantic data. As seen, the enrichments each have their own strengths and weaknesses, and each seems to affect positively on several queries, but negatively on some. It is therefore extremely hard to determine which enrichment is of the highest importance and how to best improve the overall ranking for every query. However, one possible way to do it is to look at each separate query type, to determine which enrichment performs the best. This time the result is shown with a more realistic weighting, meaning that the various enrichment has a lower weighting than the previous evaluation. This is to give a more realistic idea of how each enrichment affects the ranking results.

For type queries, the result for the various enrichments is presented in figure 29. Based on this it can be concluded that different enrichments have varying effects on the performance of type queries. While the target entity enrichment is expected to be valuable, it only performs well with the type-centric approach, while failing to identify the correct target entity types in the entity-centric approach. Additionally, the type-centric approach may not perform as well as the term rank approach in terms of overall precision for *Q1*, but it is better at identifying the most relevant entities that the user is likely seeking, such as *J.K. Rowling* in the given example. The enrichments such as Tonon's hybrid, entity linking, and local centrality scores also show improvements in performance for type queries, especially for retrieving the sought-after entity for *Q1*. However, only the type-centric enrichment is significant in improving the ranking for *Q2*, as it is the only enrichment that considered the desired awards to be of relevance. One could conclude that the enrichments target entity type-centric approach, Tonon's hybrid, entity linking, and local centrality are all valuable to type queries, where the type-centric approach proves to be the absolute most relevant.

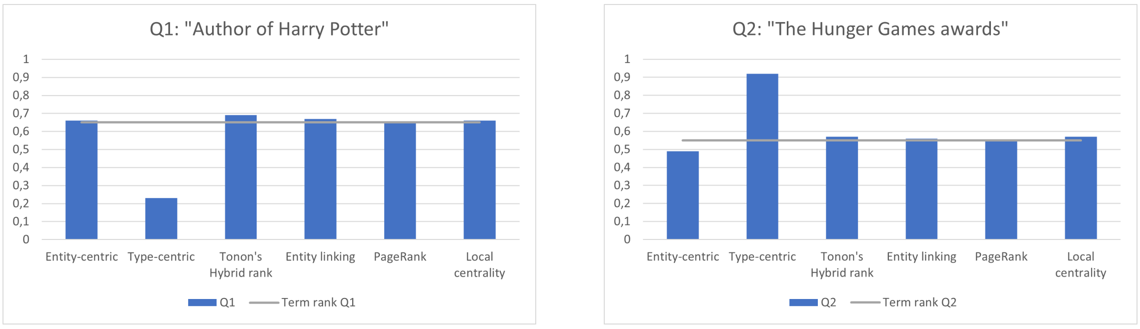


Figure 29: The NDCG scores for the type queries

Moving on, the result for named entities queries are presented in figure 30. With this query type, entity linking is expected to perform well. However, as explained above and as evident by the result, this only happens to one of the queries. In fact, for query *Q3*, non of the investigated enrichment have a positive effect on the ranking, meaning that the traditional term rank performs best for this particular query. For *Q4* on the other hand, the entity linking approach performs as expected and shows to have a valuable effect on this type of query. In addition to that Tonon's hybrid rank also improves the result, proving to be valuable for this particular query. Furthermore, it can be assumed that entity linking could have a better effect on *Q3* if improved to contain a prioritization of linked entities, which is described above. Overall, based on the results presented in figure 30 and the analysis provided, entity linking is presumed to be the best fit for named entities queries, while Tonon's hybrid rank also shows potential as a valuable enrichment for named entity queries.

