Discovering Wikipedia Conventions using DBpedia Properties

Diego Torres, Hala Skaf-Molli, Pascal Molli, Alicia Díaz

Discovering Wikipedia Conventions using DBpedia Properties

Diego Torres[1], Hala Skaf-Molli[2], Pascal Molli[2], and Alicia Díaz[1]

[1] LIFIA, Fac. Informática, Universidad Nacional de La Plata,
1900 La Plata, Argentina,
{diego.torres, alicia.diaz}@lifia.info.unlp.edu.ar,
[2] Nantes University, LINA, 2, rue de la Houssiniere, 44322 Nantes, France,
{Hala.Skaf, Pascal.Molli}@univ-nantes.fr

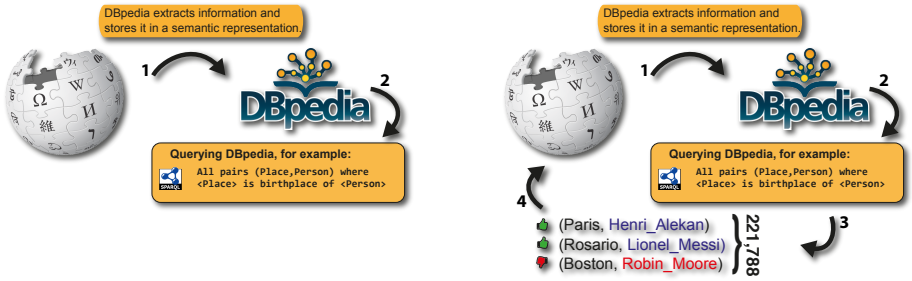
Abstract. Wikipedia is a public and universal encyclopedia where contributors edit articles collaboratively. Wikipedia infoboxes and categories have been used by semantic technologies to create DBpedia, a knowledge base that semantically describes Wikipedia content and makes it publicly available on the Web. Semantic descriptions of DBpedia can be exploited not only for data retrieval, but also for identifying missing *navigational paths* in Wikipedia. Existing approaches have demonstrated that missing navigational paths are useful for the Wikipedia community, but their injection has to respect the Wikipedia convention. In this paper, we present a collaborative recommender system approach named BlueFinder, to enhance Wikipedia content with DBpedia properties. BlueFinder implements a supervised learning algorithm to predict the Wikipedia conventions used to represent similar connected pairs of articles; these predictions are used to recommend the best convention(s) to connect disconnected articles. We report on an exhaustive evaluation that shows three remarkable elements: (1) The evidence of a relevant information gap between DBpedia and Wikipedia; (2) Behavior and accuracy of the BlueFinder algorithm; and (3) Differences in Wikipedia conventions according to the specificity of the involved articles. BlueFinder assists Wikipedia contributors to add missing relations between articles, and consequently, it improves Wikipedia content.

Keywords: Semantic Web, Social Web, DBpedia, Wikipedia, Collaborative Recommender Systems

1 Introduction

Semantic Web technologies facilitate search and navigation on the Web, while they can be additionally used to extract data from the Social Web, e.g., DBpedia is built with data extracted from Wikipedia [3] infoboxes and categories. On the other hand, knowledge managed and encoded by Semantic Web technologies can enhance data published in the Social Web; for instance, semantic annotations

[3] http://www.wikipedia.org



(a) Information flow from Wikipedia and DBpedia

(b) Complete cycle of information flow between Wikipedia and DBpedia

Fig. 1: Information flow between Social Web and Semantic Web

of data have been used to improve the Facebook graph search [1]. However, to the best of our knowledge, DBpedia [2] has not been exploited to improve Wikipedia. In this paper, we propose BlueFinder an approach to enhance the content of Wikipedia with data inferred in DBpedia.

Although Wikipedia is collaboratively edited by large user communities, Wikipedia links are not semantically defined, e.g., no types are attached to these links. In the context of Semantic Web, Wikipedia links are translated into properties in DBpedia, and they are semantically described using RDF vocabularies, i.e., DBpedia encodes semantics that is not represented in Wikipedia and provides a more expressive representation of Wikipedia links. Therefore, DBpedia allows for retrieving information that is not available in Wikipedia [3]. To illustrate, Listing 1.2 presents a SPARQL [4] query named $Q1$ to retrieve people and their born place[4] using `db-prop:birthplace`. Nevertheless, if $Q1$ is executed against the DBpedia endpoint [5], the answer includes more people than those obtained by navigating from the Wikipedia place article. The evaluation of query $Q1$ retrieves 409,812 (place, person) pairs from the DBpedia endpoint. Meanwhile, if we navigate from places to people in Wikipedia, we only obtain 221,788 connected pairs. Two Wikipedia articles are connected if a regular Wikipedia user can navigate from one article to another through a navigational path. A navigational path with a length larger than five is unreachable by a regular user [5–7]; so those articles are considered as disconnected. Thus, only 54 % of places in Wikipedia have a navigational path to those people who were born there. In this paper, we aim at adding missing navigational paths in Wikipedia, and enhancing Wikipedia content. Thus, we contribute to complete the virtuous cycle of information flow between Wikipedia and DBpedia as illustrated in Figure 1b.

To measure how important the gap between Wikipedia and DBpedia, we choose the most popular classes of DBpedia defined in [2], and the properties

[4] A place could be a Country, Province, City, or State.
[5] DBpedia of July 2013

with the highest number of triples. We call these properties *relevant properties*. Listing 1.1 shows the SPARQL query that retrieves the relevant properties that relate instances of the classes `db-o:Person` and `db-o:Place`.

```
prefix db–o:<http://dbpedia.org/ontology/>
select ?p (count(distinct ?o ) as ?count )
where {  ?s ?p ?o .
         ?s rdf:type  db–o:Person .
         ?o rdf:type  db–o:Place
 }
group by ?p
order by ?count
```

Listing 1.1: SPARQL query to retrieve relevant properties that relate instances of the classes `db-o:Person` and `db-o:Place`

We observe the same phenomenon when querying DBpedia using *relevant properties* of other classes, e.g., `db-o:Person`, `db-o:Place`, or `db-o:Work`, as shown in Table 1. The last two columns of the table provide the number of connected pairs obtained by a SPARQL query and the amount of disconnected pairs in Wikipedia for a specific property, respectively.

DBpedia Property	**from Class**	**to Class**	**# DBpedia connected pairs**	**# Wikipedia disconnected pairs**
$prop_1$: birthPlace	Place	Person	409,812	221,788
$prop_2$: deathPlace	Place	Person	108,148	69,737
$prop_3$: party	PoliticalParty	Person	31,371	15,636
$prop_4$: firstAppearance	Work	Person	1,701	142
$prop_5$: recordLabel	Company	Person	25,350	14,661
$prop_6$: associatedBand	MusicalWork	Person	365	73
$prop_7$: Company	Software	developer	14,788	2,329
$prop_8$: recordedIn	PopulatedPlace	MusicalWork	28,351	27,896
$prop_9$: debutstadium	Building	Athlete	595	393
$prop_{10}$: producer	Artist	MusicalWork	70,272	32,107
$prop_{11}$: training	Building	Artist	171	109
$prop_{12}$: previousWork	Album	MusicalWork	72,498	3,887
$prop_{13}$: recordLabel	Company	MusicalWork	118,028	75,329
$prop_{14}$: starring	Person	Film	164,073	42,584
$prop_{15}$: country	PopulatedPlace	Book	19,224	17,281
$prop_{16}$: city	PopulatedPlace	Educational Institution	34,061	8,681
$prop_{17}$: associatedBand	Band	MusicalArtist	24,846	4,100
$prop_{18}$: fromAlbum	Album	Single	18,439	1,268
$prop_{19}$: location	PopulatedPlace	Airport	10,049	2,660
$prop_{20}$: notableWork	Book	Person	1,510	73

Table 1: Results of 20 SPARQL queries for 20 properties and different classes.

Some connected resources in DBpedia are disconnected in their corresponding Wikipedia articles, i.e., resources can be navigated in DBpedia while it is not possible to navigate equivalent resources in Wikipedia. We call this missing navigational paths *information gap* between Wikipedia and DBpedia. To illustrate the gap, we analyzed $1,153,652$ connected pairs in DBpedia, and we found

that 540, 434 pairs were disconnected in Wikipedia. Consequently, the value of the information gap between Wikipedia and DBpedia is important. Figure 2 details the number of information gap between Wikipedia and DBpedia for the properties detailed in Table 1. In order to evaluate the usefulness of adding these navigational paths, we carry out a social evaluation [8]. In this evaluation, we have manually added missing navigational paths for 211 disconnected pairs and after one month, we analyzed how many navigational pairs were accepted and how many were rejected by the Wikipedia community. As detailed in [8], 90% of new navigational paths were accepted and 10% were rejected. Although the rejected navigational paths had respected the semantics of the relation, they were more general than those used by the community. For example, the proposed navigational path to connect *(Edinburgh, Charlie Aitken)*[6] with the DBpedia property *is birthplace of* was `Edinburgh / Category:Edinburgh / Category:People_from_Edinburgh / Charlie_Aitken`[7]. Wikipedia community argued that the category *People from Edinburgh* is too general and the more specific category *Sportspeople from Edinburgh* is a more appropriate link.

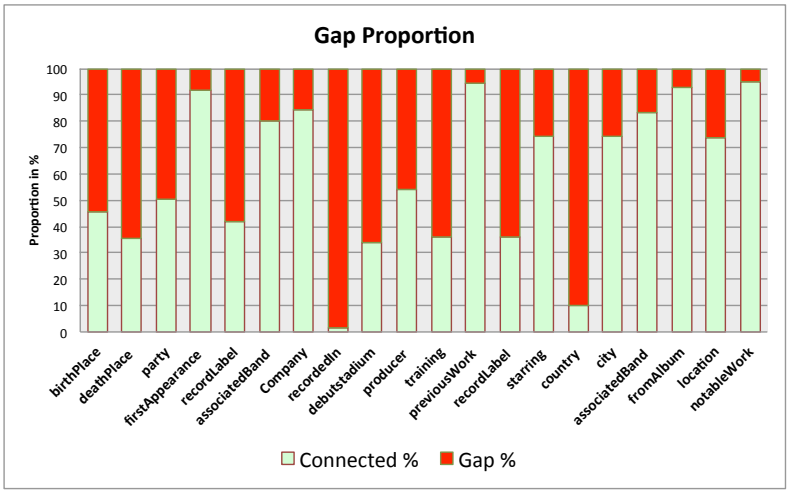


Fig. 2: Gap proportion for the twenty DBpedia properties of Table 1

For adding missing navigational paths, it is mandatory to study how the Wikipedia articles are connected respecting the Wikipedia conventions [8].

