POLITECNICO DI TORINO

Master Degree Thesis

Combining link keys and similarity-based approaches to data interlinking

Supervisors
prof. Paolo Garza

Candidates
Giorgio Orioles
matricola: 295474

Company Tutor
Jérôme David

Anno accademico 2022-2023

Summary

Data interlinking plays a crucial role in expanding and enriching linked open data. One approach to address data interlinking involves the utilization of *link keys*, which extends the concept of keys to scenarios where two RDF datasets are described using distinct ontologies. An alternative approach employs various similarities measures to evaluate the proximity of instances and decide if they have to be linked. In this work, we will employ *numerical link specifications* and *document similarity*.
Both approaches have their respective advantages and limitations, and none of them provide a comprehensive solution for all interlinking challenges.
Hence, this master thesis considers ways to combine such approaches elegantly, using the combination operators *disjunction* and *injective complementation*. We will make use of pairs of datasets to perform experiments in order to analyze the performance of different combination methods.
The results show that the combination of the two approaches was productive, although it is very depending on the task we are analyzing.

Acknowledgements

Vorrei dedicare le prime righe di questi ringraziamenti alle persone che mi hanno aiutato allo svolgimento di questa tesi, a partire dal mio tutor francese Jerome David e tutto il team *mOeX*. A loro vanno i miei più sinceri ringraziamenti perchè mi hanno accolto nel loro gruppo rendendendosi sempre disponibili all'ascolto e a darmi una mano per il completamento del report.
Vorrei ringraziare il mio referente italiano il professore Paolo Garza, di cui provo un'immensa stima, che nel corso dei mesi ha sempre sostenuto e seguito la mia ricerca. I ringraziamenti speciali vanno alla mia famiglia, che mi è sempre stata vicina in quest'anno all'estero, sostenendomi nei momenti più difficili e indirizzandomi verso la via più giusta da seguire. A loro va la dedica più sentita, per l'impegno che hanno messo nel sostenere le mie scelte e per la pazienza che hanno avuto con me.
Vorrei ringraziare anche tutte le persone che sono entrate in contatto con me negli ultimi 2 anni, i miei colleghi di Torino con cui abbiamo condiviso pranzi e cene a parlare di esami e tutti i miei amici di Trapani dislocati in ogni parte d'Italia con cui ci sentivamo regolarmente per concederci qualche momento di gi01a tutti insieme.
Infine dedico questa tesi a due persone che non potranno assistere di persona a questo giorno di festa, i miei nonni, nonostante sono sicuro che da qualche parte anche loro staranno festeggiando con un bicchiere di succo alla pera e un dolcetto di mandorle.

Contents

List of Tables

List of Figures

1 Introduction

In today's society, we have access to vast amounts of information from various sources such as governments, universities, and online newspapers. The abundance of data from various sources has led to the emergence of *linked data* [11], which refers to a collection of practices facilitating the publication of structured data on the web. Linked data principles involve the use of *Uniform Resource Identifiers* (**URIs**) as identifiers and technologies like *Resource Description Framework* (**RDF**) to establish semantic relationships between entities and describe them.
One of the significant advantages of linked data is the ability to identify the same entity across different datasets through links. For instance, these links can identify the same books or articles in distinct bibliographical data sources. By leveraging these links, we can exploit the content of multiple data sources and draw inferences between datasets. Therefore, identifying the manifestation of the same entity across various datasets is a critical task in linked data. The process of identifying and creating links is called **data interlinking**.
It is a practical approach that aims to enhance the effectiveness of data analysis and utilization, when dealing with diverse data extracted from multiple sources. Different approaches and methods have been proposed to solve this task [19], [27]. We can classify them in two approach: *numerical methods* and *logical methods*.
Numerical methods examine the property values of two resources and determine which pairs should be linked based on *similarity measurements*. On the other hand, logical methods employ an axiomatic characterization to determine when two resources are considered the same, enabling the discovery of *owl:sameAs* links between URIs from different datasets.
However, relying on a single approach may be limiting when attempting to generate all the necessary links in certain tasks. Instead of solely selecting the method with the best performance, it may be worthwhile to explore the optimal combination of both numerical and logical methods.
This report will focus on defining and combining two **link specifications** [29], expressed using different methodologies, that have been extracted from two RDF datasets. For that purpose, we will:

- Define what a link specification is.
- Explain what the two different approaches are for expressing a link specification and how to combine them.

- Provide the semantics of compound operators.
- Discuss strategies for finding the best combination.
- Evaluate these strategies with experiments.

The report will encompass both theoretical and practical aspects. The theoretical section will include definitions of link specifications and compound operators, while the practical section will involve an analysis of various experiments and the exploration of practical techniques for achieving the desired goals.
In Chapter 2 we consider related work in data interlinking, and especially approaches that consider combining link specifications, than in Chapter 3 we give the definition of link specifications, link keys and present the tools used. Moving forward (Chapter 4) we will discuss the evaluation measures used for analyzing the results obtained. Chapter 5 will primarily focus on combination operators, presenting mathematical definitions as well as practical examples. In Chapter 6, we will present the datasets and methodology employed in our experiments. Finally, in Chapter 7-8-9, we will showcase the three conducted experiments, accompanied by graphical data and a detailed explanation of the results. The report will conclude with a summary of works and conclusions.

2 Related Work

The ultimate goal of data interlinking is to find pairs of URIs that represent the same entity in two different RDF datasets [19] [27] [13]. This process results in a set of links that can be added to the datasets by associating the corresponding IRIs (Internationalized Resource Identifiers) with the *owl:sameAs* property.
The task can be simplified as follows: given two sets of identifiers I_D and $I_D{}'$ from two datasets D and $D^{'}$, we want to find the link set L which denotes the same resource in both datasets.
The work is concerned with the combination and evaluation of several interlinking approach together, we will combine *link keys* and *numerical link specification* with each other.
Most methods roughly compute a numerical link specification $\langle \sigma, \theta \rangle$ made of a similarity measure σ between the entities to be linked and a threshold θ.
In this area Wombat [33] offers a method to navigate the realm of these link specifications, commencing with the fundamental similarity between pairs of datatype property values. Through supervised learning, it acquires the capability to understand conjunction, disjunction, and differentiation of link specifications.
For numerical link specifications, links are usually produced by using a framework, such as SILK [37] and LIMES [28]. LIMES is a framework specifically created for interlinking datasets, incorporating a wide array of similarity measures that encompass numerical attributes. It offers a selection of similarity functions and customizable threshold settings, simplifying the process of defining numerical link specifications.
LIMES operates in a similar manner to SILK, with the main distinction being the absence of expressing the inverse operator and the specific mode of composition in a link specification. While Limes requires the link specification to be written in an *XML* format, Silk provides a graphical interface that enables us to easily connect property compositions from both datasets. KnoFuss [30] is a system for semantic data fusion. It takes as input two semantic datasets represented in RDF and resolves the data linking problem. For this task there are several techniques which can be applied: for example, string based and set-based similarity metrics for individual matching, using ontological constraints and belief networks for dataset matching.
Azmy et al. [8] use RDF2vec [31] for entity matching across knowledge graphs, and show a large-scale study for matching DBpedia and Wikidata, a similar

approach is introduced by Aghaei and Fensel [3], who combine RDF2vec embeddings with clustering and BERT [16] sentence embeddings to identify related entities in two knowledge graphs.
In our work we investigate the application of *link keys*, a specific variant of logical link specifications. Link keys can be seen as an extension of relational database keys, adapted to accommodate situations involving disparate datasets and the unique attributes of RDF.
While a link key can be conceptualized as a set of aligned keys, the relationship between link keys and keys is more intricate.
The key-based methods proposed by Achichi [2] and Farah [18] involve using a key extraction algorithm [34] to extract pairs of keys that can serve as link specifications. These approaches focus on extracting IN-keys, which are based on shared values between properties present in both source and target datasets. It is assumed that the two datasets are described using the same ontology, and as a result, the extracted IN-keys typically correspond to strong IN-link keys, rather than weak IN-link keys.
The distinction between strong and weak link keys lies in their treatment of duplicates when finding links between two datasets. Both types of link keys facilitate link discovery, but they vary in their allowance of multiple resources satisfying the key conditions within each dataset. Weak link keys permit duplicates, whereas strong link keys prohibit any duplicates.
Specifically, when combining keys and an alignment, it forms a *strong link key*, while *weak link keys* are named so because they may not necessarily be comprised of keys. It's worth noting that there may not necessarily be a one-to-one correspondence between keys and link keys, and searching for keys that can be converted into link keys may not always be successful [6] .
KeyRanker [18] outlines an approach for choosing and composing a disjunction of multiple pairs of such keys. The selection process initiates with the most comprehensive candidate, and subsequent key pairs are chosen based on the incremental coverage they provide upon addition.
Atencia [5] propose a method for extracting link keys between two classes from two RDF datasets, which doesn't require an initial alignment between the properties of both datasets or make assumptions about their common vocabularies. This approach extracts link key candidates from the data and then uses measures of the quality of these candidates to select the best option. It doesn't rely on supervised machine learning and, therefore, doesn't require training links.
This deliverable aims at exploring the feasibility of merging link specifications by establishing their syntax and semantics. In particular, it endeavors to expand the approach of extracting link key candidates into the realm of combining extracted link keys, offering valuable insights into the process.

3 Preliminary Definition

3.1 RDF Dataset

Before going into the main topics, I will explain RDF and present the symbolism we will use from now on.
RDF (*Resource Description Framework*) is a fundamental technology that underlies the Semantic Web. It is a standard for representing structured data and knowledge on the web in a machine-readable format and provides a flexible and extensible framework for describing resources and their relationships.
At its core, RDF represents information as triples, consisting of *subject-predicate-object* statements.
The subject refers to the resource being described, the predicate denotes a specific property or attribute of the subject, and the object represents the value or another resource associated with the subject.
In RDF, an **individual** denotes a distinct resource or an *instance* belonging to a class within an ontology. Essentially, it represents a singular object or entity in the actual world.
Overall, RDF plays a crucial role in enabling the Semantic Web vision by providing a foundation for representing, linking, and leveraging structured data on the web.
Now, we provide a definition of an RDF data set and their signature.

Definition 1. (RDF Dataset) Let U be a set of IRIs, B a set of blank nodes and L a set of literals. An RDF data set is a set of triples from $(U \cup B) \times U \times (U \cup B \cup L)$.
Given an RDF data set D, we denote by U_D, B_D and L_D, respectively, the sets of *IRIs, blank nodes* and *literals* present in D.

Definition 2. (Signature of an RDF dataset) The signature of an RDF data set D is the tuple $\langle R_D, P_D, C_D \rangle$; where R_D is the set of *object property identifiers*, P_D is the set of *datatype property identifier* and C_D is the set of *class identifiers*.

Recalling that in RDF, an individual may have several values for the same property, we give an example of a *RDF triple*: $\langle$ a1:o_1, rdf:*type*, a1:*Book* $\rangle$, i.e. all those entities with an identifier o_1 of type Book.

3.2 Link specification

In order to generate links between two RDF data sets, we give the definition of **link specification**. This term encompasses numerical and logical link specifications.

Definition 3. (Link specification) *A link specification is a function which associates to two data sets D and $D^{'}$ a set of links $L_s(D, D^{'})$.*

In order to process a link specification, we will use two frameworks Silk[1] and Linkex[2], allowing expression of numerical and logical link specifications respectively. The present work is concerned with the combination and evaluation of two link specification's methods together.
A *numerical link specification* $\langle \sigma, \theta \rangle$ is made of a similarity measure σ and a threshold θ. The idea is that two very similar entities can be considered as equals.
For a given pair of datasets D and $D^{'}$, such specifications may generate links through (adapted from [33]) :

$$\mathrm{L}_{\sigma,\theta}^{D,D^{'}} = \{\langle o, o^{'} \rangle \in \mathrm{I}_D \times \mathrm{I}_{D'}; \sigma(o, o^{'}) \geq \theta\}$$

That is, given two entities $(o,\ o')$ belonging to two sets of individual identifiers I_D and $\mathrm{I}_{D'}$ from two RDF data sets, we will consider them to be equal and include them in the set L of links if and only if their similarity value is greater than θ.

We will use Silk Workbench to show an example of numerical link specification, to do this we will use a default Silk task consisting of two datasets containing entities on the movie theme.

[1] http://silkframework.org/

[2] https://gitlab.inria.fr/moex/linkex

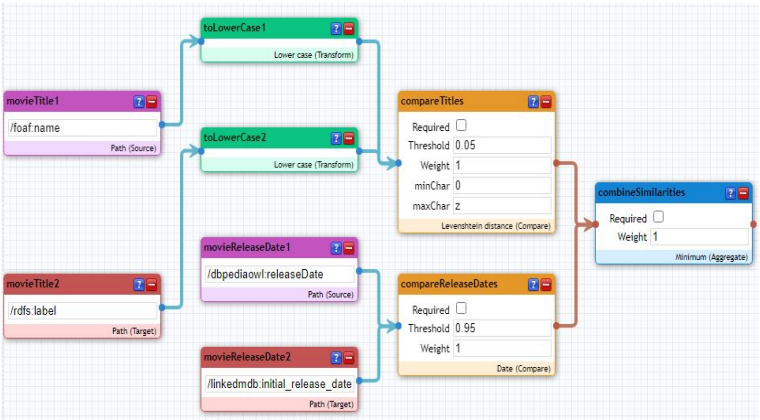


Figure 3.1: Example of a numerical link specification

Figure 3.1 illustrates an example of a numerical link specification implemented in Silk. Given two entities, they are equivalent if certain conditions are met. If the Levenshtein similarity between the "*name*" property value and the "*label*" property value is greater than 0.92 *and*, if the similarity value between the release dates of both entities is greater than 0.95, than the two entities **are** deemed **equivalent**. The *and* operator is represented by the *minimun* aggregator. As we can see in this example the choice of similarity measure will vary for each property comparison, and we should opt for the most suitable algorithm for each comparison.

A *logical link specification* is a logical axiom used to find links between two different datasets, the links are therefore a consequence. Unlike numerical specifications, it can be combined with other kinds of knowledge, such as ontologies and ontology alignments, to infer links by using reasoning [9] [10] [23] [32]. While logical link specifications do not inherently include similarity metrics, it is possible to deal with such metrics separately through the implementation of specific rules or as part of a data preprocessing stage if deemed necessary.
Key-based specification are one of the several ways to express a logical link specification.
Interlinking is usually accomplished in key-based approaches by extracting keys from RDF datasets and then integrating them with ontology alignments [18] [2] [34] [1].
In this context, we explore the utilization of ***link keys***, which are a specialized form of logical link specification. Link keys can be regarded as an extension of relational database keys to encompass scenarios involving two distinct datasets and the unique characteristics of RDF [17] [5] .