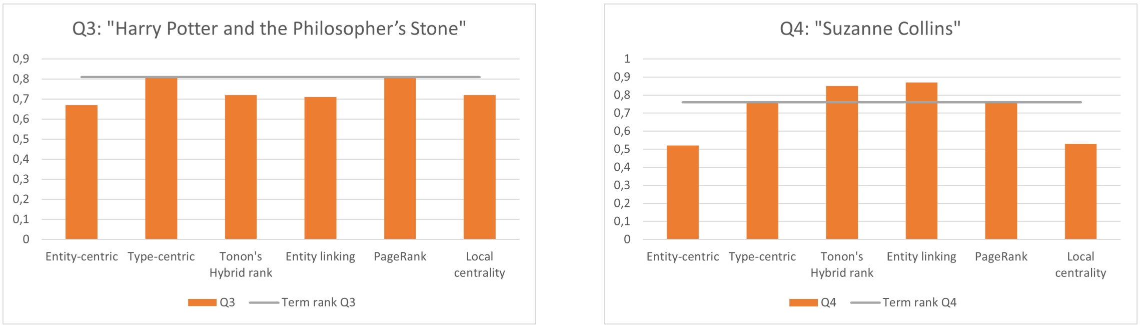
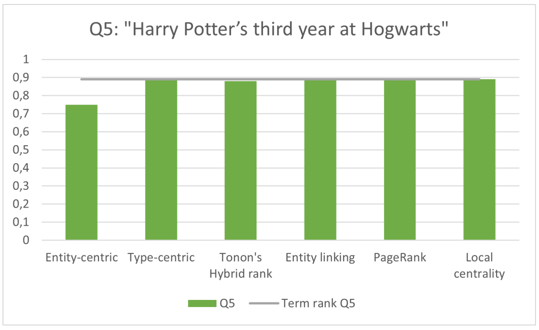


Figure 30: The NDCG scores for the named entity queries

Next, is the result for descriptive queries, presented in figure 31. Descriptive queries aim to identify entities that match a particular description, these descriptions are often found in the metadata of the entities. Because of this, the traditional term rank is expected to give an adequate ranking result as the sought entities contain term matches with the query. This is evident in the result for *Q5*, however, for *Q6* the term rank falls short. This is because the described entities, which are the junior novelization of the Jurassic Park books, are not a direct term match for all the books, resulting in only one of the desired books appearing in the top 5 results. In this case, it is clear that several of the investigated enrichments are valuable for further improving the results. The figure shows that target entity's entity-centric approach, Tonon's hybrid rank, and entity linking all significantly improve relevance for *Q6*. Therefore, it can be concluded that these enrichments are valuable additions for descriptive queries, especially Tonon's hybrid rank, which enables near-perfect results.



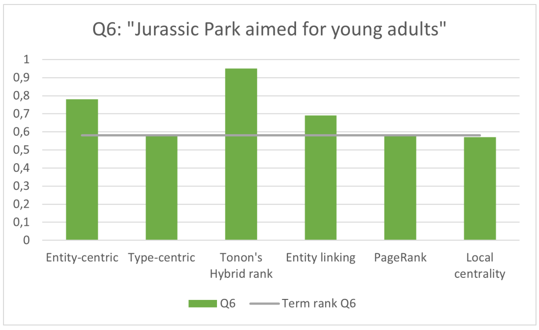


Figure 31: The NDCG scores for the descriptive queries

Lastly, the result from the list queries is presented in table 72, this time only focusing on the last relevant entity retrieved. For instance, the table shows that for query *Q7* the term rank does not retrieve every relevant entity until rank 61 and rank 26 for query *Q8*. This can be seen to be improved with most of the investigated enrichments. For instance, both target entity approaches identify the correct target type and thereby improve the ranking to retrieve all relevant books within rank 26 and 18, respectively. Another case of improvement can be seen with the local centrality enrichment, it does not perform as well as the target entity enrichment, but it still provides improved results for both queries. On the other hand, Tonon's hybrid rank and entity linking enrichments prove valuable for one query while worsening the results for the other, making it difficult to determine their fit for list queries. Therefore, it can be concluded that the target entity approaches and the local centrality enrichment is particularly valuable for list queries. However, it should be noted that list queries aim to gather a complete list of all relevant results, which none of the investigated enrichments can achieve. Further techniques and approaches should be researched to find a more suitable solution for these queries.

Table 72: The ranking number of the last retrieved relevant entity for each enrichment

| | Q7 | Q8 |
|---|---|---|
| Term rank | 61 | 26 |
| Entity-centric | 21 | 18 |
| Type-centric | 26 | 17 |
| Tonon's hybrid | 25 | 30 |
| Entity Linking | 86 | 9 |
| PageRank | 61 | 26 |
| Local centrality | 46 | 19 |

To summarize, the results show that the best enrichment varies depending on the query type. For type queries, the type-centric approach is the most valuable, while for named entities queries, entity linking is the best fit. Descriptive queries benefit from the target entity's entity-centric approach, Tonon's hybrid rank, and entity linking. For list queries, the target entity approaches and the local centrality enrichment are particularly valuable, but there is still room for improvement in this area.