Wikipedia community has defined conventions that cover a wide diversity of topics: writing style, context of the articles and relations among articles.

[6] Charlie Aitken (footballer born 1942)

[7] It must be read as *from Edinburgh article, the user navigates through a link to the category Edinburgh then he or she navigates to People from Edinburgh category, and then to Charlie Aitken article".*

[8] `http://en.wikipedia.org/wiki/Wikipedia:Conventions`

Categories, *List of* pages, and *Navigation templates* are the conventions to design the navigation for one-to-many relations among Wikipedia articles. Additionally, the conventions could be defined by the community according to the specificity of the articles [8]. For example, the DBpedia property *is birthPlace of* that relates *Boston*[9] and *Tim Barsky*[10] is represented in Wikipedia by the navigational path `Boston / Category:Boston / Category:People_from_Boston / Tim_Barsky`. However, in case of Boston and *Donna Summer*[11], the same DBpedia property is represented by the navigational path `Boston / Category:Boston / Category: People_from_Boston / Category:Musicians_from_Boston / Donna_Summer`. Donna Summer is a musician from Boston and most of the Boston's musicians belong to the category Category:Musicians_from_Boston. These differences in the convention used to express the birthplace of a person trigger the following new question: *How to find the Wikipedia convention for a navigational path?*

In this paper, we address the problem of finding Wikipedia convention(s) that represent a DBpedia property between pairs of Wikipedia articles. Then, it is possible to connect pairs of Wikipedia articles that used to be disconnected. Consequently, Wikipedia content will be improved. For example, according to DBpedia (and also Wikipedia), Boston is the birthplace of Robin Moore. However, a navigational path from Boston to Robin Moore does not exist in Wikipedia but does exist in DBpedia. Therefore, a user could ask *Which is the convention to represent the "is birthPlace of" relation for (Boston, Robin Moore) in Wikipedia?*

We introduce BlueFinder, a collaborative recommender system that recommends navigational paths that represent a DBpedia property in Wikipedia. BlueFinder pays special attention to the specificity of the resource types in DBpedia. It learns from those similar pairs already connected by Wikipedia community and proposes a set of recommendations to connect a pair of disconnected articles. BlueFinder recommender system presented in this paper is an optimization of the previous version published in [9].

Summary of our contributions:

1. We measure the information gap between DBpedia and Wikipedia for a set of twenty representative DBpedia properties.
2. We re-design and propose several optimizations for BlueFinder algorithm.
3. We propose a new Semantic Pair Similarity Distance (SPSD) function to measure the similarity between pairs of related articles based on the DBpedia types description.
4. We conduct an empirical evaluation that measures Precision, Recall, F1, Hit-rate, and the confidence of BlueFinder recommendations over the twenty properties of DBpedia. The results demonstrate that BlueFinder is able to fix in average 89 % of the disconnected pairs with good accuracy and confidence.

[9] `http://en.wikipedia.org/wiki/Boston`
[10] `http://en.wikipedia.org/wiki/Tim_Barsky`
[11] `http://en.wikipedia.org/wiki/Donna_Summer`

The paper is organized as follows. Section 2 presents related works in the field of discovering and recommending links. Section 2 describes basic definitions used in this work. Section 4 describes the problem statement. Section 5 presents the BlueFinder approach and the algorithm description. An exhaustive evaluation is described in Section 6. Finally, conclusions and future works are presented in Section 7.

2 Related Work

Want et al. [10] introduce a *collaborative approach* to recommend categories to Wikipedia Articles. The approach consists of a two-step model. The first step collects similar articles to the uncategorized one in terms of incoming and out coming links, headings and templates. The second step lies on ranking the categories obtained by the related articles and selecting the best ranked. BlueFinder uses categorization but in another context. Categorization is used to express properties of DBpedia. Less related to our approach but in line with combining recommender systems and DBpedia, *MORE*[11] is a recommender system that uses DBpedia to recommend movies in a Facebook application. A Vector space model is used to compute semantic similarity. However, *MORE* uses DBpedia as a source data set to base the recommendations and not to improve Wikipedia. Panchenko et al. [12] propose to extract semantic relations between concepts in Wikipedia applying kNN algorithms called Serelex. Serelex receives a set of concepts and returns sets where articles are semantically related according to Wikipedia information and using Cosine and Gloss overlap distance functions. In addition to the lack of using DBpedia as semantic base, Serelex cannot describe the way that two concepts are related in Wikipedia according to a property. Singer et al.[13] compute semantic relatedness in Wikipedia by means of human navigational path. They analyze human navigation paths in order to detect semantic relatedness between Wikipedia articles. The main difference with BlueFinder is the absence of DBpedia as a support to describe semantic relatedness between Wikipedia concepts, as is exploited in our work with the Semantic Pair Similarity Distance. Di Noia et al. [14] introduce a strategy to find similarity among RDF resources in the Linked Open Data. Di Noia et al. present a content-based recommender system approach based on the Web of Data. As in BlueFinder, the similarity between resources is computed by means of semantic relationships. However, in [14] it is mandatory to discover the semantic relation among the resources and then to analyze a potential similarity. In BlueFinder, we already know that the two resources in each pair are related by the same property and then we only have to compare the types of description. Finally, the Di Noia approach is applied in the Web of Data world, and BlueFinder we complement and augment the information of the Social Web with information from the Web of Data.

Nunes et al. [15] introduce a recommender system to discover semantic relationships among resources from the Social Web. The work presents an approach to measure the connectivity between resources in a particular data set based on

semantic connectivity store and co-occurence-based measure. The first metric is based on graph relations among entities; the second one relies on an approximation of the number of existing Web pages that contain the same labels in their body. Although the approach of Nunes et. al. is closely related to BuleFinder, the main difference is the direction of the information flow. In Nunes et al work, the information of the Social Web is used to improve the Semantic Web and not in the opposite direction as in BlueFinder.

Other works [16, 17] aim at fixing missing direct links in Wikipedia, while [18] proposes an approach to complete Wikipedia infobox links with information extracted from Wikipedia by using Kylin. Works in [16–18] do not use Semantic Web features as BlueFinder does, moreover, BlueFinder fixes navigation paths rather than only direct links.

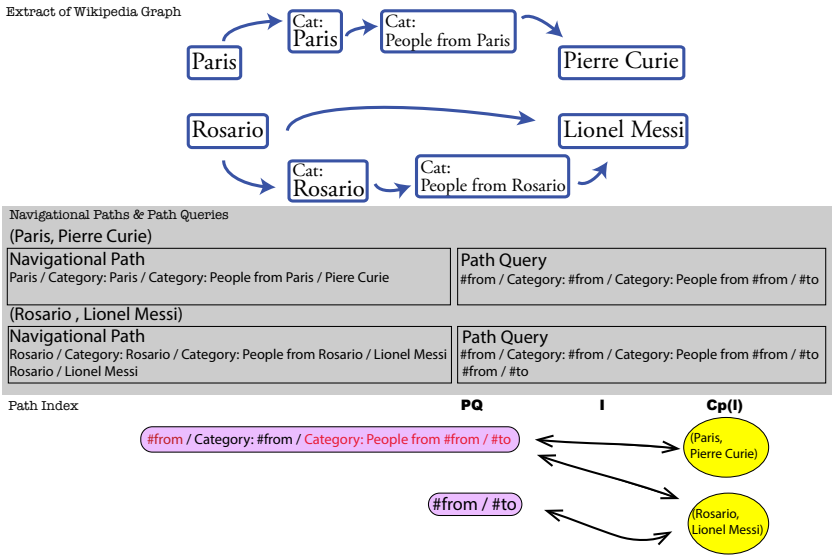


Fig. 3: Example of Wikipedia Graph, Navigational Path, Path Queries and Path Index

DBpedia has not been exploited before to improve the content of Wikipedia. BlueFinder proposes to enhance the content of Wikipedia with data inferred in DBpedia, and to complete the cycle of information flow between Wikipedia and DBpedia as illustrated in figure 1b.

3 Preliminaries Definitions

DBpedia Knowledge Base: DBpedia knowledge base is a set of RDF triples built from data extracted from Wikipedia [2]. This knowledge base has a set of

general properties and a set of infobox-specific properties, if the corresponding Wikipedia article contains an infobox. Each resource in DBpedia has types definition (*rdf:type*) coming from DBpedia ontology and Yago [19] ontology. The *foaf:isPrimaryTopicOf* property relates a DBpedia resource with its corresponding Wikipedia page. DBpedia provides a SPARQL endpoint to query DBpedia knowledge base. For example, the SPARQL query $Q1$ in Listing 1.2 retrieves from DBpedia the set of all pairs of Wikipedia pages $(from, to)$ that are related by the DBpedia property *birthplace*.

```
prefix db–o:<http://dbpedia.org/ontology/>
prefix db–p:<http://dbpedia.org/property/>
prefix foaf:<http://xmlns.com/foaf/0.1/>
select ?fr ?to
where  {  ?db_from rdf:type db–o:Person.
          ?db_to rdf:type db–o:Place .
          ?db_from db–p:birthplace ?db_to .
          ?db_from foaf:isPrimaryTopicOf ?fr .
          ?db_to foaf:isPrimaryTopicOf ?to
        }
```

Listing 1.2: Q1: DBpedia query for birthplace property.

For each property in Table 1, we define a corresponding SPARQL query Q_p that contains a triple pattern of the form: *?s db-p:p ?o* where p is the property. We define $Q_p(D)$ as the result of the evaluation of Q_p over D. In Listing 1.2, for the property $db-p:birthplace$, $Q1_{birthplace}(D)$ is the result of evaluation of the query over DBpedia, i.e., the set of couples of Wikipedia pages that relates persons and their birthplace.

Wikipedia Mode:l Wikipedia can be described as a graph where nodes are the Wikipedia articles (regular articles and categories) and hyperlinks are the edges. For example, an extract of Wikipedia graph is represented at the top of Figure 3: boxes are the nodes and arrows are the edges.