Definition 4. (Link key) *A link key expression over two signatures* $\langle R, P, C\rangle$ *and* $\langle R^{'}, P^{'}, C^{'}\rangle$ *is an element of* $2^{(P\times P^{'})\cup(R\times R^{'})}\times\ 2^{(P\times P^{'})\cup(R\times R^{'})}\times(C\times C^{'})$, *i.e.*

$$\{\langle p_i, p_i^{'}\rangle\}_{i\in EQ}, \{\langle q_j, q_j^{'}\rangle\}_{i\in IN}, \langle c, c^{'}\rangle$$

such that EQ and IN are (possible empty) finite set of indices.

An example of a link key expression is:

$\{\langle prenom, firstName\rangle\}$, $\{\langle nom, lastName\rangle\}$ *link key* $\langle Employés, Employees\rangle$

states that if an entity of the dataset *Employés* shares all the values of the property *prenom* with all the values of the property *firstName* of dataset *Employees*, and at least one value of the property *nom*, with the value of the property *lastName* of the dataset *Employees*, then they represent the same entity.
Let us clarify this concept using a practical example:

Employés

id	*prenom*	*nom*
a1	Giorgio	Orioles
a2	Julie, Mary	Jang
a3	Helene	Mayer
a4	Paul, Mark	Rudd

Employees

id	*firstName*	*lastName*
b1	Giorgio	Orioles
b2	Julie	Polar
b3	Helene, Anna	Mayer
b4	Paul, Mark	Rudd

Table 3.1: Entities of datasets **Employés** and **Employees**.

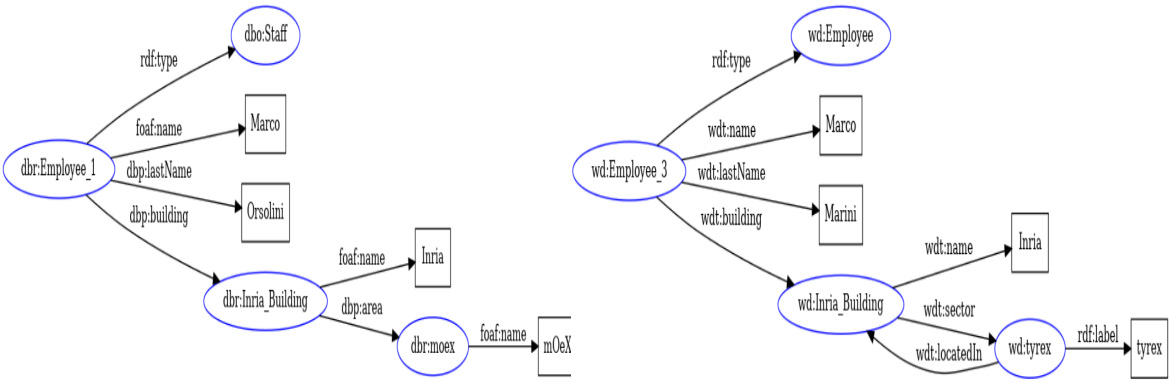
Applying the previous link key to these two datasets, we will get $\langle a1, b1\rangle$ and $\langle a4, b4\rangle$ as links; $\langle a3, b3\rangle$ is not selected because the first condition of sharing all first names is not met.

3.2.1 Symbolism

Since in the following chapters we will use a symbolism, to express a link specification, slightly different from the one presented above, I will present it and clarify its use. Let us begin with the premise that this symbolism is valid for both link keys and numerical link specifications, and in addition, from now on we will only represent and analyse *IN-keys* (sharing at least 1 value per property).
While comparing properties between two distinct datasets, the choice of how to

express a link specification depends on the specific properties being compared. However, it is not always necessary to directly compare two properties; there is also the option of composing them to create a new link specification.
Since RDF datasets are graph-based, with entities pointing to other entities, it is important to remember that object of properties can themselves be entities with their own properties. This allows us to express compositions of properties. The *composition* operator (*"/"*) will be used to represent a direct composition between two properties. To clarify this concept, let us give an example:

Example 1: Given two datasets *Staff* and *Employees*, a possible link specification could be {}, {⟨ *dbp:building/dbp:area/foaf:name, wdt:building/wdt:sector/rdf:label* ⟩}, ⟨$Staff, Employees$⟩.
Let us use graphs to better understand how it works.



(a) Staff (b) Employees

Figure 3.2: Graphs of datasets *Staff* and *Employees*

In our example, the two entities Employee_1 and Employee_3 cannot be linked because the name of the area/sector of the building in which they work are not the same ("$mOeX$" != "$tyrex$").
In addition to the composition operator, there is another one, the *inverse* operator (*"\"*) that acts in the opposite direction, searching for all entities that have a property with a value (or entity) equal to the previous property. This operator only has value if there is a composition of at least two properties, otherwise it would lose its meaning, since, as explained earlier, it searches for all entities that have the value of the property indicated after the \ equal to the value of the property indicated before the \; if we remove the first property we will not be performing the required operation, since we would have no value to compare. Let us clarify this with an example:

Example 2: Given two datasets *Staff* and *Employees*, a possible link specification with the \ operator could be $\{\langle\ \rangle\}$, $\{\langle$ *dbp:building/dbp:area/foaf:name, wdt:building\ wdt:locatedIn/rdf:label* $\rangle\}$, $\langle Staff, Employees\rangle$.

When we look at the Figure 3.2 again, we find that the two entities don't correspond. In *Employees* dataset, we need to fetch the names of the sector entities that share the same building as Employees_3. In this case, they don't match because "*mOeX*" is not equal to "*tyrex*".
In addition to generating property compositions, we can create a link specification containing several property comparisons.

Example 3: Given two datasets *Staff* and *Employees*, a possible link specification could be: $\{\langle\ \rangle\}$, $\{\langle name, name\rangle, \langle building, building\rangle, \langle team, group\rangle\}$, $\langle Staff, Employees\rangle$.

The example shows 3 different pairs of properties, links will be generated between entities that meet all three property comparisons. Thus only entities with the same name, the same building and the same team will be added to the set $L_{D,D'}$.
By examining this final example, we can gain a clearer understanding of the distinction between link keys and numerical link specifications. A link key consisted of multiple property comparisons, can be interpreted as a combination of *atomic* link specifications, resulting in the intersection of the links identified by each of them.

Example 4: Given the link specification of Ex. 3 and the three *atomic* link specification *a = (name, name)*, *b = (building, building)*, *c = (team, group)*.

$$\mathbf{L}_{D,D'} = \mathbf{L}_a \cap \mathbf{L}_b \cap \mathbf{L}_c$$

On the contrary, numerical link specifications provide the option to select aggregators that merge similarity scores from several comparisons or aggregation operators into a single score based on a designated aggregation function. Popular examples of aggregation functions include the weighted average and the minimum score yielded by all operators.
In our work we will always use the *minimum* operator, as it acts similarly to what we have seen for link keys, i.e. it finds the intersection between the link sets of the different atomic link specifications.
From now on, we will use these symbolism to express the link specifications analysed in the experiments.

3.3 Tools

Let shift our attention to the specific tools I utilized, namely *Linkex* and *Silk*. The selection of Linkex is driven by its unique capability to extract link key candidates, whereas other tools merely extract key pairs without emphasizing property composition.
On the contrary, Silk is among the numerous tools available for generating numerical link specifications. The choice was made because Silk is likely the most comprehensive and user-friendly option. Its graphical interface facilitates the creation of link specifications, and it offers a wide range of features not found in other tools currently available in the market.

3.3.1 Linkex

Linkex [1] is a tool allowing to discover *link keys candidates* between two RDF datasets.
A *link key candidate* is a pattern of a link key expression defined as:

Definition 5. (Link key candidate) *A link key expression K is a link key candidate over two classes* C, C′, *if L(K)* $\neq$ *0 and there is no link key expression H such as H* $\subseteq$ *K that generates the same link set as K i.e. L(K)* $\neq$ *L(H)*

In order to correctly extract link keys candidates, Linkex use a *Pattern Structure (PS)* [20] which is a generalization of FCA (*Formal Concept Analysis)* [21] to deal with non binary data.
After computing the PS, it generates a pattern concept lattice using a modified version of the AddIntent algorithm [35]. In this lattice, the intent of each resulting pattern concept corresponds to a link key candidate, and its extent represents the set of identity links generated by that candidate.

The tool allows to use several candidate extraction options, the main ones are:

- *-c*: define the maximum number of properties in a composition
- *-c1 and -c2*: this option allows, for both datasets, the selection of specific classes for analysis, enabling the user to focus on relevant data.
- *-i*: considers inverse of the properties (\ operator)
- *-s*: is a support threshold between [0;1] which enable to consider only properties that are instanciated for the specified percentage of instances.
- *-t ⟨ eq or in ⟩*: define the types of extracted keys (eq or/and in-link keys)

To ensure accurate generation of links, it is necessary to normalize the data by comparing properties. In our case, string normalization is the appropriate approach. Linkex has incorporated various normalization techniques into the code, such as standardizing date formats, rectifying ISBN codes, verifying if a string represents a number, and more. We are specifically interested in two types of normalizations: removal of special characters and conversion of strings to lowercase. The original code included an additional normalization algorithm that treated a string as a collection of letter or digit sequences, sorted them accordingly. However, due to the inability to replicate this normalization on Silk, I chose to remove it and maintain a consistent preprocessing phase across both tools.

After generating a link key candidate, the tool saves the corresponding set of generated links and provides the discriminability and coverage measures. These measures are used for the selection of link keys.

3.3.2 Silk

Silk [37] is an open source framework for integrating heterogeneous data sources. It is built on the Linked Data paradigm, which is founded on two fundamental concepts. Firstly, RDF provides an expressive data model for representing structured information. Secondly, RDF links enable the connection of entities across different data sources.
"Developers can utilize the declarative *Silk-Link Specification Language (Silk-LSL)* to specify which types of RDF links should be established between data sources, as well as the conditions that data items must satisfy to be interlinked" [37].
These link conditions encompass diverse similarity metrics and can incorporate the examination of the chart surrounding a data item, which is accomplished through the utilization of an RDF path language.
Furthermore, the language specifies the access parameters for the pertinent data sources and configures caching, indexing, and preselection functionalities within the framework.
Different aggregation functions can be applied to the similarity scores when creating link conditions.
Silk accesses the relevant data sources through the SPARQL protocol, making it suitable for use with both local and remote SPARQL endpoints. We will create link specifications using the Silk Workbench graphical user interface.

Data Sources

We will use a single method to input datasets into Silk, the Turtle one. We will use Apache Jena Fuseki [12], a SPARQL server that provides a REST-style interface for querying and updating RDF data and provides a web interface for executing SPARQL queries against the uploaded data.
Silk provides the option to select entities belonging to a specific class and apply property restrictions when defining link specifications. This allows for greater precision and control over the linking process.

Link Specification Language

The core of a Silk link specification is the similarity metric section, which outlines how similarity metrics are combined to calculate a total similarity value for a pair of entities. Silk provides several built-in similarity metrics for comparing property values or sets of entities. [37]
Each metric in Silk evaluates to a similarity value between [0;1], with higher values indicating a greater similarity.
The implemented similarity metrics in Silk cover comparisons between strings, numbers, dates, coordinates, etc. However, for the purpose of our project, we will only focus on the string similarity measures since those are the only ones I used.
Specifically, they are: *Jaro-Winkler distance* [39] and *Levenshtein distance* [26], I suggest referring to the book 'Data Matching' [13] to gain a better understanding of the different similarity metrics and their appropriate usage.
I chose to set the threshold for the similarity measure between *0.90* and *0.97*, based on the recommendation in the book 'Data Matching' to use a high threshold in order to achieve the most accurate results.
Aggregation functions can be applied to combine these similarity metrics.
As previously mentioned in Chapter 3, we will be using the *MIN* function. It is worth noting that Silk also provides other aggregation functions, such as maximum, average, product and euclidean distance.
Similar to Linkex, preprocessing functions are utilized in Silk to normalize data and generate a comprehensive set of links. These transformations may vary depending on the type of data being analyzed. In our case, since we are comparing strings, we will be using a filter to remove special characters and convert text to lower case.

Posterior Evaluation

After we have manually completed our link specification, we can proceed to link the data. In addition to generating the set of linked entities, we are given the

option to manually evaluate each individual pair, examining both the property value and the similarity value.
Although Silk allows for the input of reference links, either manually or via a file, to enable the automatic calculation of precision and recall values, I have not been able to use this feature, as to date, it is not working, so I use Alignment API for measuring precision and recall, as I will explain in section 4.1 .

4 Composition Operators

Now, shifting our attention to the primary objective of this work, which is to integrate the two interlinking approaches, let us explore the composition operators. These operators are introduced to be applied to form compound link specifications:

Definition 6. (Compound link specification) *Given two data set signatures* S *and* S$^{'}$*, a compound link specification over* S *and* S$^{'}$ *is* [7]

- either an atomic link specification over S and S$^{'}$, or
- s $\wedge$ s$^{'}$ (*conjunction*)
- s $\vee$ s$^{'}$ (*disjunction*)
- s $\setminus$ s$^{'}$ (*complementation*)
- s $|$ s$^{'}$ (*injective complementation*)
- s $\oplus$ s$^{'}$ (*symmetric injective complementation*)

Definition 7. (Link set of compound link specification) *Let D and $D^{'}$ be two data sets of signatures S and $S^{'}$. Let s and $s^{'}$ be two link specifications over their signatures, the link set generated by a compound link specification is the subset of $c^{D} \times c^{'D^{'}}$ defined as:*

$$\mathbf{L}_{s\wedge s'}^{D,D'} = \mathbf{L}_{s}^{D,D'} \cap \mathbf{L}_{s'}^{D,D'} \tag{4.1}$$

$$\mathbf{L}_{s\vee s'}^{D,D'} = \mathbf{L}_{s}^{D,D'} \cup \mathbf{L}_{s'}^{D,D'} \tag{4.2}$$

$$\mathbf{L}_{s\setminus s'}^{D,D'} = \mathbf{L}_{s}^{D,D'} \setminus \mathbf{L}_{s'}^{D,D'} \tag{4.3}$$

$$\mathbf{L}_{s|s'}^{D,D'} = \mathbf{L}_{s}^{D,D'} \cup (\mathbf{L}_{s'}^{D,D'} \dot{-} \mathbf{L}_{s}^{D,D'}) \tag{4.4}$$

$$\mathbf{L}_{s\oplus s'}^{D,D'} = \mathbf{L}_{(s\wedge s')|(s\vee s')}^{D,D'} \tag{4.5}$$

such that $L_{s'}^{D,D'} \dot{-} L_{s}^{D,D'} = \{\langle o, o'\rangle \in L_{s}^{D,D'};\ o \notin \pi(L_{s'}^{D,D'}) \wedge o' \notin \pi'(L_{s'}^{D,D'})\}$ *where* $\pi(L_s) = \{o \in D; \langle o, o'\rangle \in L_s\}$ *and* $\pi'(L_s) = \{o' \in D'; \langle o, o'\rangle \in L_s\}$

Since our goal is to combine link specifications, we can, for example, select their intersection as *conjunction* does or take the union of the two, like the *disjunction* operator does, or finding more complex combinations expressed by the other operators. Out of the five operators available, I employed ***injective complementation*** and ***disjunction*** in my work. This selection is justified by the suitability of these operators for our objective.
Disjunction, in particular, generates the greatest number of links compared to the other operators, it consists in the union of the links of the two link specifications.
On the other hand, *injective complementation* proves useful as it prioritizes the first link set, enhancing our ability to model the combination by selecting the optimal link sets from both approaches.
To elaborate on the notion of injective complementation, it returns links from the first link specification supplemented with links from the second link specification whose entities are not part of the first link specification.
Unlike disjunction, it gives priority to the first link specification and complements it with never found entities extracted from the second link specification. Since link keys have better precision, we will choose them as the first link specification of the operator, putting for second the numerical link specifications, which on the other hand have better recall. With the aim of integrating positive links generated by numerical link specifications without losing the precision of the link key's link set.
Hence, my anticipation is that *recall* is expected to be higher when using the disjunction operator compared to other operators, as it typically generates a larger link set. To facilitate understanding, let us give an example:

Staff

id	*name*	*company*	*team*
i1	George	Inria	Moex
i2	Julie	LIG	Sigma
i3	Elena	LJK	Dao
i4	Paul	Inria	Tyrex
i5	Marco	LIG	Sigma

Employees

id	*name*	*company*	*group*
z1	George	Inria	Moex
z2	Julie	LIG	Polar
z3	Elena	LJK	
z4	Paul	Inria	Tyrex
z5	Anna		Sigma

Table 4.1: Instances of classes **Staff** and **Employees** of two RDF datasets D and $D^{'}$.