# 8 Conclusion

## 8.1 Conclusion

After conducting a pre-study, it became evident that traditional information retrieval (IR) methods are not appropriate for semantic web data. Therefore, new technologies and approaches are needed to exploit semantically enriched data for improving information retrieval systems.

This study investigated different approaches for semantically enriched ranking, which leverage the structured information in semantic data and can complement the traditional term-based rankings. Four different techniques were identified and explored: Target entity leverages the type properties of semantic data, Tonon's hybrid rank leverages the graph structure of the data, entity linking leverages entity mentions in the queries, and centrality leverages the connections in the dataset. Different variations of these techniques were implemented and evaluated.

The results demonstrate that each enrichment has its own strengths and weaknesses in improving information retrieval on semantic data. To summarize the evaluation, the study found that the best enrichment varies depending on the query type. For type queries, the type-centric approach is the most beneficial, while for named entities queries, entity linking is the best fit. Descriptive queries benefit from the target entity's entity-centric approach, Tonon's hybrid rank, and entity linking. For list queries, the target entity approaches and the local centrality enrichment is particularly valuable. Therefore, no single enrichment provides a complete solution. Instead, a combination of enrichments may be the most effective approach to achieve optimal results. To further conclude the thesis, the research questions from chapter 1 is answered.

- **R.Q. 1:** How can existing IR technologies be modified to effectively leverage the linked Semantic Web data? This thesis has found 4 different enrichment that can be added to traditional information retrieval systems to further leverage linked semantic data.
- **R.Q. 2:** In what way can the various enrichment techniques improve the relevance of search results compared to traditional methods? The results demonstrate that the various enrichment techniques improve the relevance of search results compared to traditional methods. For instance, target entity enrichment takes into account the user's preference for specific types or categories of entities, and ranks them higher.
- **R.Q. 3:** What is the impact of different enrichment techniques on the performance of semantic data retrieval? The study found that the various enrichment has different impacts on different query types. Meaning that some enrichment is more valuable for certain types of queries than others. For instance, for type queries and list queries, the target-entity enrichment show to give good results.
- **R.Q. 4:** Which enrichment techniques are the most promising in improving the retrieval of semantic data? As mentioned, the enrichment has different impacts on different query types, and it is hard to determine one specific enrichment which is most valuable. No enrichment provides a complete solution, and a combination of them might be the best.

These findings offer valuable insights for information retrieval on semantic data and can guide future research in this area. For instance, based on the information presented in this study, search engines for semantic data can be tailored to better meet the needs of their users. Specifically, by analyzing query logs and user behavior, it is possible to determine which query types are most common and prioritize them accordingly. Prioritizing the most common query types and choosing enrichments accordingly, can optimize search systems to better serve their users.

## 8.2 Future Work

This thesis has investigated the use of different approaches for semantically enriched ranking to improve information retrieval on semantic data. The findings have highlighted the strengths and weaknesses of different enrichment techniques and provided valuable insights for future research in this area. Based on the results of this study, there are several areas where future work could be focused:

- Improving the enrichment methods: While the enrichments investigated in this study have shown promise, there is still room for improvement. For example, the entity linking method could be improved by prioritizing the linked entities.
- Investigating the weighting between the enrichment metrics: As mentioned, the study found that the best enrichment varies depending on the query type and that a combination of the enrichments may be the most effective result to achieve optimal results.
- Investigate the use of query logs and user behavior to further determine which enrichment is the most fitting for the search system.
- Investigate or develop new ranking algorithms. While the enrichments investigated in this study have shown promise, there is still room for improvement in the field of semantically enriched ranking. For instance, for the list queries, different approaches could be used to retrieve all relevant entities for the query. A suggestion could for example be to try various query template techniques for list queries.
- Investigate the use of deep learning: Deep learning techniques have shown great promise in the field of information retrieval and could be useful for improving IR on semantic web data.

Overall, the findings of this study provide valuable insights for future work in the field of information retrieval on semantic data.

# Bibliography

Agarwal, Ganesh, Govind Kabra and Kevin Chen-Chuan Chang (2010). 'Towards Rich Query Interpretation: Walking Back and Forth for Mining Query Templates'. In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: Association for Computing Machinery, pp. 1–10. ISBN: 9781605587998. DOI: 10.1145/1772690.1772692. URL: https://doi.org/10.1145/1772690.1772692.