Definition 1 (Wikipedia Graph). *$G = (W, E)$ where W is a set of nodes and $E \subseteq W \times W$ is a set of edges. Nodes are Wikipedia articles (wiki pages) and edges are links between articles. Given $w_1, w_2 \in W$, $(w_1, w_2) \in E$ if and only if there is a link from w_1 to w_2.*

Definition 2 (Navigational Path). *A navigational path $P(w_1, w_n)$ between two Wikipedia articles is a sequence of pages $w_1/\ldots/w_n$, s.t. $\forall i\, w_i \in W \wedge \forall i, j : 1 \leqslant i < j \leqslant n$, $w_i \neq w_j$, $\forall i : 1 \leqslant i \leqslant n-1$ where $(w_i, w_{i+1}) \in E$ is a link between w_i and w_{i+1}. w_1 and w_n are called the* source *page and the* target *page respectively. The length of a navigational path is the number of articles in the sequence, length $P(w_1, w_n) = n$.*

Definition 3 (Wikipedia Connected Pairs). *Let $Q_p(D)$ denote the result of the execution of the query Q_p against the dataset D. The set of pairs (f, t) in Wikipedia which are connected by a navigational path with length up to l, where*

$(f,t) \in Q_p(D)$ *is defined as :*
$C_p(l) = \{(f,t) \in Q_p(D)$ *such that* $\exists P(f,t)$ *and* $length(P(f,t)) <= l\}$.

A path query is a generalization of similar navigational paths. Usually, regular expressions are used for expressing path queries [20]. Many works have been done on path queries in different domains [20–22]. We adapt the path query definition in [21, 20] to the context of Wikipedia. We use regular expression patterns [20], i.e., patterns that include variables.

Definition 4 (Regular expression pattern [20]).
Let Σ *be an alphabet,* X *be a set of variables, the set of regular expressions* $R(\Sigma, X)$ *over* Σ *can inductively defined by: (1)* $\forall a \in \Sigma$, $a \in R(\Sigma, X)$. *(2)* $\forall x \in X, x \in R(\Sigma, X)$*; (3)* $\epsilon \in R(\Sigma, X)$. *(4) If* $\forall A \in R(\Sigma, X)$ *and* $\forall B \in R(\Sigma, X)$ *then* $A.B$, $A^* \in R(\Sigma, X)$*; such that* $A.B$ *is the concatenation of* A *and* B *and* A^* *denotes the Kleene closure.*

Definition 5 (Language defined by a regular expression pattern [20]).
Let Σ *be an alphabet,* X *be a set of variables, and* $R, R' \in R(\Sigma, X)$ *be two regular expression patterns.* $L^*(R)$ *is the set of words of* $(\Sigma \bigcup X)^*$ *defined by: (1)* $L^*(\epsilon) = \{\epsilon\}$. *(2)* $L^*(a) = \{a\}$. *(3)* $L^*(x) = \Sigma \bigcup X$. *(4)* $L^*(R.R') = \{w'.w \,|\, w \in L^*(R)$ *and* $w' \in L^*(R')\}$. *(5)*L^* $(R^+) = \{w_1 \dots w_k \,|\, \forall i \in [1...k], w_i \in L^*(R)$. *(6)* $L^*(R^*) = \{\epsilon\} \bigcup L^*(R+)$.

A path query is a generalization of similar navigational paths by regular expressions patterns. A path query is defined by:

Definition 6 (Path Query).
A Wikipedia path query (in short path query) $PQ \in R(\Sigma, X)$ *is a regular expression pattern. A pair of nodes* (x, y) *of* G *covers (or satisfies) a path query* $PQ(x, y)$ *over* Σ *and* X *if there exists a path* P *from* x *to* y *in* G *and a map* μ *from* $\Sigma \bigcup X$ *to* $term(G)$ *such that* $\Lambda(P) \in L^*(\mu(R))$ *where* $\Lambda(P) = \Lambda(a_1) \dots \Lambda(a_k)$ *over* $(\Sigma \bigcup X)^*$ *is associated to the path* $P = (a_1, ..., a_k)$ *of* G.

In the context of Wikipedia $\Sigma = W$. For the purpose of this work, we limit X to two variables $X = \{\#from, \#to\}$. For a pair of Wikipedia pages (x, y) then, $\Lambda($ #from $) = x$, $\Lambda($#to $) = y$ and $\Lambda(w) = w', w' \in W$, in case w includes in its a literal occurrence of the symbol #from or #to they will replace by x or y respectively, otherwise $w = w'$ (for example, $\Lambda(Category : \#from) = Category : Paris$ for the pair (Paris, Pierre Curie). Given a $Q_p(D)$, $C_p(l)$ is the set of all pairs $(f,t) \in Q_p(D)$ that are connected in Wikipedia by a path with length up to l. We will use path queries and computes the coverage of path queries for a set of pairs of Wikipedia articles.

The grey box at the middle of Figure 3 shows the navigational paths and path queries that can be generated for *(Paris, Pierre Curie)* and *(Rosario, Lionel Messi)* Wikipedia pairs in the example. Notice both cases are covered by a same path query (`#from / Category:#from / Category:People from #from / #to`) instead of the fact that they have different navigational paths.

Definition 7 (Path Index). *Given a $C_p(l)$, a Path Index (PI) is a bipartite graph $(PQ, C_p(l), I)$, it represents the coverage of path queries for a set of pairs of Wikipedia articles that are related by a DBpedia property p. PQ is an ordered set of path (descendent order by element degree), $I \subseteq PQ \times C_p(l)$ is the set of edges relating elements from PQ with elements from $C_p(l)$; $(pq, v) \in I \Leftrightarrow pq \in PQ \wedge v \in C_p(l) \wedge v$ covers pq. The first path query in PQ is the general representation of the property p in Wikipedia.*

Definition 8 (Rating). *Given a Path Index $PI = (PQ, C_p(l), I)$ and a path query $pq \in PQ$ the rating of the pq in the path index is defined by the degree of pq in the path index bipartite graph:*
$rating(pq, PI) = |\{e \in C_p(l) : (pq, e) \in I\}|$

Path Index are useful because they enable us to obtain the following information:

- Which is the most general path query?
- Which path queries cover a connected pair of Wikipedia articles?
- How many pairs are covered by a path query?

The Path Index for the examples of *(Rosario, Lionel Messi)* and *(Paris, Pierre Curie*) is shown at the bottom of Figure 3. The pink ellipses with the path queries are the PQ set, the yellow ellipses with the Wikipedia pairs are the $C_p(l)$ set, and the arrows in between is the set of edges I. Finally, the *rating* for `#from / Category:#from / Category:People from #from / #to` is 2, and the *rating* for `#from / #to` is 1.

The information disparity between DBpedia and Wikipedia is detected by comparing the difference between the set of connected pairs of resources in DBpedia and those corresponding pairs that are connected in Wikipedia.

Definition 9 (Wikipedia pair connection). *Two Wikipedia articles (a, b) that are related by a DBpedia property with one-to-many cardinality are connected when at least one of the following conditions is true:*

1. *There is a navigational path from a to b through the category tree with length less or equal to five [5, 7].*
2. *There is a direct link from a to b.*
3. *a has a direct link to a `List of` page that has a direct link to b.*

For instance, `Rosario,_Santa_Fe` and `Lionel_Messi` are connected according to the first condition since there is navigational path:`Rosario,_Santa_Fe / Cat:Rosario,_Santa_Fe / Cat:People_from_Rosario,_Santa_Fe / Lionel_Messi`.

`Rosario,_Santa_Fe / Lionel_Messi` are connected according to the second condition, and, finally, `Al_Pacino / List_of_awards_and_nominations_received_by_Al_Pacino / Academy_Award` connects the pair elements *(Al_Pacino, Academy_Award)* following the third condition.

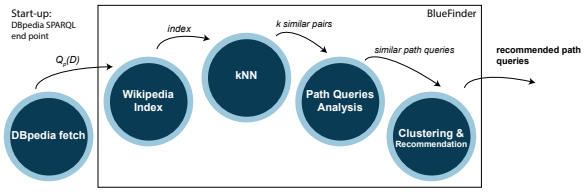


Fig. 4: BlueFinder algorithm steps

4 Problem Statement

We describe the problem of defining the best representation of missing links in Wikipedia as a collaborative recommender system problem. According to Adomavicioius and Tuzhilin [23], *"collaborative recommender systems try to predict the utility of items for a particular user based on the items previously rated by other users"*. BlueFinder predicts links between Wikipedia articles based on links previously rated by the Wikipedia community. Thus, BlueFinder can be considered as a collaborative recommender system for enhancing content of Wikipedia.

More formally, the utility function $u(c, s)$ of item s for user c is estimated based on the utilities $u(c_j, s)$ assigned to item s by those users $c_j \in C$ who are "similar" to user c. In the context of Wikipedia, BlueFinder does not directly apply recommenders to suggest Wikipedia articles to users but to suggest links between articles. BlueFinder predicts the utility of path queries for a particular pair of Wikipedia articles based on those rated by the Wikipedia community. In other words, the pairs of articles *(from,to)* will play the role of users and the path queries will be the items. Then, the utility $u(c, pq)$ of a path query pq for a pair c related by a semantic property p is estimated based on the utilities $u(c_j, pq)$ assigned to pair c by those pairs $c_j \in C_p(l)$, $u : Q_p(D) \times PQ \rightarrow R$, where R is a list of path queries sorted according to the rating (see Definition 8).

Given a property p in DBpedia, $C_p(l)$ and PQ path queries covered by the elements of $C_p(l)$. Then, for a given pair of Wikipedia articles $(from, to)$, we have to recommend the path query that maximizes the utility function. The following use case illustrates this problem statement in a practical use.

Use Case 1 A user would like to know *which is the best convention to represent the* is birthplace of *semantic relation for the pair of articles* (Boston, Robin Moore) *in Wikipedia*. The expected result is a list of recommended navigational paths that could connect Boston and Robin Moore articles respecting Wikipedia conventions. Some possible answers could be the following navigational paths:

- Boston / * / Category: Writers from Boston / Robin Moore (High confidence)

– Boston / Robin Moore (Low confidence)

Algorithm 1 BlueFinder

Require: x : unconnected pair, $maxR$: maximum number of recommendations, $Q_p(D)$, k : number of neighbors, l:max path length
Ensure: Recommendation path query set
1: $index = (PQ, C_p(l), I) \leftarrow WikipediaIndex(Q_p(D), l)$
2: $k_{neighbors} \leftarrow kNN(x, C_p(l))$
3: $knnPQ \leftarrow \bigcup_{c_i} pq : (pq, c_i) \in I, c_i \in k_{neighbors}$
4: $knnI \leftarrow \bigcup_{c_i} (pq, c_i) : (pq, c_i) \in I, c_i \in k_{neighbors}$
5: $knnPI \leftarrow (knnPQ, k_{neighbors}, knnI)$
6: $M \leftarrow NoiseFilter(knnPI)$ {M ordered in rating descendent order}
7: $M \leftarrow StarGeneralization(M, knnPI)$
8: **return** $maxRecom$ path queries of M

The first specifies a navigational path that starts in Boston article, then it could continue by several other articles but it has to finish in the category *Writers from Boston* and finally *Robin Moore* article. The second navigational path specifies the direct navigation from *Boston* to Robin Moore. Additionally, the first respects the Wikipedia convention with a higher level of confidence than the second one.