Example 5. Given the following link specifications: $k = \langle name, name \rangle$ and $m = \langle team, group \rangle$, with link sets:

$$L_k{}^{D,D'} = \{\langle i1, z1\rangle, \langle i2, z2\rangle, \langle i3, z3\rangle, \langle i4, z4\rangle\}$$
$$L_m{}^{D,D'} = \{\langle i1, z1\rangle, \langle i2, z5\rangle, \langle i4, z4\rangle, \langle i5, z5\rangle\}$$

We can compute the combination of the two using the two operators *disjunction* ($\vee$) and *injective complementation* ($|$) as follows:

$$\mathrm{L}^{D,D'}_{k\vee m} = \mathrm{L}^{D,D'}_{k} \cup \mathrm{L}^{D,D'}_{m} = \{\langle i1, z1\rangle, \langle i2, z2\rangle, \langle i2, z5\rangle, \langle i3, z3\rangle, \langle i4, z4\rangle, \langle i5, z5\rangle\}$$
$$\mathrm{L}^{D,D'}_{k|m} = \mathrm{L}^{D,D'}_{k} \cup (\mathrm{L}^{D,D'}_{m} - \mathrm{L}^{D,D'}_{k}) = \{\langle i1, z1\rangle, \langle i2, z2\rangle, \langle i3, z3\rangle, \langle i4, z4\rangle, \langle i5, z5\rangle\}$$

Beginning with the *disjunction*, the link set is formed by combining the links from the first link specification with those from the second link specification.
For the *injective complementation*'s link set, we first include all the links from the first link specification, and then we add the links from the second link specification if their entities are not already present in the first link specification. As a result, $\langle i5, z5\rangle$ are added, while $\langle i2, z5\rangle$ are excluded since *i2* is already present in the first link specification as part of the $\langle i2, z2\rangle$ link.

4.1 Analysis of operators

After introducing the two operators we intend to use, I will provide additional information regarding the expected recall values resulting from their combination. This will involve presenting the upper and lower bounds associated with each operator.
However, we cannot do the same for precision, since there are no upper or lower bounds for either operator.

- **Disjunction's Recall:** $recall(\mathrm{L}_{s\vee s'}) \geq \max(\mathrm{recall}(\mathrm{L}_s), \mathrm{recall}(\mathrm{L}_{s'}))$
- **Injective Complementation's Recall:** $recall(\mathrm{L}_{s|s'}) \geq \min(\mathrm{recall}(\mathrm{L}_s), \mathrm{recall}(\mathrm{L}_{s'}))$

Where *min* and *max* represent the minimum and maximum recall value between the link sets of the two approaches.
Both operators enhance the recall of the link specification with lowest recall, resulting in an improvement over that link specification. Regarding precision,

injective complementation favors the first link specification, which was deliberately chosen to have higher precision than the other. This preference should contribute to enhancing the accuracy of the second link specification or potentially improve both link specifications. However, it is important to note that we cannot ascertain this mathematically, as there is also a possibility of precision deterioration in both link specifications.

5 Evaluation Measures

In the field of evaluation measures, we can categorize them into two classes: *unsupervised* and *supervised* measures.
An "*unsupervised*" measurement method refers to a measurement that is calculated without any external guidance or supervision. In simpler terms, it means that we do not have a set of reference links.
On the contrary, a "*supervised*" measurement method involves having a reference link set. The algorithm utilizes the provided true links as supervision to conduct the calculation.
In this work, I employed two supervised measures, namely **recall** and **precision** [38], as the performance metrics for our evaluation. To assess the effectiveness of link specifications, we will refer to a set of reference links, which includes all links deemed to be accurate between the two datasets.
On the other hand, the selection of link key candidates was based on two unsupervised measures, **discriminability** and **coverage**.

5.1 Supervised measures

Assuming we have a ground-truth data set with the true match statuses of all record pairs, we can classify each entity pair into the following four categories [14]:

- *True positive* ***(TP).*** The following record pairs have been identified as both matching and truly representing the same entity. In other words, both records in these pairs refer to the identical entity.

- *False positive* ***(FP).*** These record pairs have been identified as matches, however they do not represent true matches. The two records in each of these pairs refer to distinct entities. The classifier has erred in identifying these pairs as matches, which makes them false matches.

- *True negative* ***(TN).*** These record pairs have been identified as non-matches, and they do indeed represent true non-matches. The two records correspond to distinct real-world entities.

- *False negative* ***(FN).*** These record pairs have been classified as non-matches, but they are in fact true matches. The two records in each of these pairs

refer to the identical entity. The classifier has erroneously labeled these pairs as non-matches, which categorizes them as false non-matches.

Based on the number of true positives , true negatives , false positives and false negatives , different quality measures can be calculated [14], we will focus on *precision* and *recall*.

- *Precision.* In information retrieval, this is a widely used metric for evaluating the effectiveness of search results. It is calculated as

$$\text{prec} = \frac{TP}{TP + FP}$$

  Precision determines the ratio of correctly classified true matches (TP) out of the total number of classified matches (TP + FP). Therefore, it quantifies how accurate a classifier is in identifying true matches.

- *Recall.* Another metric used for evaluation. It is calculated as

$$\text{rec} = \frac{TP}{TP + FN}$$

  It determines the ratio of correctly classified true matches (TP) out of the total number of actual true matches (TP + FN). Therefore, it quantifies how well a link specification has identified actual true matching record pairs as matches.

To compute these measurements using the set of reference links, I employed the Alignment API tool [15]. By providing a file with the links generated from a link specification, the tool yields precision and recall values as output.

5.2 Unsupervised measures

Turning to the unsupervised measures, we will quickly present two measures used to select the best link key candidates [5].

- *Discriminability:* A measure of how closely the links generated by a link specification align with a one-to-one mapping. The purpose of this measure is to identify and eliminate link specifications that link a single entity from one dataset to multiple entities in the other dataset, as such links do not

accurately reflect the true relationships between the datasets.
For example the link key $\{\langle\,\rangle\}$, $\{\langle name, firstName\rangle\}$, $\langle Staff, Employees\rangle$ will produce links with the same entity repeated several times, as, in general, many people share the same name.

- *Coverage:* measures the completeness of a link specification in relation to the datasets, specifically, it evaluates the proportion of instances from both classes that would be correctly linked by the link specification. It always favours the most general link specifications.

We will employ these two measures to identify the link key candidates since, with only unsupervised measures at our disposal, we sought those that can serve as approximations of precision and recall. Specifically, *discriminability* can be regarded as an approximation of precision, while *coverage* can be seen as an approximation of recall. We will focus on candidates with the highest values of discriminability and coverage among all candidates.
This selection is crucial because low discriminability increases the likelihood of reduced precision when the same entity appears multiple times despite being present only once in the reference link set.
Conversely, choosing insufficient coverage would result in too few links for analysis, leading to a lack of representativeness and variability in the data. Additionally, such a choice could introduce higher statistical uncertainty and fluctuation, thereby affecting the reliability of the obtained results.

6 Experimental Setting

In this section, we will present findings from experiments conducted with the two interlinking approaches, as well as their combination using the aforementioned operators. The initial objective of the experiments is to determine which of the two operators yield superior results. However, our primary focus is to demonstrate the potential for obtaining improved link sets by combining link keys with numerical link specification.

6.1 Experiments

After presenting the basics of the report and explaining the main protocols used, let us get into the more practical part by explaining the conducted experiments. The goal is to find the best approach that combines the benefits of link keys and numerical link specifications, to do this we will use the following three experiments.
In the initial experiment, I explored the combination of a link key and its corresponding numerical link specification, referred to as the ***"relaxed link key***", the idea is to use the higher recall of the numerical link specification to add more positive links to the link key thus improving precision and recall.
However, the results did not add value to the link key, which is why our research will progress towards combining ***different link specifications***. Here the goal is to get value from the links of another link specification, for this reason we will not only combine link keys and numerical link specification, but also two link keys.
In this case, we observe significant improvements in the results, indicating that combining two link specifications outperforms generating a single relaxed link key in terms of performance.
For the last experiment, I used a new similarity approach, the ***document similarity*** approach, overcoming the idea of creating links based on comparison of properties, generating links using a global approach that transforms entities into documents, containing their property values, and compares them based on information retrieval methods.
I combined this new approach with link keys, trying to significantly increase the recall of link keys while maintaining high values of precision. The results of the combination led to a substantial improvement over link keys, and a slight increase in the performance of the similarity approach.

6.2 Data Sets

For our three experiments, we will utilize *three distinct pairs of datasets*, with their corresponding set of reference links used for evaluation.
The *Movie Task*, which serves as the default in Silk, encompasses two datasets. One of these dataset comprises entities extracted from DBpedia, while the other contains entities extracted from linkedmdb. The datasets consist of 2132 and 1987 distinct entities, respectively. Property composition can be employed within this context. The entities included in these tasks pertain to films, thus encompassing properties related to movies, such as actors, directors, production companies, and, quite simply, film titles.

The *OAEI 2020 Task*, which is also the largest, comprises two *OAEI 2020 Campaign* datasets focused on the *Star Wars films*. More specifically, these two datasets contain entities extracted from two web pages that provide extensive information about the Star Wars saga: `http://starwars.wikia.com` and `http://swtor.wikia.com`. The first dataset consists of over 145,000 instances across 269 total classes, while the second one contains 4,180 instances across 168 total classes.
The large size of these datasets posed challenges in terms of usability, as both Silk and Linkex required substantial computation time to calculate all possible links. To address this issue, I made the decision to manually reduce the size of the two datasets by selecting entities belonging to a single class. In order to minimize the reduction, I specifically chose the "*character*" class, as it was believed to possess a substantial number of resources; 28430 instances and 804 in the two datasets.
Unlike the other dataset, this particular dataset does not offer the capability to create link specifications with property composition. This limitation arises due to both the excessive computational calculations involved and the fact that the various entities within the dataset are independent of each other.
I want to emphasize that by reducing the datasets, we intentionally decreased the recall values for each approach. However, it is important to note that this reduction in recall is not a critical concern for our objective. Our main goal is to enhance the initial performance of the two approaches without necessarily focusing on the absolute value of recall obtained.

The *Random Task*, comprises a collection of entities and their properties extracted from DBpedia and the second dataset consists of entities extracted from Wikidata. The task was generated by randomly selecting entities from Dbpedia and then matching them with corresponding wikidata entities that are intended

to represent the same entity.
These datasets contain 1184 and 1196 entities, respectively.
Since the task does not focus on a specific macro topic, the entities from the same dataset are not inherently connected to each other. Therefore, the composition of properties should not be considered meaningful.
All three datasets were accompanied by reference link files, that we used in the calculation of precision and recall.

		#instances	**#links**
Movie	*Dbpedia*	2132	*211*
	Linkedmdb	1987	
OAEI 2020	*StarWars*	28430	*1358*
	Swtor	804	
Random	*Dbpedia*	1184	*1184*
	Wikidata	1196	

Table 6.1: Tasks summary table

We will carry out the first and second experiments on the first two tasks, while for the last experiment we will use the Random task.

6.3 Methodology

To handle the first and second experiments, I followed the following methodology: first, I utilized Linkex to extract potential link key candidates, and then I filtered them based on specific criteria related to discriminability and number of links. Among all the candidates, I chose those with the highest values of discriminability and coverage, as explained in detail in chapter 5.2.
Next, I replicated the identical link specification on Silk to obtain the numerical link specification associated with the selected link key.
To do this, I had to analyse which properties were involved in the link specification in order to use the correct similarity measures and set the correct threshold.
After obtaining both sets of links, I merged the two link specifications using the injective complementation and disjunction operators. Initially, I combined two link specifications with similar set of properties, and subsequently with differing ones.
The combined entity comprising a link key and its corresponding numerical link specification will be referred to as a '*relaxed link key*'.

However, it is important to note a possible limitation in the methodology for selecting link specifications.
While Linkex provided all the link key candidates with discriminability and coverage, it was not possible to replicate this functionality on Silk. Therefore, I had to rely on the numerical link specifications corresponding to the link keys. This approach may have some drawbacks, as Linkex does not uncover all possible link specifications, in fact, it only identifies link keys that generate at least one link. Nevertheless, even if a link key does not find any links, it is highly unlikely that its corresponding numerical link specification will yield a consistent number of links and this goes against the rule I used for selecting candidates.
Having clarified this concept, let us proceed with the analysis of the tasks based on the conducted experiments.

7 First Experiment

The experimental process will unfold as follows: we will begin by introducing the two pairs of datasets and outlining the objectives of the experiment. Subsequently, we will present the selected link specifications for each task and analyze the precision and recall results using charts, along with the corresponding percentages of improvement or worsening. Lastly, we will provide the conclusions drawn from the analysis.
We will divide the analysis into two separate sections, one for each task in which the experiment is carried out.