Balmin, Andrey, Vagelis Hristidis and Yannis Papakonstantinou (2004). 'Objectrank: Authority-based keyword search in databases'. In: *VLDB*. Vol. 4, pp. 564–575.

Balog, Krisztian (2018). *Entity-Oriented Search*. Springer Charm.

Balog, Krisztian and Robert Neumayer (2012). 'Hierarchical Target Type Identification for Entity-Oriented Queries'. In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM '12. Maui, Hawaii, USA: Association for Computing Machinery, pp. 2391–2394. ISBN: 9781450311564. DOI: 10.1145/2396761.2398648. URL: https://doi.org/10.1145/2396761.2398648.

Beitzel, Steven M et al. (2005). 'Improving automatic query classification via semi-supervised learning'. In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 8–pp.

Blanco, Roi, Peter Mika and Sebastiano Vigna (2011). 'Effective and Efficient Entity Search in RDF Data'. In: *The Semantic Web – ISWC 2011*. Ed. by Lora Aroyo et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 83–97. ISBN: 978-3-642-25073-6.

Blanco, Roi, Giuseppe Ottaviano and Edgar Meij (2015). 'Fast and space-efficient entity linking for queries'. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pp. 179–188.

Bortnikov, Edward et al. (2012). 'Modeling transactional queries via templates'. In: *Advances in Information Retrieval: 34th European Conference on IR Research, ECIR 2012, Barcelona, Spain, April 1-5, 2012. Proceedings 34*. Springer, pp. 13–24.

Brin, S. and L. Page (1998). 'The Anatomy of a Large-Scale Hypertextual Web Search Engine'. In: *Seventh International World-Wide Web Conference (WWW 1998)*. URL: http://ilpubs.stanford.edu:8090/361/.

Cure, Olivier (2015). *RDF database systems : triples storage and SPARQL query processing*. eng. Waltham, Massachusetts.

Dadzie, Aba-Sah and Matthew Rowe (2011). 'Approaches to visualising linked data: A survey'. In: *Semantic Web* 2.2, pp. 89–124.

Dai, Zhuyun and Jamie Callan (2019). 'Deeper Text Understanding for IR with Contextual Neural Language Modeling'. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 985–988. ISBN: 9781450361729. DOI: 10.1145/3331184.3331303. URL: https://doi.org/10.1145/3331184.3331303.

De Vries, Arjen P et al. (2008). 'Overview of the INEX 2007 entity ranking track'. In: *Focused Access to XML Documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007 Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers 6*. Springer, pp. 245–251.

Garigliotti, Dario, Faegheh Hasibi and Krisztian Balog (2019). 'Identifying and Exploiting Target Entity Type Information for Ad Hoc Entity Retrieval'. In: *Inf. Retr.* 22.3–4, pp. 285–323. ISSN: 1386-4564. DOI: 10.1007/s10791-018-9346-x. URL: https://doi.org/10.1007/s10791-018-9346-x.

Goldberg, Ken et al. (2001). 'Eigentaste: A constant time collaborative filtering algorithm'. In: *information retrieval* 4, pp. 133–151.

Gövert, Norbert and Gabriella Kazai (2002). 'Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2002.' In: *INEX Workshop*. Citeseer, pp. 1–17.

Guo, Jiafeng et al. (2009). 'Named Entity Recognition in Query'. In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: Association for Computing Machinery, pp. 267–274. ISBN: 9781605584836. DOI: 10.1145/1571941.1571989. URL: https://doi.org/10.1145/1571941.1571989.

Han, Shuo et al. (2017). 'Keyword search on RDF graphs-a query graph assembly approach'. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 227–236.

Hasibi, Faegheh, Krisztian Balog and Svein Erik Bratsberg (2016). 'Exploiting Entity Linking in Queries for Entity Retrieval'. In: *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval.* ICTIR '16. Newark, Delaware, USA: Association for Computing Machinery, pp. 209–218. ISBN: 9781450344975. DOI: 10.1145/2970398.2970406. URL: https://doi.org/10.1145/2970398.2970406.