5 BlueFinder

BlueFinder implements a four-steps pipeline process as shown in Figure 4. A pre-processing step *DBpedia fetch* configures the BlueFinder start-up information. It fetches from DBpedia SPARQL endpoint the set of pairs of Wikipedia articles $Q_p(D)$ that are related in DBpedia by a semantic property p. After having the $Q_p(D)$, BlueFinder algorithm is ready to start.

The BlueFinder Algorithm 1 receives five inputs: (1) the unconnected pair of Wikipedia articles x, (2) maximum number of recommendations $maxR$, (3) the $Q_p(D)$ set generated by *DBpedia fetch* step, (4) the number of neighbors k , and (5) the maximum length of a path l. BlueFinder algorithm starts by invoking the *WikipediaIndex*.

The *WikipediaIndex* Algorithm 2 builds a path index. It receives $Q_p(D)$ and computes the item set, user set and item ratings. The items are the path queries, and the users are the pairs of Wikipedia pages retrieved from DBpedia. In this case, for each pair of Wikipedia articles $(from, to)$ included in a given $Q_p(D)$, the algorithm performs a depth-first search up to l starting from the $from$ article and finishing in the *to* article in the Wikipedia graph (lines 1-4). For each reaching *to* article, it generalizes a path and builds the path index as a bipartite graph (lines 5-8). Finally, it returns the path index that is ready to be used in the next step of the BlueFinder algorithm 1. BlueFinder traverses Wikipedia graph

starting from the $from$ article and finishing in the to article in the Wikipedia graph until the maximum length of a path; consequently, a depth-first search is more appropriate than the breadth-first search for building path index.

Algorithm 2 WikipediaIndex

Require: $Q_p(D)$, l: path length
Ensure: PI bipartite graph
1: $index = (\varnothing, \varnothing, \varnothing)$
2: **for all** $(from, to) \in Q_p(D)$ **do**
3: $allPaths \leftarrow \varnothing$, $curL \leftarrow 0$, $curPath \leftarrow \varnothing$
4: $GenerateAllPaths(from, to, l, curL, allPaths, curPath)$
5: **for all** $path \in allPaths$ **do**
6: $pathQuery \leftarrow BuildPathQuery(path, from, to)$
7: $index \leftarrow InsertInIndex(index, pathQuery, (from, to))$
8: **end for**
9: **end for**
10: **return** $index$

After indexing, BlueFinder performs the *kNN* step (line 2 in Algorithm 1). In this step, given a disconnected pair of articles in Wikipedia, BlueFinder identifies the k nearest connected pairs to the disconnected one.

Algorithm 3 GenerateAllPaths

Require: $from, to$: Wikipedia article, $l, curL$: integer, $allPaths$: $setOfPaths$, $curPath$: $path$
Ensure: All paths that start in $from$ and end in to in Wikipedia Graph with length up to l. The results only include paths through the category tree, the use of List_of_ pages or direct links.
if $from = to$ **then**
$allPaths \leftarrow allPaths \bigcup \{curPath\}$
else if $l > curL$ **then**
{Traverse through Wikipedia graph edges set E}
for all $neighbor \in \{n : (from, n) \in E\}$ **do**
$curPath \leftarrow curPath + neighbor$
$curL \leftarrow curL + 1$
$GenerateAllPaths(neighbor, to, l, curL, allPaths, curPath)$
$curPath \leftarrow curPath - neighbor$
$curL \leftarrow curL - 1$
end for
end if
return $allPaths$

The kNN algorithm uses a similarity measure function to select the nearest neighbors. We define the *Semantic Pair Similarity Distance (SPSD)* function to

measure the similarity between pairs of article. SPSD is based on the well-known Jaccard distance [24], it measures the degree of overlapping in the DBpedia types that describe a pair of Wikipedia articles. The range of the SPSD is from 0 (identical pairs) to 1 (totally disjoint pairs). The Semantic Pair Similarity Distance function is defined as:

Definition 10 (Semantic Pair Similarity Distance (SPSD)). *Given two pairs of pages* $c_1 = (a_1, b_1)$ *and* $c_2 = (a_2, b_2)$. *Let* t_{a_1}, t_{b_1}, t_{a_2}, t_{b_2} *data types in DBpedia for* a_1, b_1, a_2 *and* b_2 *respectively. Data types are defined as:*
$t_{a_1} = \{t : < a_1$ *rdf:type* $t > \in DBpedia\}$, $t_{b_1} = \{t : < b_1$ *rdf:type* $t > \in DBpedia\}$,
$t_{a_2} = \{t : < a_2$ *rdf:type* $t > \in DBpedia\}$, $t_{b_2} = \{t : < b_2$ *rdf:type* $t > \in DBpedia\}$.
$SPSD(c_1, c_2) = \frac{J(t_{a_1}, t_{a_2}) + J(t_{b_1}, t_{b_2})}{2}$

where J is Jaccard distance between c_1 and c_2. $J(c_1, c_2) = \frac{|c_1 \cup c_2| - |c_1 \cap c_2|}{|c_1 \cup c_2|}$.

To illustrate, we consider two pairs of Wikipedia pages $c1 = (Paris, PierreCurie)$ and $c2 = (Paris, Larusso)$. The data types are:

- $t_{paris} = \{EuropeanCaptialsOfCulture, PopulatedPlace\}$.
- $t_{PierreCurie} = \{Scientist, FrenchCheimists, PeopleFromParis\}$.
- $t_{Larusso} = \{Artist, PeopleFromParis\}$.

$$SPSD(c_1, c_2) = \frac{J(t_{paris}, t_{paris}) + J(t_{PierreCurie}, t_{Larusso})}{2} = (0 + 0.75)/2 = 0.375$$

Now, we can define the kNN [25] in our context as:

Definition 11 (kNN). *Given a pair* $r \in Q_p(D)$ *and an integer* k, *the* k *nearest neighbors of* r *denoted* $KNN(r, Q_p(D))$ *is a set of* k *pairs from* $Q_p(D)$ *where* $\forall o \in KNN(r, Q_p(D))$ *and* $\forall s \in Q_p(D) - KNN(r, Q_p(D))$ *then* $SPSD(o, r) \leq SPSD(s, r)$.

Having the kNN step computed, the *Path Queries Analysis* step starts. It obtains the path queries that connect the k neighbors in a smaller path index than the original (from line 3 to 5 in Algorithm 1). Indeed, having a $PI = (PQ, C_p(l), I)$; the value for an unknown rated $r_{c,s}$ for unconnected pair in Wikipedia c and a path query $s \in C_p(l)$, can be computed as:

$$r_{c,s} = degree(s, PI') + featured(pq, PI')$$

where $PI' = (PQ, C_p(l)', I)$ and $C_p(l)' = KNN(c, C_p(l))$ and $featured(pq, PI') = \beta$ if $degree(pq, PI') = |C_p(l)'|$, otherwise $featured(pq, PI') = 0$. β is a scalar used to promote those path queries that are a convention shared for all the k neighbors[12] and we call *featured predictions*. BlueFinder has a high level of confidence in *featured predictions*.

The generated path index contains the path queries that will be recommended and its ratings. Before the recommendations are returned, in the step *Clustering*

[12] In this work, we use $\beta = 1000$.

and Recommendation in Figure 4, BlueFinder cleans regular-user-unreachable-paths (e.g., paths that include administrative categories) by means of the noiseFilter (Algorithm 4) and similar path queries are grouped by StarGeneralization algorithm (Algorithm 5). Finally, BlueFinder returns the $maxRecom$ best ranked path queries.

The $NoiseFilter$ Algorithm 4 deletes all the paths queries that are not accessible by a Wikipedia user. Wikipedia includes several administrative categories which are used by administrators. In order to recommend path queries that can be utilized by regular users, $NoiseFilter$ deletes those categories whose names begin with "Articles_", "All_Wikipedia_", etc, such as `Cat:Articles_to _be_merged`.

Algorithm 4 NoiseFilter

Require: $PI = (PQ, C, I)$: Path index
Ensure: Set of regular user navigable path queries.
  $noise = \{"Articles\_","All\_Wikipedia\_","Wikipedia\_","Non-free","All\_pages\_","All\_non"\}$
  for all $pq = (p_1, .., p_n) \in PQ$; **do**
    if p_i contains any c $\in noise; 1 \leq i \leq n$ **then**
      $PQ \leftarrow PQ - \{pq\}$
    end if
  end for
  return PQ

BlueFinder filters path queries into star path queries in order to reduce data sparsity.

Definition 12. *A star path query $PQ^*(f,t)$ is a group of similar path queries that meet the following rules: (1) $PQ^*(f,t)$ starts with* `#from` *and ends with* `#to`*. (2) The * element can only be placed between* `#from` *and* `#to` *variables and * represents any path query. (3) The * cannot be the penultimate element in the path query because it has to make explicit the last part of the path in order to make the connection with the* `#to` *page.*

Example 1. $PQ^*(f,t)$ =`#from/*/Cat:People_from_#from/ #to` is a star path query. $PQ^*(f,t)$ =`#from/*/#to` is not a star path query.

The *StarGeneralization* Algorithm 5 groups path queries into a star path query, if possible.