7.1 Movie Datasets

7.1.1 Selected Link specification

Adhering to the methodology guidelines in chapter 6.3, I initiated the process by generating link key candidates using Linkex. Subsequently, I employed a selection criterion that prioritized candidates exhibiting high discriminability and after evaluating the coverage of the link key candidates, I proceeded to select the link keys that demonstrated above-average coverage, with a specific criterion of having a number of links exceeding 40.
This last rule pertaining to the number of links was deemed necessary due to its significance. Selecting link specifications with a small number of links might lead to the distortion of the accuracy and reliability of the analysis.
My choices were these:

- **Link specification n° 25** = {⟨ ⟩},{⟨ dbpp:cinematography \ dbo:cinematography /dbpp:title, dcterm:title ⟩,⟨ dbpp:cinematography \ dbo:cinematography/dbpp:title, rdfs:label ⟩}
- **Link specification n° 45** = {⟨ ⟩},{⟨ dbpp:editing \ dbo:editing/dbpp:name, dcterm:title⟩, ⟨ dbpp:editing \ dbo:editing/dbpp:name, rdfs:label ⟩,⟨ dbpp:editing \ dbo:editing/foaf:name, dcterm:title ⟩,⟨ dbpp:editing \ dbo:editing/foaf:name, rdfs:label ⟩}
- **Link specification n° 64** = {⟨ ⟩},{⟨ dbo:writer\dbpp:writer/dbpp:title, dcterm:title ⟩,⟨ dbo:writer\dbpp:writer/dbpp:title, rdfs:label ⟩}
- **Link specification n° 97** = {⟨ ⟩},{⟨ dbpp:producer\dbo:producer/dbpp:title, dcterm:title ⟩, ⟨ dbpp:producer\dbo:producer/dbpp:title, rdfs:label⟩}

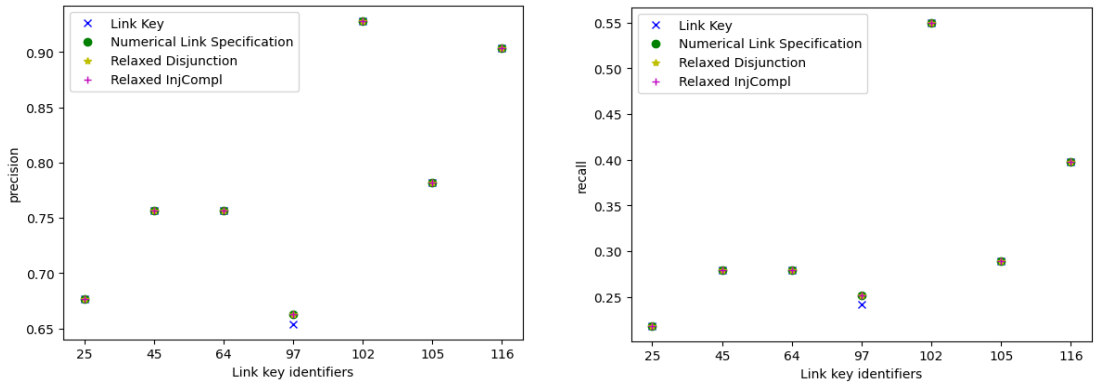
- **Link specification n° 102** = {⟨ ⟩},{⟨ dbpp:name, dcterm:title⟩,⟨ dbpp:name, rdfs:label ⟩,⟨ foaf:name,dcterm:title ⟩,⟨ foaf:name, rdfs:label ⟩}
- **Link specification n° 105** = {⟨ ⟩},{⟨ dbpp:writer\dbo:writer/dbpp:title, dcterm:title ⟩,⟨ dbpp:writer\dbo:writer/dbpp:title, rdfs:label ⟩ }
- **Link specification n° 116** = {⟨ ⟩},{⟨dbpp:title, dcterm:title ⟩,⟨ dbpp:title, rdfs:label ⟩ }

Based on the link specifications, it is apparent that the first dataset consists in general of compound properties. Specifically, only 2 of them do not contain composite properties. The reason behind this is the multilevel entity structure present in the two datasets, where an entity can have another entity as its property value.
But, the inverse function is employed to achieve a more intricate and, in our case, more accurate link specification.
Additionally, it is clear that the final interlinking property is determined by the title or name of the film being analyzed. This analysis holds significant importance in selecting the appropriate string similarity for the numerical link specifications, where I chose to utilize the *Levenshtein distance*, with a threshold of 0.92. The choice was not dictated by an in-depth analysis of property values; rather, I chose the most commonly used string measure.
Each of the chosen link specifications is independent of one another, meaning their link sets are not subsets of each other. This decision was made to maximize the diversity of the link specifications, aiming for higher precision and recall values when combined. By selecting distinct link specifications, the likelihood of duplicate links is reduced, resulting in improved performance.

7.1.2 Results

We now present the results obtained for the relaxed link keys, which I recall being the combination of a link key and its corresponding numerical link specification. Together with charts, I will give some information on the link sets generated for the two approaches. I used an histogram to illustrates the variation in links generated by two approaches utilizing identical link specifications. The purpose of this analysis is to identify any correlations between a higher number of links and improved precision and recall. While it is already established that the numerical link specification yields a greater number of links, we will examine whether the increased discovery of additional links positively impacts recall and precision.

We will now proceed to present the obtained results using two charts that show the precision and recall values of four alternatives: *original link key, numerical link specification, relaxed link key with disjunction and injective complementation.*
The x-axis of the charts will display the reference number of the link specification for the corresponding values.



(a) Precision (b) Recall

Figure 7.1: Charts of precision and recall for relaxed link keys

The precision and recall values frequently are similar between the four alternatives, except for link specification n° 97, which demonstrates a distinct link set, it is the only one that obtain better precision for *numerical link specification.*
In order to provide clearer insights into the results before drawing conclusions, we will present the average improvement percentages of precision and recall for each compound operators (RLK Disjunction = *Relaxed Link key with disjunction* and RLK InjCompl = *Relaxed Link key with injective complementation*) compared to the precision and recall of both specific link specifications involved.

	Precision		**Recall**	
	Link Key	**Sim. Approach**	**Link Key**	**Sim. Approach**
RLK Disjunction	+1.3%	0%	+3.9%	0%
RLK InjCompl	+1.3%	0%	+3.9%	0%

Table 7.1: Average improvement of performance

Upon examining the results on Table 7.1, one could argue that both operators yield similar outcomes, with the numerical link specification often matching the results of the corresponding link key.
However, upon closer inspection of the charts, it becomes evident that the numerical link specification produces identical results to the link key, which was not expected. In theory, the numerical link specification should identify more links, leading to a potential decrease in precision but an improvement in recall.
To investigate this anomaly further, I delved deeper into the issue and discovered that the property involved in the linking process was a string, specifically film titles that often consist of multiple words. Recognizing the error in selecting the Levenshtein distance, I will reanalyze the same task using the *Jaro-Winkler* distance as string similarity measure, as it is better suited for this particular purpose.
Before proceeding with the analysis using the Jaro-Winkler distance, we will first present bar charts displaying the number of links for each of the two approaches.
Following that, we will examine any potential correlations between the difference in links and the performance improvement observed in the relaxed link key approach.

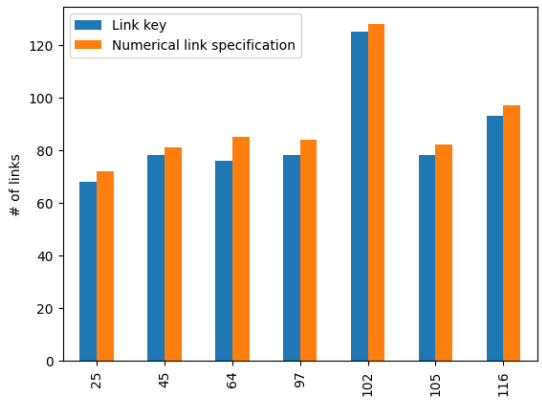


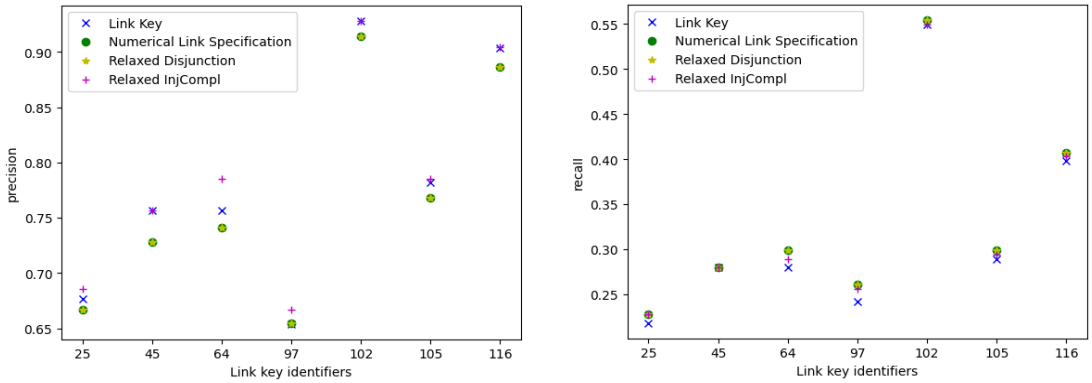
Figure 7.2: Number of links

From the Figure 7.2, it is apparent that the numerical link specification generates approximately 5-10% more links compared to the link key approach.
This could be advantageous since having a significant number of additional links, when combined with the link key's links, may enhance overall performance.
However, if the numerical link specification produced a substantially larger number of links than the link key, there would be a higher likelihood of including mostly incorrect links. In such a scenario, combining the two sets of links would likely result in poorer performance values.

Move on to the charts and table displaying the average increments of the measures.

	Precision		Recall	
	Link Key	**Sim. Approach**	**Link Key**	**Sim. Approach**
RLK Disjunction	-2.1%	0%	**+3.64%**	**0%**
RLK InjCompl	**+0.7%**	**+2.88%**	+2.65%	-1.23%

Table 7.2: Average improvement of performance



(a) Precision (b) Recall

Figure 7.3: Charts of precision and recall for relaxed link keys

7.1.3 Disjunction

Upon analyzing the Table 7.2, it becomes evident that when using the disjunction operator with relaxed link keys, there is a decrease in precision while maintaining a value equal to the maximum recall, obtained by numerical link specifications.
In agreement with the values shown in chart 7.3b, the numerical link specifications have a recall that is greater than or equal to the recall of the link keys. Therefore, the combination of the two approaches did not result in a tangible improvement; instead, it brought the combination's recall to the same level as that of the similarity approach.
Shifting our attention to the charts, we can meticulously observe the precision and recall values for each link specification. It is noticeable that the relaxed link keys with disjunction (represented by the yellow star) consistently exhibit

lower precision values than the link keys, while their recall and their precision is equal to that of the numerical link specification, indeed, one can see the overlap of relaxed link keys with disjunction and numerical link specifications in charts 7.3a and 7.3b.
In terms of recall, the simple union approach employed by disjunction aims to include as many links as possible. However, due to the inclusive nature of disjunction, there is a higher likelihood of adding incorrect links, resulting in a reduction in precision.

7.1.4 Injective Complementation

By transitioning to injective complementation, we observed a minimal average improvement of 0.7% in precision, alongside a slight decrease in recall of 1.23%. Specifically, upon examining the charts, we notice that in two out of the seven link specifications, the precision value is marginally higher than that of the link keys, while the recall value is lower compared to the numerical link specification. For a more comprehensive analysis of the precise values associated with each method and to gain a clearer understanding of the overlapping points, I recommend referring to the appendix A, which contains the complete tables.
The slight increase in precision observed with injective complementation can be attributed to its emphasis on the links provided by the link keys.
The chart 7.3a illustrates that the link keys exhibit a notably high precision value. Moreover, injective complementation effectively integrates all the correct links not detected by the link keys but generated through the less restrictive condition of the numerical link specification in the most optimal manner.

7.1.5 Comparison between operators

Upon examining the common trends depicted in the charts, we observe a noticeable contrast in precision between the link keys and the relaxed link keys with injective complementation. However, in terms of recall, injective complementation yields a lower value compared to the numerical link specification.
Conversely, for disjunction, the presence of multiple links leads to a significant difference in precision when compared to the link keys.
As previously announced we look for correlations between the bar chart and the individual values of each link specification.
The bar chart clearly illustrates that link specifications n° 64 and n° 97 exhibit the largest disparity in the number of links between the two approaches. Based on this analysis, it becomes apparent that a larger difference in links can lead to a degradation in performance when using disjunction. However, it provides an opportunity to enhance precision by utilizing the injective complementation

operator.
Nonetheless, it is important to acknowledge that the improvements and deteriorations observed are minimal, as the relaxed link keys never exhibit a change greater than 1 percent. Therefore, it is crucial to further investigate this difference in links when combining multiple link specifications together.

In summary, the results indicate that injective complementation has the highest potential for improving performance since it increases precision and marginally reduces recall. On the other hand, disjunction, while having the potential to increase recall, also carries a significant risk of substantially reducing precision. It is important to note that neither operator demonstrated outstanding results, as the changes in precision and recall were less than or equal to 1 percent respect to the best approach. Therefore, it is advisable to proceed with the analysis of combining two different link specifications, under the assumption that this approach has the potential to significantly enhance performance.

7.2 OAEI 2020 Datasets

Now, we will replicate the analyses carried out previously to determine if the attained improvements of the first experiment can be replicated in this new task. The methodology employed remains unchanged, with the only addition being a data selection process to mitigate the computational complexity associated with these extensive datasets.

7.2.1 Selected Link specifications

I used Linkex to calculate the link keys. However, due to the large size of the datasets, I had to remove the properties composition. This means that the entities were compared directly based on the value of their properties.
The candidates in this experiment exhibited a unique behavior, there were only 32 completely independent link keys, but each had a high discriminability value and number of links. To select the link keys, I chose three completely independent ones and two additional ones that can be considered as children of the initial three in a lattice of link keys.
All 5 link keys have high discriminability values and a link count of more than 150, which expresses excellent coverage, so they are good candidates.

