BlueFinder: Recommending Wikipedia Links Using DBpedia Properties

Diego Torres[*]
LIFIA, Fac.
Informática, UNLP
50 y 120, S/N
1900 La Plata,
Argentina

Hala Skaf-Molli[†]
LINA, Nantes
University
2, rue de la
Houssiniere
44322 Nantes, France

Pascal Molli[‡]
LINA, Nantes
University
2, rue de la
Houssiniere
44322 Nantes, France

Alicia Díaz[§]
LIFIA, Fac.
Informática, UNLP
50 y 120, S/N
1900 La Plata,
Argentina

ABSTRACT

DBpedia knowledge base has been built from data extracted from Wikipedia. However, many existing relations among resources in DBpedia are missing links among articles from Wikipedia. In some cases, adding these links into Wikipedia will enrich Wikipedia content and therefore will enable better navigation. In previous work, we proposed PIA algorithm that predicts the best link to connect two articles in Wikipedia corresponding to those related by a semantic property in DBpedia and respecting the Wikipedia convention. PIA calculates this link as a path query. After introducing PIA results in Wikipedia, most of them were accepted by the Wikipedia community. However, some were rejected because PIA predicts path queries that are too general. In this paper, we report the BlueFinder collaborative filtering algorithm that fixes PIA miscalculation. It is sensible to the specificity of the resource types. According to the conducted experimentation we found out that BlueFinder is a better solution than PIA because it solves more cases with a better recall.

ACM Classification Keywords

H.3.3 Information Search and Retrieval: Information filtering

General Terms

Algorithms, Experimentation

Author Keywords

DBpedia, Wikipedia, Recommendation

[*]diego.torres@lifia.info.unlp.edu.ar

[†]Hala.Skaf@univ-nantes.fr

[‡]Pascal.Molli@univ-nantes.fr

[§]alicia.diaz@lifia.info.unlp.edu.ar

WebSci'13, May 1 – May 5, 2013, Paris, France.
Copyright 2013 ACM 978-1-4503-1889-1 . . . $ 10.00

INTRODUCTION

The Semantic Web brings the ability to have better search and navigability on the Web. It is mainly build from metadata extracted from the Social Web. DBpedia [8] knowledge base is built from data extracted from Wikipedia [1] infoboxes and categories. The semantic capacities of DBpedia enable SPARQL [16] queries to retrieve information that is not present in Wikipedia [21]. For instance, it is possible to make a query over DBpedia to find *"everyone in Wikipedia that born in Boston"*. This query produces a set of couples *(Boston, Person Name)* which are related by the semantic property of DBpedia *"is birth place of"*. Surprisingly, the list of people retrieved by the previous query could include more people than those obtained by navigating from Boston[1] article in Wikipedia.

This shows that some information existing in DBpedia is missing in Wikipedia. The question now is: *Is it really necessary to add this information to Wikipedia?* In some cases, the missing links could be intentionally hidden for the simplicity of the page content[2]. In other cases, adding these links enables to enrich Wikipedia content and enables better navigation. However, adding these missing links to Wikipedia is not an easy task since it is necessary to respect Wikipedia conventions[3].

Wikipedia community has defined conventions that cover a wide diversity of topics: writing style, context of the articles and relations among articles. It is not always evident to understand these conventions. *Categories* or *list of pages* are the conventions to describe one-to-many relationships. At first glance, the use of categories is the convenient convention to represent the relationship *"is birth place of"* in Wikipedia. The category `Cat:People_from_<cityName>` is the general rule of this relationship and it is usually subcategory of `Cat:<cityName>`. For example, the *"is birth place of"* relation between *Boston*[4] and *Dona_Summer*[5] is described using the category `People_from_Boston`. Finally, the

[1] In the scope of this article we will use Boston instead of Boston, Massachusetts

[2] **http://en.wikipedia.org/wiki/Wikipedia:OVERLINK**

[3] **http://en.wikipedia.org/wiki/Wikipedia: Conventions**

[4] **http://en.wikipedia.org/wiki/Boston**

[5] **http://en.wikipedia.org/wiki/Donna_Summer**

relation is represented by the navigational path Boston/Cat:Boston/Cat:People_from_Boston/Dona_Summer which must be read: *"from Boston article, the user navigates through a link to the category Boston then he or she can navigate to the People_from_Boston category and then to the Dona_Summer article"*.