Solving the Use Case 1 We start by fetching from Wikipedia all the pairs of Wikipedia articles with the form (Place, Person) that are related by the semantic relation *is birthplace of*. The SPARQL query evaluated in the DBpedia endpoint is Q1 (Listing 1.2) and the result is the $Q_p(D)$ set. Later, we initialize the BlueFinder algorithm with the following parameters: (Boston, Robin Moore) as

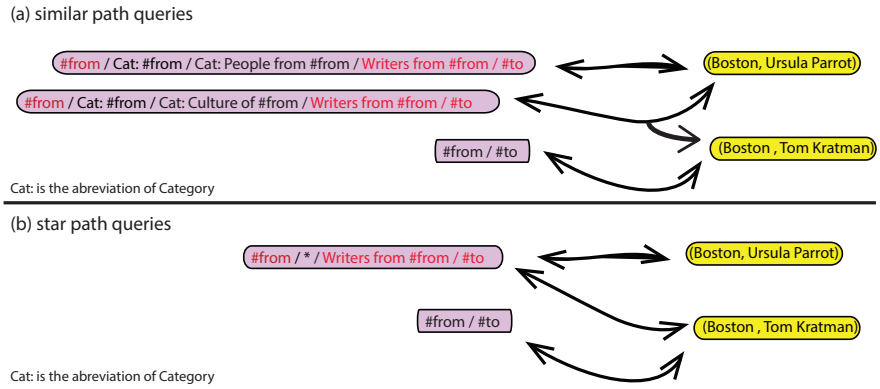


Fig. 5: BlueFinder: Similar path queries and star path queries in the execution of Use Case 1

the unconnected pair x, the limit of recommendation with $maxR = 3$, in order to simplify the example only 2 neighbors ($k = 2$), and the maximum length of pairs with 5. In Section 6 we evaluate the behavior of BlueFinder with a combination of several values of k and $maxR$.

The Wikipedia Index step generates a similar but larger index than the one exemplified in Figure 3. This *index* includes 409,812 connected pairs ($C_p(l)$), 65,262 path queries (PQ) and 648,631 edges (I). With the index, the next step selects the 2 connected pairs of articles that are most similar to (Boston, Robin Moore). The SPSD detects (Boston, Ursula Parrot) and (Boston, Tom Kratman) as the most similar pairs because both of them are sharing most of the DBpedia types with (Boston, Robin Moore). For example all of them are writers from Boston. After that, BlueFinder only analyzes the path queries that cover the k neighbors in a smaller index as shown in Figure 5 (a). Those similar path queries are generalized into star path queries in the *Clustering and Recommendation* step. The generalization generates the index shown in Figure 5 (b).

Finally, the path query `#from / .../ Category: Writers from #from / #to` will be a featured recommendation because it covers all the neighbors. The other path query covers only one. Then, it retrieves the following recommendations customized for Boston and Robin Moore.

- Boston / . . . / Category: Writers from Boston / Robin Moore (Featured Recommendation - High confidence)
- Boston / Robin Moore (Low confidence)

With this recommendations, the user publishes in Wikipedia a navigational path that starts in *Boston* article, and has to finish in the category *Writers from Boston*. For this case, the user only has to add the article *Robin Moore* to the category *Writers from Boston*.

6 Evaluation

In this section we analyze the behavior of our approach by means of measuring the prediction the accuracy of BlueFinder predictions over the 20 properties shown in Table 1. The evaluation is conducted to answer the following questions:

1. What is the best combination of k and $maxRecom$ values to observe the best accuracy from BlueFinder?
2. Does BlueFinder retrieve path queries that can fix missing relations in Wikipedia?
3. Does the confidence level provided by BlueFinder correlate with the accuracy of the predictions?
4. Does the Wikipedia Community use different conventions to represent a DBpedia property?

In this section we describe the method of the evaluation, the evaluation metrics, and then the data sets used in the experimentation are presented. Finally, the results and discussions are introduced.

Algorithm 5 StarGeneralization

Require: PQ: set of path queries, PI: Path index
Ensure: PQ^*: set of star path queries
  $PQ^* \leftarrow \emptyset$
  for all $pq = (p_1, .., p_{n-1}, p_n) \in PQ$; **do**
    if p_{n-1} starts with "`Cat:`" **then**
      $PQ^* \leftarrow PQ^* \bigcup \{(p_1, *, p_{n-1}, p_n)\}$
    else
      $PQ^* \leftarrow PQ^* \bigcup \{pq\}$
    end if
  end for
  return PQ^*

6.1 Method

In order to answer the questions described in the previous section, an offline evaluation was designed; user interaction is not considered in the study. The central idea of this evaluation is based on disconnecting connected pairs of articles in Wikipedia and then observing whether BlueFinder is able to recreate them. The important fact here is that BlueFinder has to recreate the Wikipedia community conventions that were defined to connect the pairs and not only to discover the disconnection. This approach is based on the assumption that all connected pairs in Wikipedia follow *Wikipedian* conventions. Figure 6 summarizes the idea of the evaluation method. For the purpose of this evaluation all the path queries

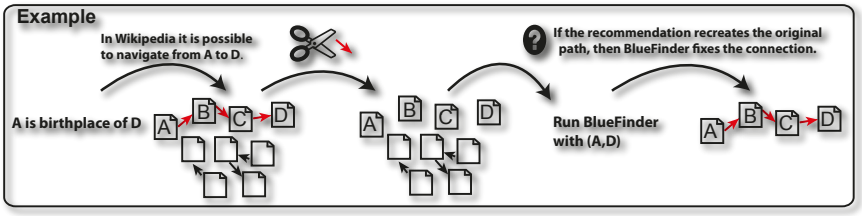


Fig. 6: Evaluation method

that connect a pair of pages that are related in DBpedia by a property p, are considered the correct paths that represent the property p.

A sample of 10% of the connected pairs was taken for the evaluation. They are randomly selected and kept in a set called N. For instance, for $prop_1$ in Table 1, 188,324 pairs are connected in Wikipedia (i.e. 409,812 - 221,788), so 18,832 randomly selected of those pairs will be in the set N. After that, for each connected pair (w_1, w_2) in N the evaluation repeats the following steps:

1. All paths currently connecting (w_1, w_2) in Wikipedia are stored in the $\mu_{relevant}$ set, and immediately all them are eliminated from Wikipedia to "disconnect" (w_1, w_2).
2. BlueFinder is executed to predict the paths that could connect (w_1, w_2). The resulting predictions are kept in $\mu_{predicted}$.
3. The $\mu_{predicted}$ set is compared with $\mu_{relevant}$ set in order to compute the metrics detailed below such as precision, recall and F1.
4. Finally, Wikipedia is restored up to the state before the pair disconnection. This means that the (w_1, w_2) pair is reconnected by means of $\mu_{relevant}$.

In this evaluation, BlueFinder behavior is evaluated in each property mentioned in Table 1, and then aggregates the values of all the metrics to have a general point of view. For example, the evaluation measures the precision metric for $prop_1$, then for $prop_2$ and then it continues with the rest of metrics and properties. After all the metrics and properties are computed, the mean of all metric values is calculated.

In order to have an analysis of the best combination of the number of neighbors and the number of the BlueFinder recommendations, the BlueFinder execution is configured with many combinations of the parameter k and $maxRecom$ for each disconnected pair. The values for k are from 1 to 10, and the values for $maxRecom$ are 1, 3, 5 and *unlimited*. The limit of path queries l was fixed in 5 according to the analysis presented previously.

Evaluation metrics We measured the accuracy of BlueFinder predictions based on the standard metrics of *Precision* (P), *Recall* (R), *F-measure*(F_1) and *hit-rate*.

Precision relates the number of correct path queries that are predicted by BlueFinder to the total of recommended path queries.

$$P = \frac{|\mu_{relevant} \bigcap \mu_{predicted}|}{|\mu_{predicted}|} \tag{1}$$

where $\mu_{predicted}$ is the set of predicted path queries and $\mu_{relevant}$ is the set of expected path queries.

Recall computes the ratio of expected path queries to the total of recommended path queries.

$$R = \frac{|\mu_{relevant} \bigcap \mu_{predicted}|}{|\mu_{relevant}|} \tag{2}$$

F_1 *score* is the combination of precision and recall.

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{3}$$

We also use the hit-rate recommendation accuracy [26, 27] that measures the number of cases where BlueFinder recommends at least one correct path query.

$$hit - rate = \begin{cases} 1 & if |\mu_{relevant} \bigcap \mu_{predicted}| > 0 \\ 0 & otherwise \end{cases} \tag{4}$$

The previous measures are extended by studying the distribution of path queries predicted by BlueFinder. In this work, we measure the statistical dispersion of each path query i according to the proportion $p(i)$ of a pair coverage using Gini index.

The *Gini index* [28] measures the distribution of recommended path queries. A value close to 0.0 indicates that all path queries are equally recommended while a value close to 1.0 represents that a particular path query is recommended always.

$$GI = \frac{1}{n-1} \sum_{j=1}^{n} (2j - n - 1) p(i_j) \tag{5}$$

where $i_1, ..., i_n$ is the list of path queries ordered according to increasing *p(i)*.

Finally, we combine the previous metrics to analyze the confidence of the predictions. A high level of confidence in a prediction means that the system trusts its prediction while a low confidence means the opposite [29]. BlueFinder determines confidence in two levels: *featured predictions*, and the position of each prediction in the recommendation set. In order to evaluate the confidence, we will compare the confidence with the hit-rate of each prediction.

Limitations In statistical terms, the $\mu_{relevant}$ set is used as the gold standard for each pair (w_1, w_2) in N. However, $\mu_{relevant}$ could contain paths that are not related to a property in Table 1, or even a potentially correct prediction could be absent in the $\mu_{relevant}$ set. For example, the $\mu_{relevant}$ set for the pair

(London , Richard_Blanshard) with the property `deathPlace` was `#from / * / Cat:People_from_#from / #to` and the first two predictions in the prediction set were `#from / * / Cat:People_from_#from / #to` and `#from / * / Cat:Death_in_#from / #to`. The second prediction could be correct but, as it is not included in $\mu_{relevant}$, the evaluation rejects it as a correct one. Taking into account these considerations, the $\mu_{relevant}$ set is an estimation of the actual path queries and in consequence the BlueFinder is evaluated in the context of the worst case.

Datasets We evaluate BlueFinder with the twenty semantic properties detailed in Table 1. For each property denoted by $prop_i$, a SPARQL query was evaluated on the DBpedia SPARQL endpoint. The SPARQL query for each property follows the template showed in Listing 1.3 and the values of *DBpediaSemanticProperty*, *fromType* and *toType* are replaced in each property scenario for the specific values of the first, second and third column respectively that are detailed in Table 1. For instance, the SPARQL query in Listing 1.2 corresponds to $prop_1$. The number of the Wikipedia connected pairs of each property is the difference between the numbers of the DBpedia connected pairs minus the number of the Wikipedia disconnected pairs (columns fourth and fifth of Table 1). The evaluation was run with a local copy of the English Wikipedia and DBpedia download in July 2013 and they were stored in a MySQL database.

```
prefix db-owl:<http://dbpedia.org/ontology/>
prefix db-p:<http://dbpedia.org/property/>
prefix  foaf:<http://xmlns.com/foaf/0.1/>
select ?fr  ?to
where  { ?db_from a <fromType>  .
         ?db_to a <toType>  .
         ?db_to db-p:<DBpediaSemanticProperty> ?db_from  .
         ?db_from foaf:isPrimaryTopicOf ?fr  .
         ?db_to  foaf:isPrimaryTopicOf ?to
       }
```

Listing 1.3: SPARQL query template for evaluation scenarios.