- *Independent link sets*
  - **Link specification n° 7**= {⟨ ⟩}, {⟨ ns5:wikiPageWikiLinkText,ns4:altLabel⟩}

  - **Link specification n° 25**= {⟨ ⟩}, {⟨ ns5:wikiPageWikiLinkText,ns1:name⟩}
  - **Link specification n° 30**= {⟨ ⟩}, {⟨ns5:wikiPageWikiLinkText, ns5:wikiPageWikiLinkText ⟩}

- *Dependent link sets*
  - **Link specification n° 81**= {⟨ ⟩}, {⟨ ns5:wikiPageWikiLinkText,ns1:name⟩, ⟨ ns5:wikiPageWikiLinkText,ns4:altLabel⟩, ⟨ ns5:wikiPageWikiLinkText,ns5:wikiPageWikiLinkText ⟩}
  - **Link specification n° 82**= {⟨ ⟩}, {⟨ ns5:wikiPageWikiLinkText,ns4:altLabel⟩, ⟨ ns5:wikiPageWikiLinkText,ns5:wikiPageWikiLinkText⟩, ⟨ ns5:wikiPageWikiLinkText,rdfs:label ⟩}

It is evident that, similar to the first task, the linking property used in this experiment is consistently the name or label of the entity. Despite the presence of many other properties that describe a character, none of them produced link keys with satisfactory discriminability.
Hence, we can conclude that the *name* property holds significant importance and serves as the primary property for entity linking. Therefore, we will continue to prioritize it in our analysis.
As for generating the corresponding numerical link specifications, I decided to employ the *Jaro-Winkler distance* once again, but this time with a higher threshold of *0.97*. The rationale behind this decision is that with a lower threshold, a significant disparity was observed between the links generated by the numerical link specifications and the link keys. This resulted in lower precision values for the numerical specifications, while recall values improved significantly.
However, our objective is to enhance both precision and recall or find a reasonable balance between them. By using a higher threshold, we aim to strike a balance between the two measures, as a lower threshold would have predominantly focused on maximizing recall.
Before computing the similarity measure I've applied the filters of translating lowercase characters and removing special characters.

7.2.2 Results

The goal of the experiment, is to confirm the superiority of the injective complementation operator and explore alternative methods beyond the two main interlinking approaches.
We will begin by examining the relaxed link key approach. However, to simplify calculations and considering the negative impact of disjunction observed in the *Movie* task, we will solely use the injective complementation operator.

Based on the results observed so far, the hypothesis is that using relaxed link keys may yield slight improvements in performance.
Before showing the results, we present a bar chart illustrating the number of links for each of the two interlinking approaches. This charts can be a valuable resource for later analysis, since we have a significantly larger datasets compared to the previous experiment, so the results may differ from what we have previously observed.

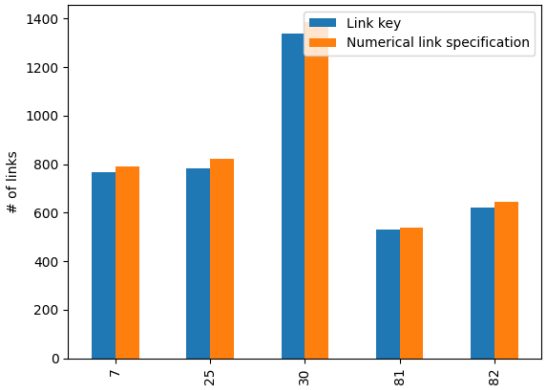
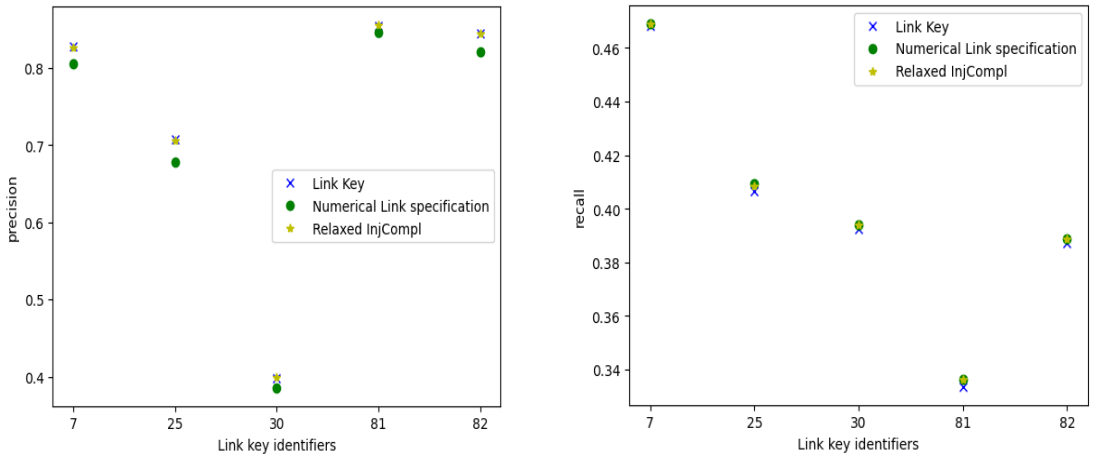


Figure 7.4: Number of links

The observed differences between the two approaches are negligible, with only two link specifications (n° 25 and n° 30) displaying a difference of approximately 4%. This suggests a potential for greater improvement in recall but an uncertain impact on precision.
Moving forward, we will now analyze the results using the provided table and charts.

	Precision		**Recall**	
	Link Key	**Sim. Approach**	**Link Key**	**Sim. Approach**
RLK InjCompl	+0.04%	+2.65%	+0.4%	-0.03%

Table 7.3: Average improvement of performance



(a) Precision (b) Recall

Figure 7.5: Charts of precision and recall for relaxed link keys

Upon examining the Table 7.3, it is evident that the results did not show any improvement or deterioration, with both precision and recall values remaining similar to the maximum values. However, it is worth noting that unlike the previous task, the recall value has remained virtually unchanged for the relaxed link key.
Although this is not a significant result in terms of performance improvement, it does confirm our decision to exclude the disjunction operator, as its inclusion would have hypothetically improved recall but adversely affected precision.

7.2.3 Numerical Link Specification

Regarding recall, as mentioned earlier, there is substantial overlap between the relaxed link key and the numerical link specification, except for one link specification where there is a slight reduction in value.
The behavior of this approach can be attributed to the datasets themselves and the numerical link specification. Specifically, since the two datasets are directly compared, and since the task contains very few links that cannot be found from the link key candidates used, the addition of one or two links to the link set leads to marginal increases in precision.
In fact, when comparing the precision values of the numerical link specification, we observe a notable increase of 2.65%. This can be confirmed by examining the precision chart 7.5a, where the yellow point consistently remains above the green point.
The recall chart 7.5b presents more challenges in interpretation due to extreme

overlapping of the points. This overlapping is evident in the mere average decrease of only 0.03% between the relaxed link keys and the numerical link specifications.

7.2.4 Link key

Similarly to the recall chart 7.5b, the difference in precision between the link keys and the relaxed link keys is so minimal that it is almost imperceptible. Therefore, for a more detailed analysis, we refer to the appendix containing the numerical tables.
Despite a slight increase of 0.04% in precision, it is not significant enough to be considered as improved performance. The value is too low to indicate any substantial enhancement.
Regarding recall, the difference between the positive links in the combination link sets and the link key link sets is minimal. This is evident from the marginal improvement of only 0.4% in recall. The reason could be found in the minimal difference between the link keys's recall and the numerical link specifications's recall of 0.43% in favor of numerical link specifications. Consequently, when combining the two approaches, even a small number of additional positive links are sufficient to achieve a recall comparable to that of the numerical link specification.

7.2.5 Comparison between approaches

Due to the reasons outlined earlier, it is challenging to establish a correlation between the number of links generated by the two approaches and the performance of relaxed link keys. Additionally, the link specifications with a 4% difference of links between the link key link set and the combination link set, do not demonstrate a notable increase in correct links compared to the other link specifications, indicating a similar behavior among them.

In conclusion, this task presents unique characteristics that limit the generation of relaxed link keys. However, we have further confirmed the suitability of the injective complementation operator for our objectives.
Let us now proceed to combine two different link specifications to try to overcome the task limitations.

7.3 Conclusion

Upon analyzing the results of the two tasks, we observed that the disjunction operator is comparatively less effective than injective complementation. The simple union approach utilized by disjunction, though aiming to maximize link inclusion for improved recall, tends to introduce a higher likelihood of incorrect links, thereby diminishing precision.
On the other hand, injective complementation yielded minimal enhancement in both tasks, slightly elevating precision but offsetting this gain with a corresponding reduction in recall.
Consequently, our findings highlight the inadequacy of this method in synergizing the two approaches. For this reason, our forthcoming experiment will involve exploring the fusion of diverse link specifications.

8 Second Experiment

The idea of the second experiment is to combine different link specifications with each other in order not only to overcome the limitations of link key link discovery, but also with the aim of combining two link specifications with totally different links and get a more complete and more accurate link set.
We will present the results, as in the previous experiment, dividing them into two sections one for each task.

8.1 Movie Datasets

To further advance the analysis, we will proceed with combining multiple link specifications with the objective of obtaining an enhanced link set and test the superiority of the injective complementation operator over disjunction.
The selection of pairs for combination was conducted randomly, as all the link specifications are independent of each other.
The forthcoming results will present combinations of two link specifications utilizing both operators. For the sake of thorough analysis, we will not only combine a link key with a numerical link specification but also the two link keys. Specifically, in the former case, we will select the link key that yielded the highest precision since, as explained earlier, injective complementation gives precedence to the first link specification.
The hypothesis posits that the combination of the link key and numerical link specification with injective complementation will offer the highest precision. Conversely, in terms of recall, it is anticipated that disjunction will yield the highest value.
Before unveiling the results, we will provide a concise legend summarizing the approaches employed and the specific link specifications utilized.

- *Link-Link Disjunction* = disjunction of two different link keys
- *Link-Link InjCompl* = combination of two different link keys using the injective complementation operator
- *Link-Sim Disjunction* = disjunction of a link key and a numerical link specification
- *Link-Sim InjCompl* = combination of a link key and a numerical link specification using the injective complementation operator

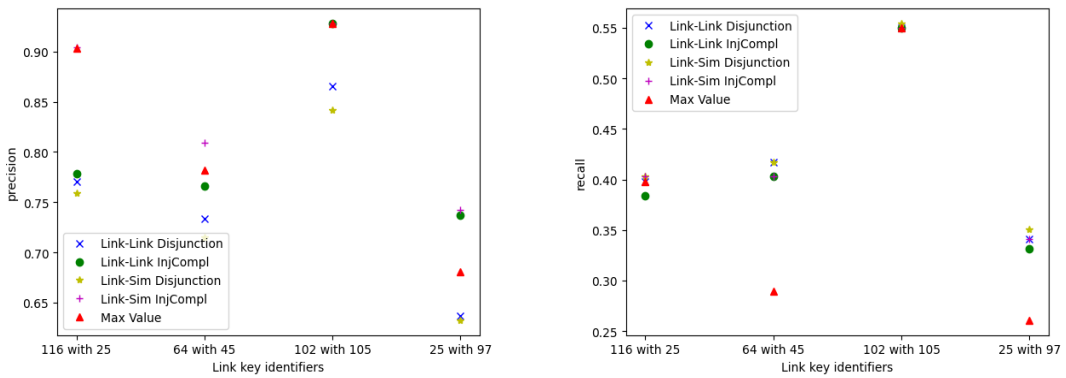
The pairs of link specifications, presented in section 7.1.1, employed for the combinations are as follows:

- *Link specification n° 116 WITH link specification n° 25*
- *Link specification n° 64 WITH link specification n° 45*
- *Link specification n° 102 WITH link specification n° 105*
- *Link specification n° 25 WITH link specification n° 97*

In the table below we will use the terms *First LS* and *Second LS* to express the two link specification on which we are comparing the values.

	Precision		**Recall**	
	First LS	**Second LS**	**First LS**	**Second LS**
Link-Link Disjunction	-8.52%	+4.24%	+24.37%	**+63.48%**
Link-Link InjCompl	-1.92%	+11.37%	+21.19%	+59.68%
Link-Sim Disjunction	-10.23%	+5.22%	**+26%**	+61.66%
Link-Sim InjCompl	**+3.15%**	**+20.97%**	+23.43%	+59.03%

Table 8.1: Average improvement of performance



(a) Precision (b) Recall

Figure 8.1: Charts of precision and recall for two link specifications combined

8.1.1 Disjunction

Table 8.1 reveals that all four methods show significant enhancements in recall, with none displaying an increase of less than 21%. As anticipated, disjunction

yielded the highest values as it incorporates a larger number of links into its link sets compared to injective complementation.
Conversely, the operator demonstrated a significant deterioration in precision, since the addition of new links belonging to the second link set led to a substantial decrease in precision.
By examining both charts and the table, it becomes evident that *Link-Link Disjunction* demonstrates superior precision compared to *Link-Sim Disjunction*, whereas the second approach exhibits better recall.
However, it should be noted that despite the low precision values for both approaches in comparison to the maximum value, recall takes precedence. Consequently, in this particular experiment, the combination of two link keys with disjunction proved to be less effective than combining a link key and a numerical link specification using the same operator.

8.1.2 Injective Complementation

Shifting our focus to injective complementation, the only approach that exhibited a slight improvement of 3% in precision was the combination of a link key and a numerical link specification with the injective complementation operator.
As predicted, both combination approaches involving injective complementation exhibit the highest precision values, thanks to the inherent behavior of the operator itself.
Continuing from the previous paragraph, we now compare the two approaches, namely *Link-Link InjCompl* and *Link-Sim InjCompl*. It becomes evident that the latter yields superior values for both measures, thus reaffirming that combining a link key with a numerical specification is more effective than combining two link keys.
To explain this observation in terms of recall, it is clear that the link set of the numerical link specification is more extensive than its corresponding link key, which increases the likelihood of achieving higher recall.
In terms of precision, however, injective complementation introduces additional links to the link set of the Link-Sim InjCompl approach. In our specific case, more positive links are added than negative links, leading to an improvement in precision.

8.1.3 Comparison between operators

In addition to the values presented for each method, the charts 8.1a and 8.1b now include the maximum measure between the two link specifications that were present prior to the combination (red triangle). This inclusion provides further clarity regarding the results observed in the table.