In previous work [21, 20], we proposed Path Index Algorithm (PIA) that allows to discover the Wikipedia conventions. PIA predicts the best representation in Wikipedia for a given semantic property in DBpedia. To evaluate the results obtained by PIA, we added manually the new discovered links to Wikipedia, therefore, Wikipedia community was able to evaluate these new added links. After two months, we noticed that some of the added links were accepted and other were rejected. When we analyzed the rejected results, we noticed that these links were very general. For example, the article *Liverpool* and *Chris_Lawler*[6] must be connected, according to PIA, by the navigational path Liverpool/Cat:Liverpool/Cat:People_from_Liverpool/Chris_Lawler. In general, Wikipedia conventions suggest to categorise people from Liverpool with the category Cat:People_from_Liverpool but there are exceptions. For example, the conventions propose Cat:Sportspeople_from_Liverpool for an athlete from Liverpool. Although this path query appears in the index generated by PIA, it is placed at the bottom of the index. The main drawback of PIA is that it does not differentiate the different types of people born in Liverpool. More generally, PIA does not pay special attention to the specificity of the resource and then, it can predict the specific category.

In this paper, we introduce the BlueFinder algorithm that filters the results of PIA to recommend better representation for a DBpedia semantic property in Wikipedia. BlueFinder pays special attention to the specificity of the resource types in DBpedia in order to fix PIA drawback. It follows a collaborative filtering approach [13] to recommend a relation between a given couple of Wikipedia pages and a given semantic property.

The experimentations of the algorithm with real data from Wikipedia and DBpedia showed that BlueFinder is better than PIA. BlueFinder increases the number of correct path obtained by the previous approach PIA.

The main contributions of this paper are: (1) the definition of BlueFinder algorithm as a collaborative filtering recommender system which is sensible to articles type specificity, (2) a similarity function between pairs of related articles and (3) an empirical evaluation of BlueFinder algorithm.

This paper is organized as follows: The next section presents related works. Then, we introduce preliminaries definitions used in our proposal. After that, the article describes the problem statement. Then, BlueFinder section details the BlueFinder approach and algorithms. Following, a section presents experimentations and results. Finally, conclusions and further work are presented.

RELATED WORK

In previous work, we introduced PIA algorithm [20] which retrieves the best general representation in Wikipedia for a given DBpedia semantic property. If we apply PIA to the *"is birth place of"* property, we will obtain the path #from/Cat:#from/Cat:People_from_#from/#to as the best answer. To obtain the most representative path, PIA builds a path index where the paths are sorted by coverage of the connected path queries (more details are given in the next section). A path query is a generalisation of similar paths. Usually, regular expressions are used for expressing path queries [5]. The path query that covers most of the cases will be the most representative. Table 1 shows some results of PIA. The analysis of the generated PIA index showed us the following facts: (1) the most representative path is #from/Cat:#from/Cat:People_from_#from/#to; (2) the path #from/Cat:#from/Cat:People_from_#from/Cat:Sportspeople_from_#from/#to was detected by PIA but it appears so far from the top of the index (rank 1092); (3) in rank 75 appears a path that contains the Cat:Sportspeople_from_#from at penultimate position. This different paths with the same ending confirm that the conventions are also present in the development of category hierarchy. In some cases, the Cat:Sportspeople_from_#from belongs to Cat:People_from_#from and in other cases it belongs to Cat:People_from_#from_by_occupation.

The work of Yan Wang et al. [22] introduces a "collaborative approach" to recommend categories to Wikipedia Articles. The approach consists of a two-step model. In the first step, they collect similar articles to the uncategorized one in terms of incoming and out coming links, headings and templates. The second step lies on rank the categories obtained by the related articles and select the best ranked. In this work, we deal with categorization of the article but in the context of expressing a semantic property from DBpedia, this feature is absent in Yan Wang et al. approach.

Less related to our approach but in the line of combining recommender systems and DBpedia,*MORE* is a recommender system that uses DBpedia and Wikipedia to recommend movies. MORE uses RDF datasets to compute semantic similarity between movies. Panchenko et al. [15] introduces an approach to extract semantic relations between con-

Rank	Path Query
1	#from/ Cat:#from/ Cat:Peopel_from_#from/ #to
2	#from/ #to
...	...
75	#from/ Cat:#from/Cat:People_from_#from_by_occupation/Cat:Sportspeople_from_#from/#to
...	...
1092	#from/Cat:#from/Cat:People_from_#from/Cat:Sportspeople_from_#from/#to

Table 1. Extract of PIA index for *"is birth place of"* example.