Evaluation results are described and discussed in the next section. The complete values of all the metrics values with the different values for k and $maxRecom$ of this evaluation are in `https://sites.google.com/site/bfrecommender/publications/`.

6.2 Results and discussion

We start by explaining the information gap presented in Table 1, then we report results for each evaluation metric.

Gap analysis The last column in the in the Table 1 presents the gap of missing information in Wikipedia. By analyze the ratio of the gap, shown in Figure 2,

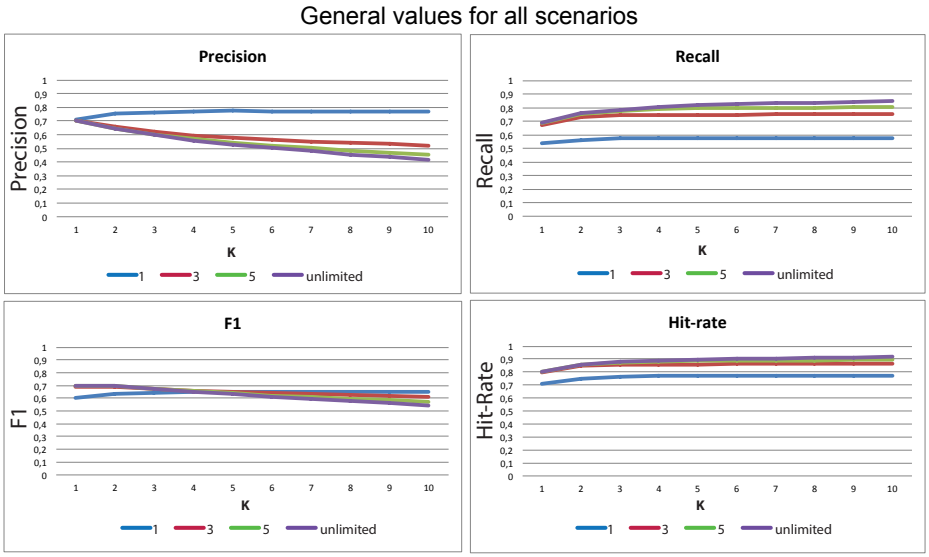


Fig. 7: Precision, Recall, F1, and Hit-rate mean of all properties

we noticed that 9 out of 20 properties have more than 50 % of missing information. In addition, the number of disconnected pairs of the properties $prop_1$ (`birthplace`), $prop_2$ (`deathplace`), and $prop_{13}$ (`recordlabel`) shown in the Table 1, is equivalent to more than the 50 % of all disconnected pairs of other properties.

The smallest gap was in $prop_{20}$ (`notableWork`) and $prop_{12}$ (`previousWork`) with only 5 %. In both cases, this is because the links represent basic information of the connected articles and they are expressed as direct links (`#from / #to`) between the articles.

Accuracy To assess the best behavior of BlueFinder, we analyze the values of accuracy metrics for the 20 properties from a general perspective. Figure 7 shows four line-charts with the mean values of $precision$, $recall$, $F1$ and $hit{-}rate$ obtained for each property. Each chart describes the relation between $maxRecom$ and k values for each metric.

BlueFinder is able to find, on average, between 75 % and 82 % of the relevant paths, and according to the hit-rate values it is able to fix around 88 % of the cases for k greater than 4 and $maxRecom = 3$, 5 or $unlimited$. However, the limitations is that the precision values decrease according to the variation of the k values and the number of recommendations.

To detect the best correlation between precision and recall we use the F1 metric. According to the Figure 7, all the $maxRecom$ curves converge at k=5 with value 0.65. Therefore, $maxRecom = 5$ and $k = 5$ determine the best accuracy for BlueFinder. The number of correct path queries tips the scales in favor

of recall and hit-rate rather than precision. This assumption is based on the fact that the recommendations are presented to the users in descending confidence order, and consequently, the users have extra information to determine the accuracy of the recommendation. Finally, the unlimited $maxRecom$ was dismissed because it had similar recall than $maxRecom = 5$ but lower precision.

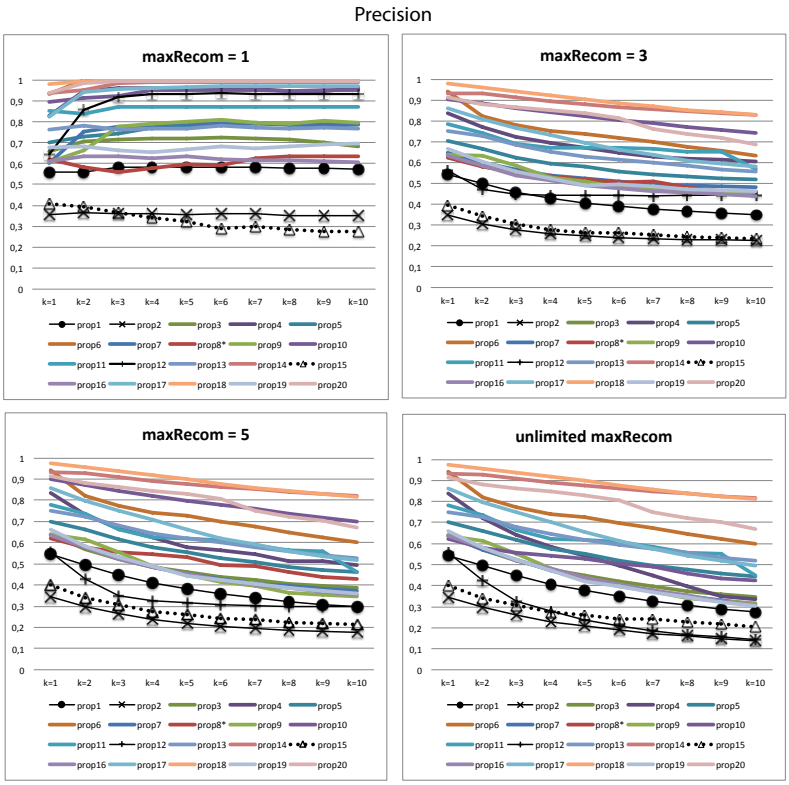


Fig. 8: Precision all properties

Precision As presented in Figure 8 most of the precision curves decrease, due to BlueFinder introduced non-expected path queries in $\mu_{predicted}$ set. This is because of the size of $maxRecom$ has increased but also because the distant neighbors insert noisy paths. Nevertheless, the 70 % of the properties had precision higher than 0.5 at k=5 and $maxRecom$=5. This evidences that in general terms the precision of BlueFinder was considerably good taking into account that, as we have mentioned, the predictions are presented in confidence order bringing to the users better information (more details in Section 6.2).

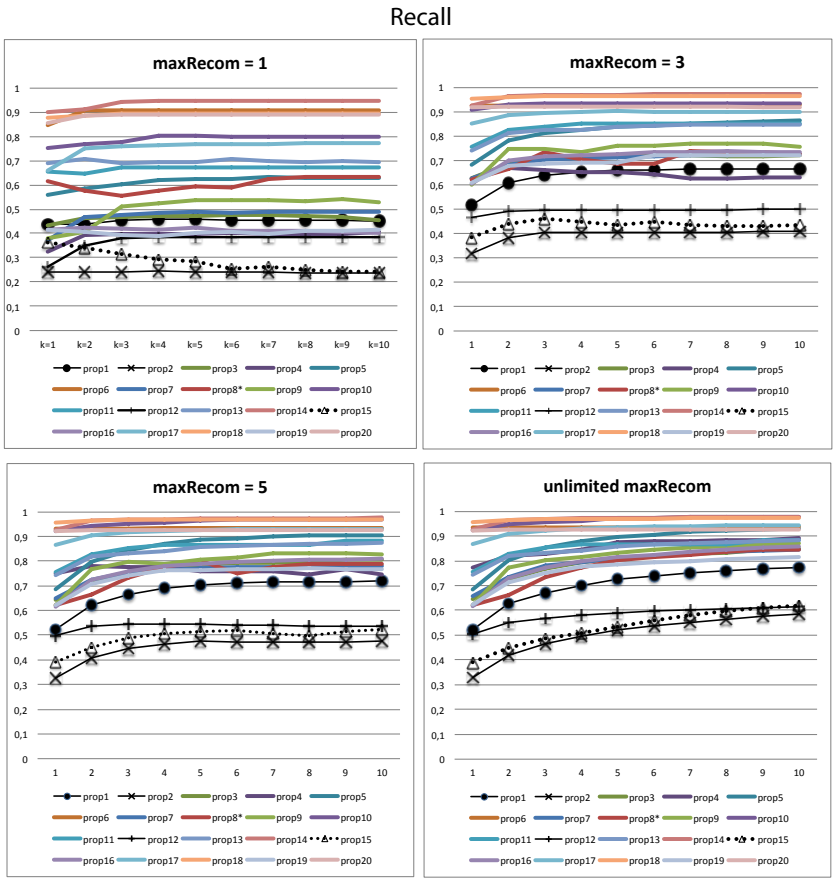


Fig. 9: Recall for all properties

Properties $prop_1$: *birthplace*(• line in charts), $prop_2$: *deathplace* (line with ×), and $prop_{15}$: *country* (line with △) have low precision less than 0.44 for any *maxRecom* value. This is because they have a high number of disconnected pairs and shows up that BlueFinder, as many recommender systems, is sensitive to the sparsity of data. On the other hand, property $prop_{12}$: *previousWork* (line with +) had a high precision: 0.93 with $maxRecom = 1$ and $k = 5$, but it sharply decreased when the *maxRecom* increased (0.44 with $maxRecom = 3$, 0.31 with $maxRecom = 5$ and 0.23 with $maxRecom = unlimited$). This is because although BlueFinder was able to predict the correct path query for this property, the other path queries that are included in the prediction set are not correct.