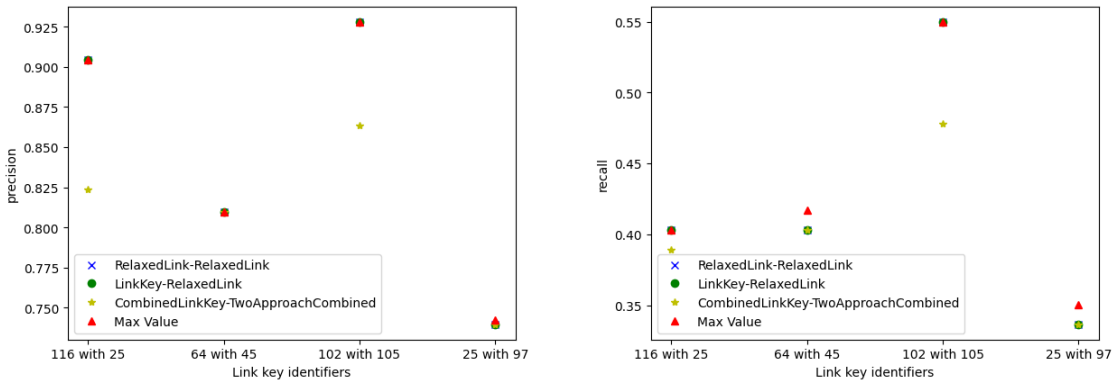
Upon examination, it becomes evident that only two out of the four combinations generated significant maximum results for both measures. Specifically, these combinations involve link specification n° 64 with link specification n° 45 and link specification n° 25 with link specification n° 97. These particular combinations yielded a satisfactory average increase in performance.
From these findings, we can conclude that the overall approach is productive. However, the specific choice of link specifications has a notable impact on the magnitude of improvements achieved. As our selection of pairs was random, we cannot outline a definitive protocol for choosing the most optimal combination of link specifications.
To determine which approach generally performed better, we can exclude the "Link-Link InjCompl" method as its counterpart, "Link-Sim InjCompl," produced superior results for both measures. Furthermore, the "Link-Link Disjunction" method displayed a lower average increase in recall compared to its counterpart, "Link-Sim Disjunction," but exhibited slightly higher precision. Nevertheless, since the deterioration was evident in both approaches, we can also discard the former.
Thus, we have established that combining two link keys may not yield the optimal results for either precision or disjunction. Henceforth, in line with the objective of this report, we will prioritize combining only link keys with numerical link specifications.
Finally, the best-performing method was the "Link-Sim InjCompl", as it improved both measures. Although it may not be the top choice for increasing recall, it still delivered a highly satisfactory outcome. Indeed, the improved recall achieved by the other approaches resulted in a negative alteration of precision, which is not the desired outcome.

The main finding of this study is that combining two different link specifications together yields significantly better performance than creating relaxed link keys.
Furthermore, we have demonstrated that the disjunction operator generally performs inferiorly to injective complementation and moreover the best combination approach identified is the combination of a link key and a numerical link specification, surpassing the performance of combining two link keys. This finding holds substantial importance as it aligns with the objective of our report to explore the combination of these two approaches.
For a more detailed analysis, the next chapter will involve further combinations of the approaches utilized thus far.

8.1.4 Combining previous approaches

In this chapter, building upon the previous chapter's promising results, we will explore the possibility of further combining the approaches employed so far.
The underlying concept involves integrating the previous combination approaches, thereby adding an additional layer of combination to enhance overall performance. Given the demonstrated superiority of the injective complementation operator, all approaches will be conducted using this operator.
The hypothesis posits that the results may exhibit slight improvements, albeit not exceeding 2%. This expectation arises from the belief that in previous experiments, we have already approached the upper limit of performance achievable through the combination of two link specifications.
Before showing the charts, I will outline the approaches to be analyzed in the following list.

- *RelaxedLink-RelaxedLink* = combination of two different "relaxed" link keys.
- *LinkKey-RelaxedLink* = combination of two different link specification one is a link key and the other one is a "relaxed" link key
- *CombinedLinkKey-TwoApproachCombined* = combination of the injective complementation of the two link key and the injective complementation of a link key and a numerical link specification.



(a) Precision (b) Recall

Figure 8.2: Charts of precision and recall for combined approaches

The results depicted in the charts refute the hypothesis put forth, as none of the three approaches yielded improvements. However, it is worth noting that the utilized approaches did not disappoint. In fact, two out of the three approaches matched the maximum values of precision and recall obtained thus far. These two methods are the combination of two relaxed link keys and the combination of a link key with a relaxed link key. The latter approach demonstrated slightly lower values compared to the maximum, particularly in terms of recall.
If we were to provide an explanation for the inability to overcome the limitations of previous experiments, we must consider the role of the similarity approach in precision improvement. In this case, instead of adding new links, we are creating new combined link specifications using relaxed link keys to enhance performance.
Regarding the last combination attempted in the experiment, the results were too varied to analyze its behavior effectively. However, a noticeable trend is a decrease in values compared to the maximum calculated. This can be attributed to the fact that one element of the injective complementation is the combination of the two link keys, which, as observed in 8.1, does not perform as well.
These factors contribute to the challenges in surpassing the previous limitations and achieving further improvements.
While the experiment can be regarded as positive for matching the best method identified thus far, it is essential to acknowledge that the complexity of these two methods is higher compared to the methods employed in previous experiments.
In conclusion the experiment showed us how it is not possible to improve the measures calculated in second experiment, and that moreover the combination of link key and numerical link specification is not only the most efficient in terms of precision, but also as computational complexity, as it requires fewer steps (combinations) to generate the same result presented in this subsection.

8.2 OAEI 2020 Datasets

We proceed to perform the same experiment on another task in order to check whether the results match the previous task, and thus be able to give general conclusions about the second experiment.
Since three out of five link specifications are totally independent, we will combine them with each other, and finally we will also combine the two "child" links specifications with each other. We therefore present the pairs of link specifications, presented in section 7.2.1, to be combined:

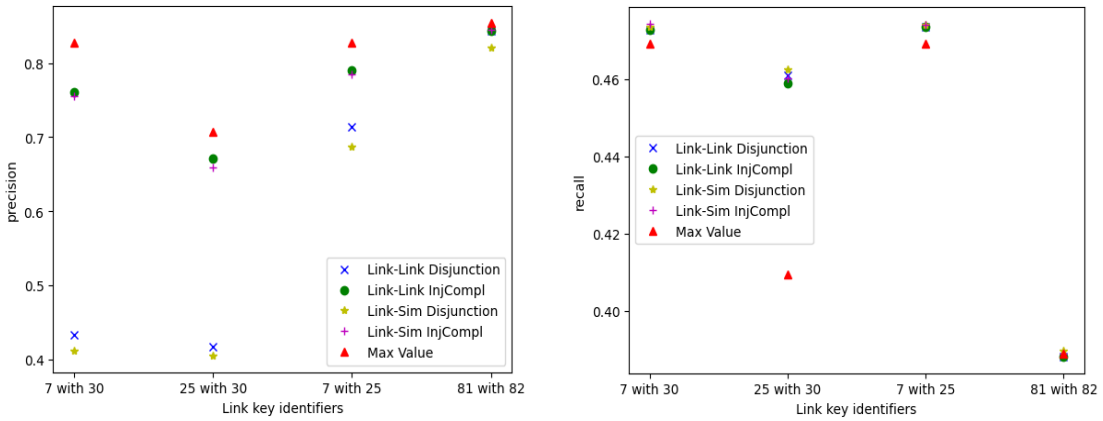
- *Link specification n°7 WITH link specification n°30*
- *Link specification n°25 WITH link specification n°30*

- *Link specification n°7 WITH link specification n°25*
- *Link specification n°81 WITH link specification n°82*

As previously stated, the four approaches will be presented again in this experiment, but with different assumptions.
Firstly, concerning the disjunction-based methods, it is expected that they will exhibit significantly lower precision compared to the approaches utilizing injective complementation. However, their behavior regarding recall may show the opposite.
Additionally, considering the findings from the relaxed link key results, it is possible that the combination of two link keys performs better than the combination of a link key and a numerical link specification. My hypothesis is that both approaches could be quite similar to each other.

	Precision		**Recall**	
	First LS	**Second LS**	**First LS**	**Second LS**
Link-Link Disjunction	-25.88%	+3.64%	+7.96%	**+13.64%**
Link-Link InjCompl	**-4.4%**	+42.79%	+7.86%	+13.01%
Link-Sim Disjunction	-28.3%	+3.33%	**+8.22%**	+13.38%
Link-Sim InjCompl	-5.5%	**+46.44%**	+8.08%	+13.26%

Table 8.2: Average improvement of performance



(a) Precision (b) Recall

Figure 8.3: Charts of precision and recall for different link specifications combined

8.2.1 Disjunction

As anticipated, both approaches (*Link-Link Disjunction* and *Link-Sim Disjunction*) experienced a significant decline in precision due to the large number of links in each link set.
Thus, performing a direct union of the sets resulted in a considerable deterioration of precision, just as predicted. However, I expected to observe a more pronounced improvement in recall.
Surprisingly, both approaches demonstrated a similar increase in recall of 7.96% and 8.22%. This suggests the fact that the link sets of the link key and numerical link specification, for the same link specification, are similar.
Consequently, combining the link sets of both approaches does not yield significantly different results. However, in our specific case, the numerical link specifications's link sets consistently have a slightly higher number of links.
As a result, the Link-Sim Disjunction method achieves the highest recall value among all four approaches.
One possible explanation for this similarity could be that, although expressed using different properties (e.g., ns1:name and ns4:altLabel), the interlinking property values for the link specifications are actually the same.
Moving forward, both approaches demonstrate a decrease in precision of over 25% when compared to the maximum value. To elucidate this finding, we can refer back to the earlier explanation, since the link sets of the two link specifications are highly similar and contain the same correct links, combining them only introduces more negative links, thus worsening the recall.
An important aspect to highlight is the significantly low precision values obtained by the disjunction-based combinations involving the link specification n° 30. This is likely due to the fact that this particular link specification has a much lower precision compared to the link specification it is paired with. Hence, the selection of link specification n° 30, while independent of the others, was not optimal for improving performance.

8.2.2 Injective Complementation

As hypothesis earlier, the combination approaches involving injective complementation did not result in an increase in precision, but have produced better value than disjunction. In fact, they exhibited a discrete decrease of 4.4 percent and 5.5 percent, respectively, compared to the best value.
The explanation lies in the composition of the combined pairs of link sets. As previously mentioned, regarding the similarity of recall values for the disjunction operator, the various link sets share a significant number of common values, particularly positive links. This explains why combining two link sets primarily

adds negative links, consequently leading to a decrease in precision.
In terms of recall, the two different approaches obtained similar values of 7.86% and 8.08%. This further emphasizes that all four combination approaches yielded very comparable recall values. The underlying reason for this similarity can once again be attributed to the resemblance of link sets between link keys and numerical link specifications.

8.2.3 Comparison between operators

The most crucial finding is that the two methods combining link keys outperform the other approach. This aligns with our initial assumption, as we observed in the previous chapter that the numerical link specification generated only a few additional correct links compared to its corresponding link key.
Thus, when we combine two link keys, we achieve a recall that is quite similar to that obtained by combining a link key with a numerical link specification. Moreover, by having fewer negative links, we can attain a higher precision value.

In conclusion, unlike the first experiment, we were unable to achieve a more efficient method in this case. The charts indeed support this observation. In the precision chart 8.3a, it is evident that no value surpasses the red triangle, which represents the maximum value achieved by the two basic approaches.
Conversely, in the recall chart 8.3b, all values exceed the maximum.
Furthermore, we observed that, for this specific task, the most effective approach for combination is actually the one involving two link keys. This is likely because the link keys are already responsible for generating the majority of correct links, while the numerical link specification primarily generates new links, but with a higher likelihood of being incorrect.

8.2.4 Adding new classes

I initially chose to focus on a single class to reduce computational complexity. However, we encountered complications due to limitations within the datasets itself, which prevented us from effectively combining the two approaches.
To address this, we have decided to expand our datasets by adding more classes, allowing us to analyze a greater number of entities. Moreover, this expansion presents an opportunity to discover new properties that can generate additional link keys for use in the combination process.
I have added five new classes, resulting in a tripling of the number of links within the datasets. However, the link key candidates remained the same, only with a greater number of links. As expected, the recall has increased, but the precision

values have remained largely unchanged for each approach and specific link.
I will refrain from presenting the data again to avoid repetition of the same charts and tables.

In comparing the two experiments conducted thus far, we can not deny that the combination of the two approaches of interlinking is more performant than the approaches themselves, however we have been able to see how the dataset type, the properties utilized, and the dataset's capacity itself, is fundamental for our purpose.
To strive towards a new method that combines the two approaches, our next step in the subsequent chapter will involve replacing the property-based numerical link specification with a document similarity approach.

8.3 Conclusions

Analyzing the experimental outcomes for both tasks reveals that, in the context of Movie datasets, combining link keys with numerical link specifications outperforms the combination of two link keys. Notably, the injective complementation operator emerges as the most effective strategy.
Conversely, the OAEI 2020 datasets exhibit a distinct pattern. Contrary to the initial task, the same experimental setup not only fails to enhance overall performance but also yields results conflicting with the earlier findings. Here, the fusion of two link keys slightly outperforms the combination of a link key and a numerical link specification.
These disparities can be attributed to the pivotal role of dataset characteristics in shaping combination performance. Varying entity types and their interconnections can favor or impede particular approach combinations, ultimately impacting the global performance.
In fact, depending on the datasets we may generate link specifications that are very similar to each other, making it unnecessary to combine them.
The next step was to use a new similarity approach that does not focus on finding links by comparing property values, but rather seeks equality between two entities by comparing them. This could allow us to have greater interlinking capabilities and thus have an additional opportunity for performance improvement.

9 Third Experiment

For this final experiment, we will dedicate a bigger chapter due to its unique methodology and strategy, setting it apart from the previous two experiments. This includes discussing the approach we employ for extracting links in the similarity method, as well as the specific combination operations we utilize.
Subsequently, we will explore into the theoretical framework behind this experiment and outline the step-by-step process that will lead us to the results for analysis.

Rather than relying on string similarity, we aim to explore an alternative form of similarity.
The fundamental concept behind this experiment revolves around replacing the link sets obtained through property-based numerical link specifications with a link set generated through document similarity analysis between the two datasets. This approach involves converting each entity into a text document containing its specific property values and subsequently establishing a link with an entity in the other dataset that exhibits an high similarity value.

9.1 Document similarity

Document similarity [25] involves assessing the degree of resemblance or overlap between two or more documents, considering factors such as their content, structure, and context. Its purpose is to quantify the level of similarity between documents.
To achieve this, mathematical models or algorithms are commonly used to extract relevant information from the documents and compare them.
They analyze factors like word frequency, term co-occurrence, semantic meaning, or syntactic structure to determine document similarity.
The outcome of a document similarity analysis is typically a similarity score or matrix, where higher scores indicate a greater level of similarity. This score serves various purposes, such as ranking and retrieving documents based on their relevance to a given query or identifying similar documents within a large collection.

In our scenario, the ***TF.IDF*** [4] algorithm will be employed for calculating

term weights in the documents during the embedding phase. Subsequently, ***cosine similarity*** [24] will be utilized to compare these weights and ascertain the similarity between the documents during the computation phase. In practical terms, TF.IDF are utilized to represent documents as vectors in the term space, enabling the calculation of cosine similarity between these vectors.

9.2 TF.IDF

TF.IDF, which means Term Frequency-Inverse Document Frequency, is a numerical statistic used to determine the significance of a term within a document or a collection of documents [4].
TF.IDF takes into account two primary factors: **term frequency** (*TF*) and **inverse document frequency** (*IDF*).