Hoffart, Johannes et al. (2012). 'KORE: keyphrase overlap relatedness for entity disambiguation'. In: *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 545–554.

Hogan, Aidan, Andreas Harth and Stefan Decker (2006). 'Reconrank: A scalable ranking method for semantic web data with context'. In.

Huang, Haoda and Benyu Zhang (2009). 'Text Indexing and Retrieval'. In: *Encyclopedia of Database Systems.* Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, pp. 3055–3058. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_417. URL: https://doi.org/10.1007/978-0-387-39940-9_417.

Jain, Vishal and Mayank Singh (2013). 'Ontology based information retrieval in semantic web: A survey'. In: *International Journal of Information Technology and Computer Science* 5.10, pp. 62–69. DOI: 10.5815/ijitcs.2013.10.06.

Jeh, Glen and Jennifer Widom (2002). 'Simrank: a measure of structural-context similarity'. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 538–543.

Kaur, Harpreet and Vishal Gupta (2016). 'Indexing process insight and evaluation'. In: *2016 International Conference on Inventive Computation Technologies (ICICT).* Vol. 3, pp. 1–5. DOI: 10.1109/INVENTIVE.2016.7830087.

Kim, Jinyoung, Xiaobing Xue and W. Bruce Croft (2009). 'A Probabilistic Retrieval Model for Semistructured Data'. In: *Advances in Information Retrieval.* Ed. by Mohand Boughanem et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 228–239. ISBN: 978-3-642-00958-7.

Laclavik, Michal et al. (2015). 'Search Query Categorization at Scale'. In: *Proceedings of the 24th International Conference on World Wide Web.* WWW '15 Companion. Florence, Italy: Association for Computing Machinery, pp. 1281–1286. ISBN: 9781450334730. DOI: 10.1145/2740908.2741995. URL: https://doi.org/10.1145/2740908.2741995.

Metzger, Steffen, Ralf Schenkel and Marcin Sydow (Dec. 2017). 'QBEES: Query-by-Example Entity Search in Semantic Knowledge Graphs Based on Maximal Aspects, Diversity-Awareness and Relaxation'. In: *J. Intell. Inf. Syst.* 49.3, pp. 333–366. ISSN: 0925-9902. DOI: 10.1007/s10844-017-0443-x. URL: https://doi.org/10.1007/s10844-017-0443-x.

Metzler, Donald and W Bruce Croft (2005). 'A markov random field model for term dependencies'. In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 472–479.

Pérez-Agüera, José R et al. (2010). 'Using BM25F for semantic search'. In: *Proceedings of the 3rd international semantic search workshop*, pp. 1–8.

Pershina, Maria, Yifan He and Ralph Grishman (2015). 'Personalized page rank for named entity disambiguation'. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 238–243.

Pound, Jeffrey, Peter Mika and Hugo Zaragoza (Jan. 2010). 'Ad-hoc object retrieval in the web of data'. In: pp. 771–780. DOI: 10.1145/1772690.1772769.

Robertson, Stephen, Hugo Zaragoza and Michael Taylor (Nov. 2004). 'Simple BM25 extension to multiple weighted fields'. In: pp. 42–49. DOI: 10.1145/1031171.1031181.

Shen, Wei, Jianyong Wang and Jiawei Han (2014). 'Entity linking with a knowledge base: Issues, techniques, and solutions'. In: *IEEE Transactions on Knowledge and Data Engineering* 27.2, pp. 443–460.

Singhal, Amit et al. (2001). 'Modern information retrieval: A brief overview'. In: *IEEE Data Eng. Bull.* 24.4, pp. 35–43.

Tonon, Alberto, Gianluca Demartini and Philippe Cudre-Mauroux (2012). 'Combining inverted indices and structured search for ad-hoc object retrieval'. In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pp. 125–134.

Tunkelang, Daniel (2009). *Faceted search.* Vol. 5. Morgan & Claypool Publishers.

Vallet, David and Hugo Zaragoza (July 2008). 'Inferring the most important types of a query: a semantic approach'. In: pp. 857–858. DOI: 10.1145/1390334.1390541.

Wu, Ledell et al. (2019). ‘Scalable zero-shot entity linking with dense entity retrieval’. In: *arXiv preprint arXiv:1911.03814*.

Yu, Liyang (2011). *A developer’s guide to the semantic Web*. Springer Science & Business Media.