Recall and F1 As depicted in Figure 9, the recall of seventeen properties is greater than 0.7, and the recall of eleven properties is greater than 0.8; all of

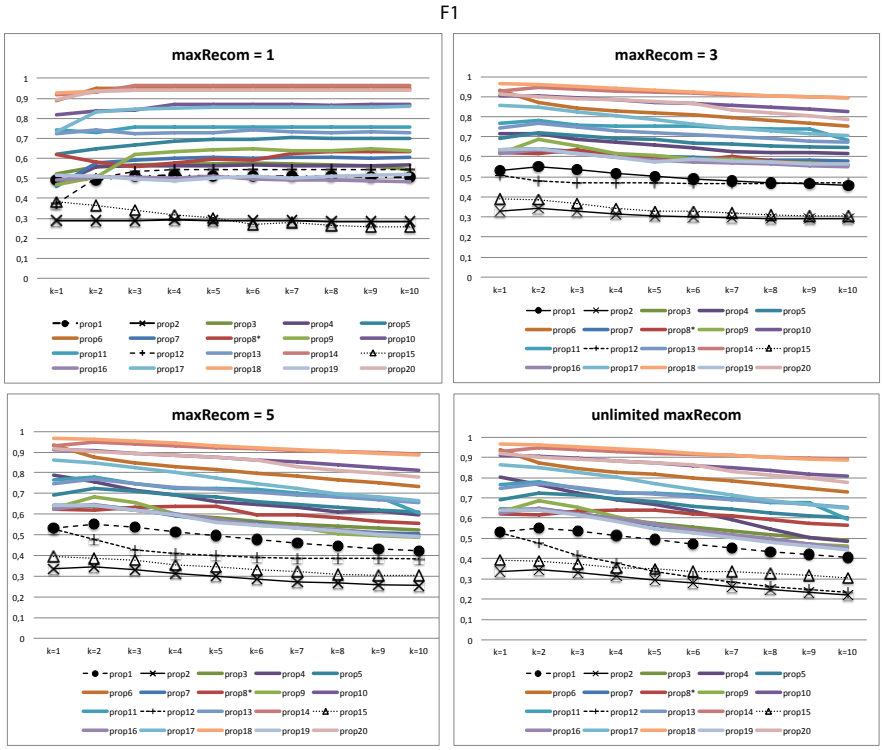


Fig. 10: F1 for all properties

them with k=5 and $maxRecom$=5. Again properties $prop_{12}$, $prop_2$, and $prop_{15}$ are out of norm. The lowest recall value with k=5 and $maxRecom$=5 is 0.473 for $prop_2$ property, and the maximum value is 0.972 for $prop_{14}$ property.

Figure 10 presents the valus of F1 metric, although the values for the properties $prop_2$, $prop_{12}$ and $prop_{15}$ are low, twelve properties out of twenty have F1 values greater than 0.6 at k=5 and $maxRecom$=5.

Hit-rate As depicted in Figure 11, 80 % of the properties (16 out of 20) have a hit-rate greater than than 0.84, and only two properties have values lower than 0.6. The properties with the lowest hit-rate are $prop_{15}$ and $prop_2$; both confirmed the same tendency that appeared in the previous accuracy metric. Although the hit-rate values are low, according to its high level of information gap, the average of hit-rate for the property $prop_8$ is greater than 75 %. These values are promising since 98 % of properties pairs were disconnected in Wikipedia.

The accuracy values demonstrated that BlueFinder retrieves good recommendations. The hit-rate curves confirmed the best combination of k and $maxRecom$ values by setting $k = 5$ and $maxRecom = 5$.

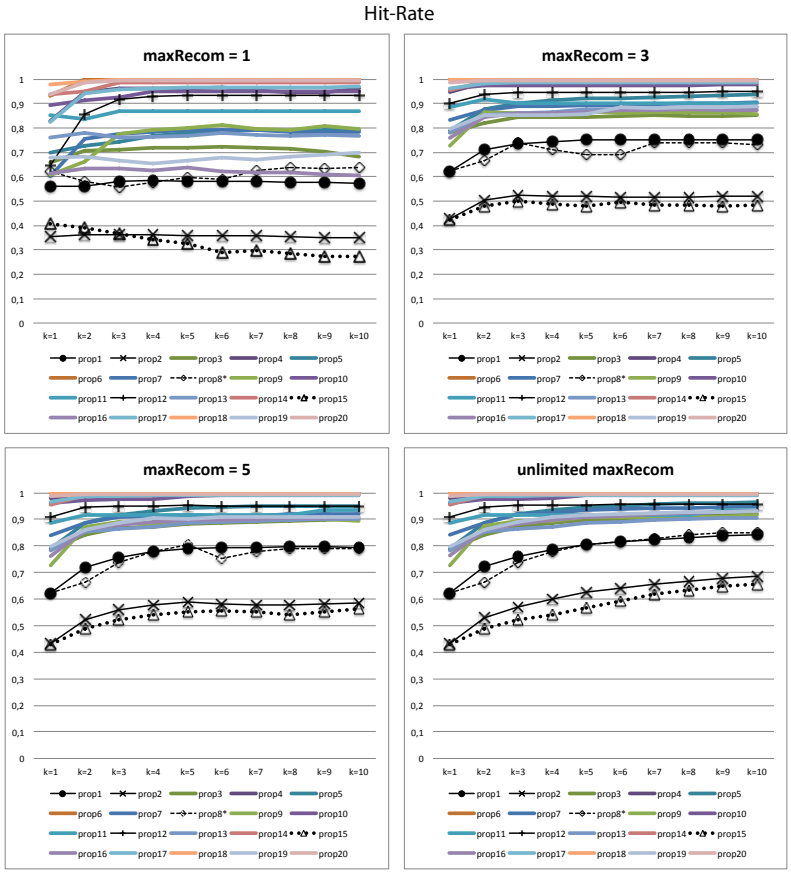


Fig. 11: Hit-rate for all properties

The BlueFinder predictions are sorted in confidence descendent order; consequently, first ranked predictions may have a better hit-rate than the following ones. In order to answer the third question of the evaluation *Does the confidence level provided by BlueFinder correlate with the accuracy of the predictions?* the hit-rate of BlueFinder featured predictions for the twenty semantic properties is shown in Table 2. As we can see, featured predictions made by BlueFinder are chiefly prominent: the lowest ratio was 0.84 for property $prop_1$ and the highest ratio was 1 for properties $prop_6$ and $prop_{15}$. The mean of all the hit-rate values was nearly 0.95 and the geometric mean was similar ($\simeq$0.95). This means that BlueFinder is able to fix nearly all the cases where it recommends a featured prediction. Additionally, more than 50 % of the BlueFinder recommendation in this evaluation were featured recommendations. This means, making a projection over the unconnected cases, around 270,367 of new connections in Wikipedia.

DBpedia Property	Hit-rate
$prop_1$	0.849264
$prop_2$	0.853425
$prop_3$	0.914697
$prop_4$	0.992218
$prop_5$	0.954365
$prop_6$	1
$prop_7$	0.918067
$prop_8$	0.968254
$prop_9$	0.97541
$prop_{10}$	0.977757
$prop_{11}$	0.938776
$prop_{12}$	0.93861
$prop_{13}$	0.950509
$prop_{14}$	0.994295
$prop_{15}$	1
$prop_{16}$	0.868421
$prop_{17}$	0.979371
$prop_{18}$	0.997813
$prop_{19}$	0.938967
$prop_{20}$	0.993056
Mean	**0.95016375**
Geometric Mean	**0.94896**

Table 2: Confidence and hit-rate for High Confidence predictions

Confidence Additionally, Figure 12 extends the information from Table 2 to all the recommendation positions in a line-chart which compares the hit-rate to the first five positions of the BlueFinder recommendation. As we expected, the curve is in descending order while the first position has the best hit-rate (0.78) and the last position the lowest (0.17). This confirms the correlation between the confidence and hit-rate of the predictions. Unfortunately, the hit-rate curve descends more rapidly than we expected to second position and continues descending until the last position.

Distribution of the BlueFinder Recommendations Wikipedia editors mainly use one convention to represent 18 out of 20 properties in Table 1. BlueFinder is able to predict one path query for these properties; in this case the Gini index is greater than 0.8.

However, because Wikipedia editors use more than one convention for properties such as $prop_9$ and $prop_{11}$, the BlueFinder predictions have Gini index values between 0.474 and 0.778 for different k, and BlueFinder predicts several path queries. For instance, the convention for the property $prop_9$: *debutStadium* is either a *Standalone list* or a *Category* such that `#from / List_of_West_Coast_-Players / #to` and `#from / * / Category: West_Coast_Players / #to`.

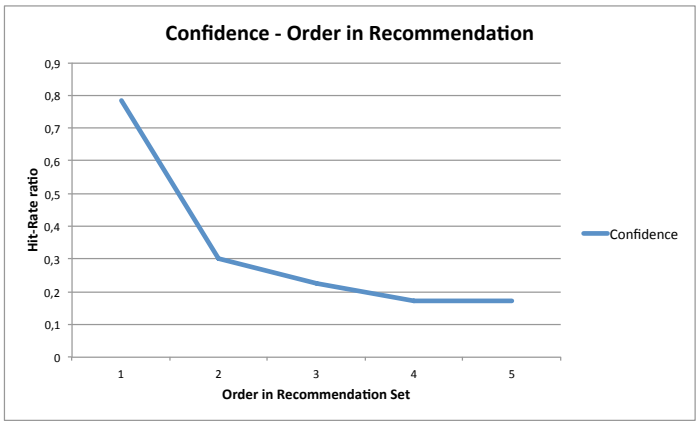


Fig. 12: Confidence and hit-rate according to prediction order in the recommendation set

General Evaluation Conclusions Evaluation show that the information gap between DBpedia and Wikipedia is a real and important problem. According to the evaluations, the best accuracy of BlueFinder is obtained with $k = 5$ and $maxRecom = 5$, and this answers the first question of the evaluation. With these values, the BlueFinder predictions maximize the expected results with a balanced F1 value.

Additionally, on average 89 % of the disconnected pairs are fixed by BlueFinder according to the hit-rate values and almost all the featured predictions fix the disconnection. A Wikipedia editor could use BlueFinder and fix unconnected pairs in Wikipedia.