- *Term frequency* (TF) measures how frequently a term appears in a document. It reflects the significance of a term within a specific document and is calculated by dividing the number of times a term appears in a document by the total number of terms in that document. It provides a local indication of the significance of a term within a document.

- *Inverse document frequency* (IDF) evaluates the rarity or uniqueness of a term across a collection of documents. It is computed by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that ratio. IDF helps to identify terms that are more distinctive or informative across the entire document collection.

The TF.IDF score is obtained by multiplying the term frequency (TF) by the inverse document frequency (IDF) for each term in a document. The resulting score indicates the relative importance of a term in the document and is higher when the term appears frequently in the document but is relatively rare in the overall document collection.

Definition 8. (TF-IDF) Given a term x and a document y the tf.idf value is expressed as follow:

$$w_{x,y} = tf_{x,y} * log\frac{N}{df_x} \tag{9.1}$$

where $tf_{x,y}$ represent the frequency of x in y, N is the total number of documents and df_x is the number of documents that contain x

In summary, TF.IDF provides a way to measure the importance of terms in

documents by considering both their frequency within a document and their rarity across a collection of documents. It enables the identification of key terms and plays a crucial role in various text analysis applications.

9.3 Cosine similarity

Cosine similarity quantifies the degree of similarity between two non-zero vectors in a vector space by calculating the cosine of the angle between them. The resulting value, ranging from -1 to 1, indicates the extent of similarity or dissimilarity between the vectors.
To grasp the concept of cosine similarity, let's imagine a vector space containing two vectors, A and B. These vectors can represent various entities with numerical attributes, such as documents, images, or user preferences. Each dimension of the vector corresponds to a specific feature or characteristic associated with the entity.
Cosine similarity is calculated using the dot product of the vectors and the magnitudes of the vectors.
Example 6. Given the two vectors A and B, the *cosine similarity* formula is:

$$cos(A, B) = (A * B)/(\|A\| * \|B\|) \tag{9.2}$$

By computing the dot product of vectors A and B and normalizing it based on the magnitudes of the vectors, cosine similarity provides a measure of how closely the vectors align in direction and magnitude. A cosine similarity of 1 indicates strong similarity, 0 denotes no similarity (orthogonality), and -1 signifies high dissimilarity between the vectors.

9.4 Methodology

As our approach has changed, we will provide a revised explanation of the methodology employed in our latest experiment. Based on the definition of numerical link specification, this approach can be categorized as a *numerical link specification*. It involves applying a similarity measure to two entities from separate RDF datasets. However, there is a distinction in the methodology: instead of utilizing a threshold for all comparisons, the approach selects the top five values returned by the similarity measure and includes them in the link set. When it comes to extracting link keys, we have maintained our existing procedure of selecting the best link keys suggested by Linkex.

The key distinction lies in the generation of a link set for the similarity-based approach. Initially, we save the property values of each entity from both datasets into separate text documents, organizing them into two folders representing their respective datasets. Next, we utilize Elasticsearch to load the data onto a server, optimizing computational efficiency.
Elasticsearch [22] is an analysis software based on Lucene, employs Lucene's inverted index [36] for data storage and retrieval. It is possible to create programs that facilitates rapid and scalable indexing, searching, and analysis of large datasets, catering to unstructured, semi-structured, and structured data.
In our case, we create two indexes, one for each datasets, and upload the corresponding folders with documents.
Subsequently, we transform each entity of both datasets into a vector of *tf-idf* scores and we calculate the cosine similarity between each source entity and every target entity, selecting the *top five* entities with the highest values.
This was done to avoid assuming that the first generated link is always a positive link. Therefore, I increased the range to include the top 5 links, allowing for a broader consideration of potential connections, enabling a more comprehensive exploration of possible links while accounting for variations in similarity measurements.
After a step involving the retrieval of entity URIs, we generate our link set. From this set, we retrieve only the links where the entities appear exactly once in the entire set of links. To achieve this, starting from the first link, we search for all occurrences of both entities and remove them, this algorithm is called *stable marriage*.
In conclusion, we will merge the two approaches together. However, unlike before, we will not combine multiple link specifications, instead, we will integrate link keys with an extensive link set.
This is because the link set represents the similarity between the two datasets, rather than reflecting the similarity between entities based on properties comparisons.
It's important to acknowledge that enhancing precision may present challenges. Therefore, we will prioritize increasing recall while recognizing the potential for improvements in precision as well.

9.5 Results of Random Datasets

The analysis will differ from previous experiments, as we possess a comprehensive link set encompassing both datasets, for this reason we will merge the extracted link keys with this global link set.
Given the significant disparity in the number of links between the two sets, we

will not employ the disjunction operator solely to enhance recall. Instead, we will aim for balanced performance by utilizing injective complementation giving priority to the link keys.
Before making any assumptions about the expected results, we will present the selected link keys and conduct a separate analysis of the link sets derived from the similarity approach.

9.5.1 Link keys

The candidate generation process involved a property composition value of 3. The chosen candidates exhibit optimal discriminability, with all having a discriminability score greater than 0.92. Aside from link n° 1108, the other candidates display a moderate number of links, approximately 100.
Regarding the link specifications, it is observed that all five candidates don't possess a property composition. This is attributed to the random nature of the dataset, making it highly improbable to obtain links with multiple property compositions.

These are the candidates:

- **Link specification n°594**= {⟨ ⟩}, {⟨ dbo:deathDate, wdt:P570 ⟩}
- **Link specification n°601**= {⟨ ⟩}, {⟨ dbo:birthDate, wdt:P569 ⟩}
- **Link specification n°862**= {⟨ ⟩}, {⟨ foaf:homepage, wdt:P856 ⟩}
- **Link specification n°1108**= {⟨ ⟩}, {⟨ foaf:name, rdfs:label ⟩}
- **Link specification n°1148**= {⟨ ⟩}, {⟨ foaf:name, wdt:P373 ⟩}

Before proceeding to the next stage of combining approaches, it is important to highlight an additional characteristic of the link keys: their performance values.
Upon examination, all link specifications demonstrate outstanding precision, with each exceeding 0.92. However, due to the relatively small size of some link sets, the recall values are low, even below 4%.
An exception to this pattern is link key n° 1108, which stands out with its 637 links. Despite the high precision of 0.972, it achieves a recall of 0.524.
This remarkable recall value indicates that the link key candidate already serves as an excellent representation of the intersection between the two datasets.

9.5.2 Similarity based approach

I present the performance of the link set obtained from the document similarity.

$$precision = 0.958 \qquad recall = 0.667$$

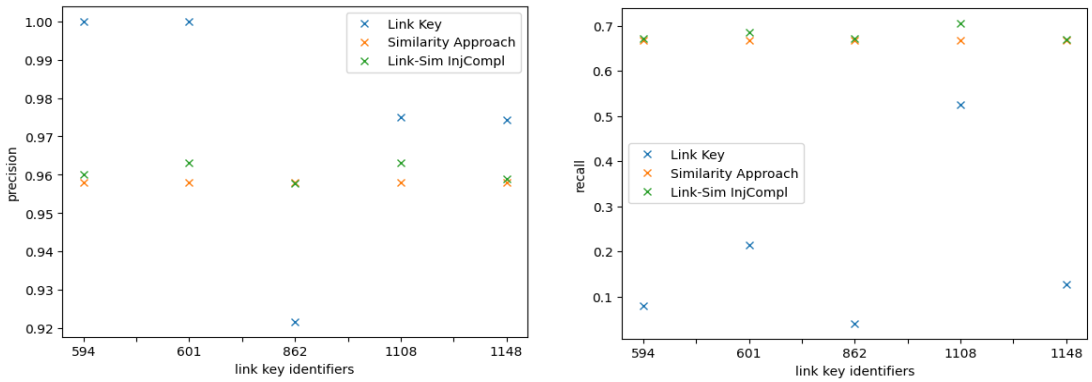
While we have achieved a globally representative link set, the recall value falls short of expectations, capturing only 66% of the available positive links.
This can be attributed to the filtering process where we specifically chose to include only links with a discriminability equal to 1. As a result, a significant number of positive links were removed, leading to a 96% decrease in total links and a decrease in recall from 0.74 to 0.667.
Considering this, combining the link set with the link keys is expected to enhance the recall value. However, given that the precision values of the link keys are exceptionally high, with 2 out of 5 candidates achieving a precision of 1, it is clear that we might not further improve precision.
In fact, considering the high precision of the similarity approach, it is possible that we might obtain a negative average value for precision.
Therefore, while the combination has the potential to increase recall, it may not result in a substantial improvement in overall performance due to the precision-oriented nature of the key link candidates.

9.5.3 Combination's result

As previously mentioned, I will combine the two approaches using only injective complementation, as utilizing the disjunction operator would significantly diminish the high precision of the link keys.
For this experiment, my hypotheses are as follows: we can expect a probable decrease in precision compared to the link keys, as it is unlikely to be improved due to the substantial difference in links between the two link sets. On the other hand, we anticipate an increase in recall. Without further ado, let's examine the calculated table and charts.

	Precision		**Recall**	
	Link Key	**Sim. Approach**	**Link Key**	**Sim. Approach**
Link-Sim InjComp	-1.31%	**+0.3%**	**+603%**	**+2.8%**

Table 9.1: Average improvement of performance



(a) Precision (b) Recall

Figure 9.1: Charts of precision and recall

9.5.4 Link Key

Upon analyzing the Table 9.1, it is evident that a decrease in link key's precision did occur, meeting the hypothesis made earlier.
The reason behind this can be attributed to the fact that the similarity approach link set was generated with a discriminability of 1, similar to the link sets of the link keys. On average, the similarity approach link set contains approximately three times more distinct entities than the link key link sets, as a result, a considerable number of links, both positive and negative, are added to the similarity approach side, leading to a decrease in precision.
Due to the significantly larger number of entities present in the link set of the similarity approach compared to the link set of the link keys, the increase in recall is notably substantial, 603%. Overall, this approach distinctly favors improvement over link keys, as the combination significantly enhances recall while experiencing minimal loss in precision.

9.5.5 Similarity approach

When comparing the precision values of the relaxed link keys with those of the similarity approach, we observe an average improvement of 0.3%. This improvement arises because the precision of the similarity approach is quite high, 0.958. In addition, the combination produced a 2.8% increase in recall; in the chart 9.1b, we observed an improvement in recall from link keys n° 594 and n° 862, which initially had a low recall, achieving slightly higher recall than similarity approach's recall after the combination.

Additionally, it is noteworthy that even link key n° 1148, with a recall of approximately 17%, obtained a recall very similar to the previous two link keys after the combination.
The combination actually improved performance compared to the similarity approach, however, the percentages are so low that we actually did not get added value to the similarity approach.

9.5.6 Comparison between approaches

The charts, reveals that all five combination values are remarkably similar, despite variations in the values of each individual link key.
To comprehend this phenomenon, I observed that the similarity approach, through injective complementation, essentially incorporates in the other candidates the same links present in link set n° 1108 (the largest one). Additional links are subsequently added, resulting in slightly different precision and recall values among the various link key candidates.
Upon analyzing the data, I noticed that the link keys with the highest number of links tend to perform better in both measures. These link keys exhibit a higher initial recall, which is further enhanced through the combination process.
In terms of precision, a larger number of links in the link key link set reduces the likelihood of adding links from the similarity approach, which, as we know, possesses lower precision compared to the five link keys.
To provide a more comprehensive analysis, I have included a bar chart below, comparing the number of links in the link key candidate and the number of links generated as a result of the combination.
This chart serves to further illustrate the relationship between link quantity and the outcomes of the combination process.

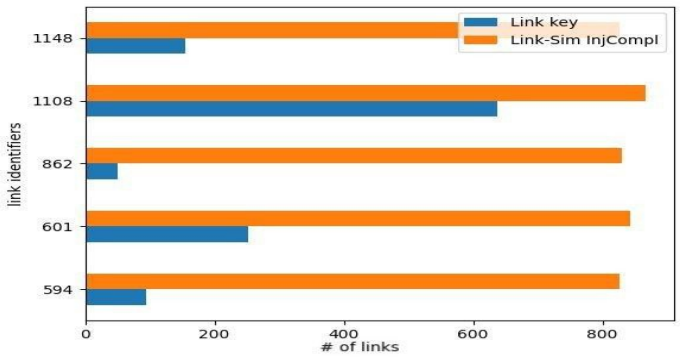


Figure 9.2: Number of links

In conclusion, this experiment demonstrates the possibility of utilizing different similarity approaches in combination with link keys. However, in this case, while an increase in recall was achieved, the decrease in precision was too significant to consider it an overall performance improvement; we can say that for this last experiment, the combination effectively combined the benefits of both approaches, but failed to significantly outperform them.
It is evident that the combination with injective complementation between a link set representing a single link specification and a link set representing the similarity between two datasets may not be effective, as it results in the addition of numerous links from the larger link set, shifting the priority towards the second link set in the formula 4.2.

10 Conclusion

This paper explores the utilization of both link keys and a similarity-based approach to establish connections between RDF datasets; the goal was to find an approach that merge link keys and numerical link specification, in order to combine the benefits of both and get a better interlinking approach of the two. Given the abundance of approaches utilizing similarity, we chose not to restrict ourselves to solely relying on property-based numerical link specification. Instead, we ventured to directly compare datasets using a document similarity method, specifically employing the tf-idf algorithm.
Initially, we regarded link keys as the most promising interlinking strategy. However, as we gained a deeper understanding of the effectiveness of similarity approaches, we have presented methods that combine these approaches through the use of two combination operators, namely disjunction and injective complementation.
Experimental evaluations demonstrate that among the two operators employed, injective complementation proves to be more suitable for combining the two approaches. It consistently achieves superior precision values and slightly lower recall values than disjunction, while still maintaining improved performance. On the other hand, disjunction primarily serves the purpose of increasing the recall of the combined link set.
We have also shown that combining the two approaches with each other is more effective than combining two link keys with each other.
However, we have observed that the effectiveness of such combination methods varies depending on the characteristics of the datasets being analyzed. Consequently, it can be concluded that the research did not yield optimal results since injective complementation obtained marginal improvements over the two approaches, so it can mitigate weaknesses of link key and similarity approach, although not entirely eliminate them.
Future research could focus on using new similarity approaches to combine with link keys, an example would be RDF2Vec, the idea is to represent RDF entities as dense numeric vectors in an embedding space. RDF2Vec generate the vector embeddings, and then compare the vectors to find similar entities. To generate vector embeddings with RDF2Vec, the approach uses RDF graph walking. Basically, random or controlled paths are created in the RDF graph, in which they move from one entity to another through relationships defined in RDF triples. These paths generate sequences of entities, which are then used to train the machine learning model, such as Word2Vec or Doc2Vec.