[6] http://en.wikipedia.org/wiki/Chris_Lawler

cepts 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. Additionally to the lack of using DBpedia as semantic base, Serelex cannot describe the way that two concepts are related in Wikipedia according to a semantic property.

In the field of improving Wikipedia information, several related works can be mentioned. The articles of Adafre et al [3] and Sunercan et al. [18] are aimed to fix missing direct links in Wikipedia. Hoffman et al. [12] introduced an approach to complete Wikipedia infobox links with information extracted from Wikipedia by using Kylin. The main difference with our work is that they do not use semantic Web features and also our approach propose to fix more general relations than direct links.

PRELIMINARIES

DBpedia and Wikipedia provide data sets for the BlueFinder recommender system. In this section, we give formal definitions for this two data sources.

DBpedia Knowledge base

The knowledge base of DBpedia has rich set of properties. In addition to specific properties (birthPlace, city, ...), each resource in DBpedia has types definition coming from DBpedia ontology and Yago [17] ontology. For instance, the *rdf:type* describes resources types. This knowledge provides datasets for the BlueFinder recommender through SPARQL queries. Basically, DBpedia knowledge base is an RDF graph without Blank nodes. We recall the following RDF semantics definitions [16] :

DEFINITION 1. *The Sets I (IRI Identifiers), B (Blank Nodes), L (Literals) and Υ (Variables) are four infinite and pairwise disjoint sets. We also define $T = I \cup B \cup L$. An* RDF-Triple *is 3-tuple $(s, p, o) \in (I \cup B) \times I \times T$. An* RDF-Graph *is a set of RDF-Triples.*

DEFINITION 2. *A* triple pattern *is a tuple $t \in (I \cup \Upsilon \cup L) \times (I \cup \Upsilon) \times (I \cup \Upsilon \cup L)$. A* Basic Graph Pattern *is a finite set of triple patterns. Given a triple pattern t, $var(t)$ is the set of variables occuring in t, analogously, given a basic graph pattern B, $var(B) = \cup_{t \in B} var(t)$. Given two basic graph patterns B_1 and B_2, the expression $B_1\ AND\ B_2$ is a graph pattern.*

DEFINITION 3. *A* mapping *μ from Υ to T is a partial function $\mu : \Upsilon \to T$. The domain of μ, $dom(\mu)$, is the subset of Υ where μ is defined.*

DEFINITION 4. *Given a triple pattern t and a mapping μ such that $var(t) \subseteq dom(\mu)$, $\mu(t)$ is the triple obtained by replacing the variables in t according to μ. Given a basic graph pattern B and a mapping μ such that $var(B) \subseteq dom(\mu)$, then $\mu(B) = \cup_{t \in B} \mu(t)$.*

DEFINITION 5. *Two mappings μ_1, μ_2 are* compatible *(we denote $\mu_1 \shortparallel \mu_2$) iff for all $?X \in (dom(\mu_1) \cap dom(\mu_2))$, then $\mu_1(?X) = \mu_2(?X)$. This is equivalent to say that $\mu_1 \cup \mu_2$ is also a mapping.*

Two important corollaries of this last definition are: *i*) two mappings with disjoint domains are always compatible, *ii*) the empty mapping (the one with empty domain) is compatible with any other mapping.

DEFINITION 6. *Let Ω_1, Ω_2 two sets of mappings. The join between Ω_1 and Ω_2 is defined as: $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \wedge \mu_2 \in \Omega_2 \wedge \mu_1 \shortparallel \mu_2\}$*

DEFINITION 7. *Given an RDF-Graph G, the* evaluation *of a triple pattern t over G corresponds to: $[[t]]_G = \{\mu \mid dom(\mu) = var(t) \wedge \mu(t) \in G\}$. The* evaluation *of a basic graph pattern B over G is defined as: $[[B]]_G = \bowtie_{t \in B} [[t]]_G$. The* evaluation *of a Graph Pattern B' of the form $(B_1$ AND $B_2)$ over G is as follows: $[[B']]_G = [[B_1]]_G \bowtie [[B_2]]_G$*