In order to answer the third question, BlueFinder gives the user the recommendations in descending confidence order. The confidence is also correlated with the hit-rate of the prediction. This enables users to make a better choice of the predictions. The hit-rate of the predictions in the first position is accurate and it is also better when the prediction is a *featured prediction*. Wikipedia editors, in general, only use one convention to represent links in Wikipedia but in some cases; two in this evaluation; some communities define particular conventions. We can conclude this by taking into account that the prediction distribution is centralized in one convention. Additionally, the evaluations showed that those predictions are accurate.

BlueFinder gives good predictions even when the contributors have different conventions. Indeed, this can be concluded by taking into account that the Gini Index, in most of the cases, defined a centralized distribution of a path query in the recommendations, and also the accuracy level of BlueFinder in those cases.

DBpedia Property	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
$prop_1$: birthPlace	0.946	0.921	0.902	0.887	0.875	0.866	0.857	0.850	0.843	0.836
$prop_2$: deathPlace	0.922	0.890	0.866	0.849	0.835	0.823	0.814	0.806	0.798	0.792
$prop_3$: party	0.954	0.938	0.928	0.921	0.912	0.906	0.902	0.898	0.894	0.891
$prop_4$: firstAppearance	0.909	0.899	0.885	0.873	0.862	0.854	0.849	0.838	0.835	0.831
$prop_5$: recordLabel	0.956	0.938	0.925	0.912	0.902	0.895	0.889	0.881	0.876	0.871
$prop_6$: associatedBand	0.942	0.929	0.923	0.915	0.907	0.896	0.885	0.872	0.865	0.856
$prop_7$:Company	0.941	0.917	0.901	0.886	0.876	0.872	0.866	0.860	0.855	0.850
$prop_8$:recordedIn	0.861	0.833	0.830	0.813	0.802	0.790	0.769	0.757	0.747	0.736
$prop_9$:debutstadium	**0.676**	**0.648**	**0.613**	**0.594**	**0.595**	**0.581**	**0.576**	**0.561**	**0.545**	**0.524**
$prop_{10}$:producer	0.959	0.944	0.933	0.927	0.921	0.915	0.911	0.908	0.904	0.900
$prop_{11}$:training	**0.778**	**0.729**	**0.641**	**0.629**	**0.626**	**0.594**	**0.585**	**0.578**	**0.570**	**0.474**
$prop_{12}$:previousWork	0.955	0.941	0.938	0.931	0.931	0.931	0.927	0.924	0.921	0.919
$prop_{13}$:recordLabel	0.959	0.948	0.938	0.930	0.925	0.920	0.917	0.912	0.906	0.905
$prop_{14}$:starring	0.988	0.977	0.964	0.953	0.942	0.933	0.926	0.918	0.910	0.904
$prop_{15}$:country	0.943	0.929	0.920	0.908	0.900	0.891	0.886	0.882	0.875	0.873
$prop_{16}$:city	0.960	0.942	0.927	0.914	0.904	0.896	0.889	0.882	0.878	0.874
$prop_{17}$:associatedBand	0.963	0.939	0.926	0.915	0.911	0.904	0.899	0.893	0.889	0.885
$prop_{18}$:fromAlbum	0.967	0.953	0.942	0.932	0.923	0.913	0.901	0.885	0.872	0.869
$prop_{19}$:location	0.967	0.945	0.927	0.911	0.898	0.888	0.881	0.872	0.866	0.861
$prop_{20}$:notableWork	0.976	0.962	0.954	0.950	0.948	0.940	0.933	0.934	0.934	0.931

Table 3: Gini index of the properties

Property	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
$prop_9$	0.678	0.655	0.630	0.627	0.643	0.649	0.661	0.652	0.662	0.680
$prop_{11}$	0.778	0.729	0.650	0.636	0.648	0.636	0.623	0.616	0.607	0.509

Table 4: Gini index of the properties $prop_9$: debutstadium and $prop_{11}$: training with $maxRecom$=5

7 Conclusions and Further Work

In this paper, we introduce the information gap between Wikipedia and DBpedia. To reduce this gap, we have to discover Wikipedia conventions to represent a DBpedia property between a pair of Wikipedia articles. We propose BlueFinder, a collaborative recommender system that recommends navigational paths to represent a DBpedia property in Wikipedia, while respecting Wikipedia conventions. BlueFinder learns from those similar pairs already connected by Wikipedia community and proposes a set of recommendations to connect a pair of disconnected articles. BlueFinder exploits DBpedia types to define a similarity function. Experimental results demonstrate that BlueFinder is able to fix in average 89 % of the disconnected pairs with good accuracy and confidence.

Currently, BlueFinder is tailored for Wikipedia/DBpedia where entities matching are well-defined. However, BlueFinder can be generalized to other datasets with established entities matching.

As a further work, we plan to update Wikipedia with BlueFinder recommendations. We have detected more than 50 % of the recommendation are *featured recommendations*. This means around 270,367 new links will be added to Wikipedia. The future work will be based on a crowdsourcing activity and a monitoring program which evaluates the community agreement of the new connections. Moreover, we are going to adapt this approach in combination with

non-English versions of Wikipedia. Finally, we plan to extend the approach to any property in DBpedia in combination with other languages of Wikipedia and to offre the next generation of BlueFinder as a service for any Wikipedia editor.

Acknowledgements This work is supported by the French National Research agency (ANR) through the KolFlow project (code: ANR-10-CONTINT-025), part of the CONTINT research program.

References

1. Lu, C., Stankovic, M., Laublet, P.: Desperately searching for travel offers? formulate better queries with some help from linked data. In: The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings. (2015) 621–636
2. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal **6**(2) (2015) 167–195
3. Torres, D., Molli, P., Skaf-Molli, H., Díaz, A.: Improving Wikipedia with DBpedia. In Mille, A., Gandon, F.L., Misselis, J., Rabinovich, M., Staab, S., eds.: WWW (Companion Volume), ACM (2012) 1107–1112
4. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of sparql. ACM Trans. Database Syst. **34**(3) (September 2009) 16:1–16:45
5. Landauer, T.K., Nachbar, D.: Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width. ACM SIGCHI Bulletin **16**(4) (1985) 73–78
6. Larson, K., Czerwinski, M.: Web page design: implications of memory, structure and scent for information retrieval. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '98, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (1998) 25–32
7. Otter, M., Johnson, H.: Lost in hyperspace: metrics and mental models. Interacting with computers **13**(1) (2000) 1–40
8. Torres, D., Molli, P., Skaf-Molli, H., Diaz, A.: From dbpedia to wikipedia: Filling the gap by discovering wikipedia conventions. In: 2012 IEEE/WIC/ACM International Conference on Web Intelligence (WI'12). (2012)
9. Torres, D., Skaf-Molli, H., Molli, P., Diaz, A.: BlueFinder: Recommending Wikipedia Links Using DBpedia Properties. In: ACM Web Science Conference 2013 (WebSci 13), Paris, France (May 2013)
10. Wang, Y., Wang, H., Zhu, H., Yu, Y.: Exploit semantic information for category annotation recommendation in wikipedia. Natural Language Processing and Information Systems (2007) 48–60
11. Mirizzi, R., Di Noia, T., Ragone, A., Ostuni, V.C., Di Sciascio, E.: Movie recommendation with dbpedia. In: IIR, Citeseer (2012) 101–112
12. Panchenko, A., Adeykin, S., Romanov, A., Romanov, P.: Extraction of semantic relations between concepts with knn algorithms on wikipedia. In: Proceedings of Concept Discovery in Unstructured Data Workshop (CDUD) of International Conference On Formal Concept Analysis. (2012) 78–88

13. Singer, P., Niebler, T., Strohmaier, M., Hotho, A.: Computing semantic relatedness from human navigational paths: A case study on wikipedia. International Journal on Semantic Web and Information Systems (IJSWIS) **9**(4) (2013) 41–70
14. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D., Zanker, M.: Linked open data to support content-based recommender systems. In: 8th International Conference on Semantic Systems (I-SEMANTICS 2012). ICP, ACM Press (2012)
15. Nunes, B.P., Dietze, S., Casanova, M.A., Kawase, R., Fetahu, B., Nejdl, W.: Combining a co-occurrence-based and a semantic measure for entity linking. In: The Semantic Web: Semantics and Big Data. Springer (2013) 548–562
16. Adafre, S.F., de Rijke, M.: Discovering missing links in wikipedia. In: Proceedings of the 3rd international workshop on Link discovery. LinkKDD '05, New York, NY, USA, ACM (2005) 90–97
17. Sunercan, O., Birturk, A.: Wikipedia missing link discovery: A comparative study. In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence, AAAI (2010)
18. Hoffmann, R., Amershi, S., Patel, K., Wu, F., Fogarty, J., Weld, D.S.: Amplifying community content creation with mixed initiative information extraction. In: Proceedings of the 27th international conference on Human factors in computing systems. CHI '09, New York, NY, USA, ACM (2009) 1849–1858
19. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th international conference on World Wide Web. WWW '07, New York, NY, USA, ACM (2007) 697–706
20. Alkhateeb, F., Baget, J.F., Euzenat, J.: Extending sparql with regular expression patterns (for querying rdf). Web Semantics: Science, Services and Agents on the World Wide Web **7**(2) (2011)
21. Abiteboul, S., Vianu, V.: Regular path queries with constraints. In: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. PODS '97, New York, NY, USA, ACM (1997) 122–133
22. Arenas, M., Conca, S., Pérez, J.: Counting beyond a yottabyte, or how sparql 1.1 property paths will prevent adoption of the standard. In: Proceedings of the 21st international conference on World Wide Web. WWW '12, New York, NY, USA, ACM (2012) 629–638
23. Adomavicius, G., Tuzhilin, A.: Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering **17**(6) (2005) 734–749
24. Jaccard, P.: Nouvelles recherches sur la distribution florale. Bulletin de la Sociète Vaudense des Sciences Naturelles **44** (1908) 223–270
25. Lu, W., Shen, Y., Chen, S., Ooi, B.: Efficient processing of k nearest neighbor joins using mapreduce. Proceedings of the VLDB Endowment **5**(10) (2012) 1016–1027
26. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS) **22**(1) (2004) 143–177
27. O'Sullivan, D., Smyth, B., Wilson, D.C., Mcdonald, K., Smeaton, A.: Improving the quality of the personalized electronic program guide. User Modeling and User-Adapted Interaction **14**(1) (2004) 5–36
28. Fleder, D.M., Hosanagar, K.: Recommender systems and their impact on sales diversity. In: Proceedings of the 8th ACM conference on Electronic commerce, ACM (2007) 192–199
29. Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Recommender systems handbook. Springer (2011) 257–297