The research could move into the exploration of new compound operators, an hypothesis of a new operator is the conjunction of injective complementation of the first link specification with the second and injective complementation with the link specifications reversed. The idea is to use both link specifications by giving them priority in injective complementation and finally use the intersection of the two different results.
Another idea is to continue the research done by Atentia in 2019, which searches for the best way to extract automatic link key candidates using disjunction, replacing the latter operator with injective complementation.
So instead of looking for the best disjunction between link keys, we replace a link key with a numerical link specification and try to exploit the same method but with injective complementation.

A Appendices

A.1 Movie Task

A.1.1 Relaxed Link Key Levenshtein

	Precision	**Recall**
Link Keys n25	0.6765	0.218
Similarity Links n25	0.6765	0.218
Relaxed Disjunction	0.6765	0.218
Relaxed InjCompl	0.6765	0.218
Link Keys n45	0.75641	0.27962
Similarity Links n45	0.75641	0.27962
Relaxed Disjunction	0.75641	0.27962
Relaxed InjCompl	0.75641	0.27962
Link Keys n64	0.7763	0.2796
Similarity Links n64	0.7763	0.2796
Relaxed Disjunction	0.7763	0.2796
Relaxed InjCompl	0.7763	0.2796
Link Keys n97	0.6538	0.2417
Similarity Links n97	0.6625	0.25118
Relaxed Disjunction	0.6625	0.25118
Relaxed InjCompl	0.6625	0.25118
Link Keys n102	0.928	0.54976
Similarity Links n102	0.928	0.54976
Relaxed Disjunction	0.928	0.54976
Relaxed InjCompl	0.928	0.54976
Link Keys n105	0.7821	0.28909
Similarity Links n105	0.7821	0.28909
Relaxed Disjunction	0.7821	0.28909
Relaxed InjCompl	0.7821	0.28909
Link Keys n116	0.9032	0.3981
Similarity Links n116	0.9032	0.3981
Relaxed Disjunction	0.9032	0.3981
Relaxed InjCompl	0.9032	0.3981

A.1.2 Relaxed Link Key Jaro-Winkler

	Precision	**Recall**
Link Keys n25	0.6765	0.218
Similarity Links n25	0.6666	**0.22749**
Relaxed Disjunction	0.6666	**0.22749**
Relaxed InjCompl	**0.6857**	**0.22749**
Link Keys n45	**0.75641**	0.27962
Similarity Links n45	0.72839	0.27962
Relaxed Disjunction	0.72839	0.27962
Relaxed InjCompl	**0.75641**	0.27962
Link Keys n64	0.7763	0.2796
Similarity Links n64	0.74117	**0.29857**
Relaxed Disjunction	0.74117	**0.29857**
Relaxed InjCompl	**0.78481**	0.2891
Link Keys n97	0.6538	0.2417
Similarity Links n97	0.65476	**0.26066**
Relaxed Disjunction	0.65476	**0.26066**
Relaxed InjCompl	**0.6666**	0.2559
Link Keys n102	**0.928**	0.54976
Similarity Links n102	0.9141	**0.5545**
Relaxed Disjunction	0.9141	**0.5545**
Relaxed InjCompl	**0.928**	0.54976
Link Keys n105	0.7821	0.28909
Similarity Links n105	0.7683	**0.2986**
Relaxed Disjunction	0.7683	**0.2986**
Relaxed InjCompl	**0.7848**	0.2938
Link Keys n116	0.9032	0.3981
Similarity Links n116	0.8866	**0.4076**
Relaxed Disjunction	0.8866	**0.4076**
Relaxed InjCompl	**0.9043**	0.4028

A.1.3 Combining Link Keys

	Precision	**Recall**
Link Specification n25 WITH Link Specification n116		
Link-Link-Disjunction	0.77064	0.39811
Link-Link-InjCompl	0.77885	0.38389
Sim-Link-Disjunction	0.7589	**0.4028**
Sim-Link-InjCompl	**0.9043**	**0.4028**
Link Specification n45 WITH Link Specification n64		
Link-Link-Disjunction	0.7333	**0.41706**
Link-Link-InjCompl	0.7658	0.40285
Sim-Link-Disjunction	0.7154	**0.4176**
Sim-Link-InjCompl	**0.8095**	0.4028
Link Specification n105 WITH Link Specification n102		
Link-Link-Disjunction	0.8657	**0.54976**
Link-Link-InjCompl	**0.928**	**0.54976**
Sim-Link-Disjunction	0.8417	0.5545
Sim-Link-InjCompl	**0.928**	**0.54976**
Link Specification n97 WITH Link Specification n25		
Link-Link-Disjunction	0.6372	0.3412
Link-Link-InjCompl	0.7368	0.3318
Sim-Link-Disjunction	0.6325	**0.3507**
Sim-Link-InjCompl	**0.7423**	0.3412

A.1.4 Comparison between operators

	Precision	**Recall**
Link Specification n25 WITH Link Specification n116		
RelaxedLink-RelaxedLink	**0.9043**	**0.4028**
LinkKey-RelaxedLink	**0.9043**	**0.4028**
CombinedLinkKey-TwoApproachCombined	0.8235	0.3981
Link Specification n45 WITH Link Specification n64		
RelaxedLink-RelaxedLink	**0.8095**	0.4028
LinkKey-RelaxedLink	**0.8095**	0.4028
CombinedLinkKey-TwoApproachCombined	**0.8095**	0.4028
Link Specification n105 WITH Link Specification n102		
RelaxedLink-RelaxedLink	**0.928**	**0.5497**
LinkKey-RelaxedLink	**0.928**	**0.5497**
CombinedLinkKey-TwoApproachCombined	0.8635	0.4781
Link Specification n97 WITH Link Specification n25		
RelaxedLink-RelaxedLink	**0.7396**	0.3365
LinkKey-RelaxedLink	**0.7396**	0.3365
CombinedLinkKey-TwoApproachCombined	**0.7396**	0.3365

A.2 OAEI 2020 Task

A.2.1 Relaxed Link Keys

	Precision	**Recall**
Link Keys n7	**0.8281**	0.4683
Similarity Links n7	0.8053	0.4691
Relaxed InjCompl.	0.8262	**0.4691**
Link Keys n25	**0.7068**	0.4065
Similarity Links n25	0.6778	**0.4094**
Relaxed InjCompl.	0.7061	0.4087
Link Keys n30	0.3986	0.3925
Similarity Links n30	0.3855	0.394
Relaxed InjCompl.	**0.3996**	**0.394**
Link Keys n81	0.8547	0.3336
Similarity Links n81	0.8463	**0.3365**
Relaxed InjCompl.	**0.8558**	**0.3365**
Link Keys n82	0.8443	0.3873
Similarity Links n82	0.8211	**0.3889**
Relaxed InjCompl.	**0.8448**	**0.3889**

A.2.2 Combining Link specifications

	Precision	Recall
Link Specification n7 WITH Link Specification n30		
Link-Link-Disjunction	0.4335	0.4728
Link-Link-InjCompl	**0.7607**	0.4728
Sim-Link-Disjunction	0.412	0.4735
Sim-Link-InjCompl	0.7555	**0.47435**
Link Specification n25 WITH Link Specification n30		
Link-Link-Disjunction	0.4173	0.461
Link-Link-InjCompl	**0.6713**	0.4588
Sim-Link-Disjunction	0.4049	**0.4624**
Sim-Link-InjCompl	0.6593	0.4602
Link Specification n7 WITH Link Specification n25		
Link-Link-Disjunction	0.7144	**0.4735**
Link-Link-InjCompl	**0.7909**	**0.4735**
Sim-Link-Disjunction	0.6873	0.4742
Sim-Link-InjCompl	0.7854	0.4742
Link Specification n81 WITH Link Specification n82		
Link-Link-Disjunction	0.8445	0.3881
Link-Link-InjCompl	0.8445	0.3881
Sim-Link-Disjunction	0.8214	**0.3895**
Sim-Link-InjCompl	**0.8448**	0.3889

A.3 Random Task

	Precision	Recall
Similarity Approach	0.958	0.667
Link Keys n594	**1.0**	0.0794
Relaxed InjCompl.	0.9601	**0.6706**
Link Keys n601	**1.0**	0.2137
Relaxed InjCompl.	0.9632	**0.6858**
Link Keys n862	0.9216	0.0397
Relaxed InjCompl.	**0.9578**	**0.6715**
Link Keys n1108	**0.9749**	0.5245
Relaxed InjCompl.	0.9631	**0.7052**
Link Keys n1148	**0.9742**	0.1275
Relaxed InjCompl.	0.9589	**0.6698**

Bibliography

[1] Nacira Abbas, Jérôme David, and Amedeo Napoli. "Linkex: A tool for link key discovery based on pattern structures". In: *ICFCA 2019-workshop on Applications and tools of formal concept analysis*. 2019, pp. 33–38.

[2] Manel Achichi et al. "Automatic key selection for data linking". In: *Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20*. Springer. 2016, pp. 3–18.

[3] Sareh Aghaei and Anna Fensel. "Finding Similar Entities across Knowledge Graphs". In: *Proceedings of the 7th International Conference on Advances in Computer Science and Information Technology, Vienna, Austria*. 2021, pp. 20–21.

[4] Akiko Aizawa. "An information-theoretic perspective of tf–idf measures". In: *Information Processing & Management* 39.1 (2003), pp. 45–65.

[5] Manuel Atencia, Jérôme David, and Jérôme Euzenat. "Data interlinking through robust linkkey extraction." In: *ECAI*. 2014, pp. 15–20.

[6] Manuel Atencia, Jérôme David, and Jérôme Euzenat. "On the relation between keys and link keys for data interlinking". In: *Semantic Web* 12.4 (2021), pp. 547–567.

[7] Manuel Atencia et al. *Relational concept analysis for circular link key extraction*. en. Deliverable. ELKER, 2021. 57 pp. URL: https://moex.inria.fr/files/reports/elker-1.1.pdf.

[8] Michael Azmy et al. "Matching entities across different knowledge graphs with graph embeddings". In: *arXiv preprint arXiv:1903.06607* (2019).

[9] Mustafa Al-Bakri et al. "Inferring same-as facts from linked data: an iterative import-by-query approach". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[10] Mustafa Al-Bakri et al. "Uncertainty-sensitive reasoning for inferring sameAs facts in linked data". In: *22nd european conference on artificial intelligence (ECAI)*. IOS press. 2016, pp. 698–706.

[11] Tim Berners-Lee. "Linked data-design issues". In: *http://www. w3. org/DesignIssues/Linked html* (2006).

[12] Jeremy J Carroll et al. "Jena: implementing the semantic web recommendations". In: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. 2004, pp. 74–83.

[13] Peter Christen and Peter Christen. *The data matching process*. Springer, 2012.

[14] Peter Christen and Karl Goiser. "Quality and complexity measures for data linkage and deduplication". In: *Quality measures in data mining* (2007), pp. 127–151.

[15] Jérôme David et al. "The alignment API 4.0". In: *Semantic web* 2.1 (2011), pp. 3–10.

[16] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[17] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*. Vol. 18. Springer, 2007.

[18] Houssameddine Farah, Danai Symeonidou, and Konstantin Todorov. "KeyRanker: Automatic RDF key ranking for data linking". In: *Proceedings of the Knowledge Capture Conference*. 2017, pp. 1–8.

[19] Alfio Ferrara, Andriy Nikolov, and François Scharffe. "Data linking for the semantic web". In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 7.1 (2011), pp. 46–76.

[20] Bernhard Ganter and Sergei O Kuznetsov. "Pattern structures and their projections". In: *Conceptual Structures: Broadening the Base: 9th International Conference on Conceptual Structures, ICCS 2001 Stanford, CA, USA, July 30–August 3, 2001 Proceedings 9*. Springer. 2001, pp. 129–142.

[21] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[22] Clinton Gormley and Zachary Tong. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.", 2015.

[23] Aidan Hogan et al. "Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora". In: *Journal of Web Semantics* 10 (2012), pp. 76–110.

[24] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. "Cosine similarity to determine similarity measure: Study case in online essay assessment". In: *2016 4th International Conference on Cyber and IT Service Management*. IEEE. 2016, pp. 1–6.

[25] Michael D Lee, Brandon Pincombe, and Matthew Welsh. "An empirical evaluation of models of text document similarity". In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 27. 27. 2005.

[26] Gonzalo Navarro. "A guided tour to approximate string matching". In: *ACM computing surveys (CSUR)* 33.1 (2001), pp. 31–88.

[27] Markus Nentwig et al. "A survey of current link discovery frameworks". In: *Semantic Web* 8.2 (2017), pp. 419–436.

[28] Axel-Cyrille Ngonga Ngomo and Sören Auer. "Limes-a time-efficient approach for large-scale link discovery on the web of data". In: *integration* 15.3 (2011).

[29] Axel-Cyrille Ngonga Ngomo et al. "Raven-active learning of link specifications". In: *Ontology Matching* 2011 (2011).

[30] Andriy Nikolov, Victoria Uren, and Enrico Motta. "KnoFuss: A Comprehensive Architecture for Knowledge Fusion". In: *Proceedings of the 4th International Conference on Knowledge Capture*. K-CAP '07. Whistler, BC, Canada: Association for Computing Machinery, 2007, pp. 185–186. ISBN: 9781595936431. DOI: 10.1145/1298406.1298446. URL: https://doi.org/10.1145/1298406.1298446.

[31] Petar Ristoski and Heiko Paulheim. "RDF2Vec: RDF Graph Embeddings for Data Mining". In: *The Semantic Web – ISWC 2016* (2016), pp. 498–514. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46523-4_30. URL: http://dx.doi.org/10.1007/978-3-319-46523-4_30.

[32] Fatiha Saıs, Nathalie Pernelle, and Marie-Christine Rousset. "L2r: A logical method for reference reconciliation". In: *Proc. AAAI*. 2007, pp. 329–334.

[33] Mohamed Ahmed Sherif, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. "W ombat–a generalization approach for automatic link discovery". In: *The Semantic Web: 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part I 14*. Springer. 2017, pp. 103–119.

[34] Danai Symeonidou et al. "Sakey: Scalable almost key discovery in RDF data". In: *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*. Springer. 2014, pp. 33–49.

[35] Dean Van Der Merwe, Sergei Obiedkov, and Derrick Kourie. "Addintent: A new incremental algorithm for constructing concept lattices". In: *Concept Lattices: Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004. Proceedings 2*. Springer. 2004, pp. 372–385.

[36] CJ Van Rijsbergen. "Information retrieval 2nd edition butterworths". In: *London available on internet* (1979).

[37] Julius Volz et al. "Silk-a link discovery framework for the web of data." In: *Ldow* 538 (2009), p. 53.

[38] Ian H Witten et al. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[39] William E Yancey. "Evaluating string comparator performance for record linkage". In: *Statistics* 5 (2005), p. 38.