Consider the following SPARQL query over a data set D: $Q =$ SELECT x_i, y_i WHERE $x_1 p_1 y_1$ AND ... AND $x_n p_n y_n$

The answer for this query $Q(D)$ is an assignment of distinguished variables (those variables in the SELECT part of the query) i.e. the *e*valuation of a triple pattern t over D. For instance, the following SPARQL query over DBpedia gives a couple of Wikipedia pages of people and their birth place.

```
PREFIX o:<http://dbpedia.org/ontology/>
PREFIX d:<http://dbpedia.org/property/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>

#Q1: Cities and People born there.
SELECT ?wcity, ?wperson WHERE{
?person a o:Person.
?city a o:City.
?person d:birthPlace ?city.
?city foaf:isPrimaryTopicOf ?wcity.
?person foaf:isPrimaryTopicOf ?wperson.}
```

Listing 1. SPARQL query over DBpedia

In this work, we use SPARQL queries that have a triple pattern of the form: s d:property o where d:property is a specific DBpedia property for s. For instance, in the previous query the property p is $d : birthPlace$, we denote the result of the query by $Q_p(D)$.

Wikipedia Formal Definition

Wikipedia provides data set for the PIA algorithm. Path queries [2, 5] are used to query Wikipedia. To define path queries, we need to introduce the following definitions. Wikipedia consists of a set of articles and hyperlinks among them. The Wikipedia graph G can be defined as:

DEFINITION 8. *$G = \{W, E\}$ 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.*

DEFINITION 9. *A 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 path is the number of articles in the sequence, length $P(w_1, w_n) = n$.*

Given a $Q_p(D)$, the set of all pairs $(f, t) \in Q_p(D)$ that are connected in Wikipedia by a path with length up to l is defined as:

DEFINITION 10. *$C_p(l) = \{(f, t) \in Q_p(D)$ s.t. $\exists P(f, t)$ and $length(P(f, t)) <= l\}$*

A path query is a generalization of similar paths. Usually, regular expressions are used for expressing path queries [5]. Informally, the set of answers of a path query $PQ(w_1, w_2)$ over G is the set of all pairs of nodes in G connected by a directed path such that the concatenation of the labels of the nodes along the path forms a word that belongs to the language denoted by L^*. Many works have been done on path queries in different domains [2, 5, 6]. We adapt the path query definition in [2, 5] to the context of Wikipedia.

Let Σ be an alphabet, a language over Σ is a sequence of elements of Σ called words. Regular expressions can be used to define language over Σ. We use regular expression patterns [5] i.e. patterns that include variables. Let X be a set of variables.

DEFINITION 11. *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*

The language defined by a regular expression pattern is:

DEFINITION 12. *Let $R, R' \in R(\Sigma, X)$ 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^* $(R^+) = \{w_1 \ldots w_k \,|\, \forall i \in [1...k], w_i \in L^*(R)$;(6) $L^*(R^*) = \{\epsilon\} \bigcup L^*(R+)$.*

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

DEFINITION 13. *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 $\Lambda(P) = \Lambda(a_1) \ldots \Lambda(a_k)$ 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`$\}$. 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. The PIA algorithm uses path queries and computes the coverage of path queries for a set of pairs of Wikipedia articles.

DEFINITION 14 (PIA INDEX). *Given a $C_p(l)$, PIA index 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), $I = PQ \times C_p(l)$ 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 semantic property p in Wikipedia.*

Table 1 gives the rank, degree and path queries of PIA index for the property "is birth place of" of the example given in the introduction.

PROBLEM STATEMENT

In this section, we are going to define the problem of defining the best representation of missing links in Wikipedia as a recommender system problem. According to Adomavicioius and Tuzhilin [4], *"collaborative recommender systems try to predict the utility of items for a particular user based on the items previously rated by other users"*. More formally, the utility $u(c, s)$ 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, we do not directly transpose recommenders to suggest Wikipedia articles to users but to suggest links between articles. We want to predict the utility of path queries for a particular pair of Wikipedia articles based on those rated by 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 totally ordered set and l is the maximum length of the path queries[7].

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 maximise the utility function.

BLUEFINDER

BlueFinder is a collaborative filtering recommender system. It uses a memory based algorithm [9] to make rating predictions based on the entire collection of previously rated path queries. The value of the unknown rating $r_{c,s}$ for a pair c and path query s will be computed as an aggregate rating of other k similar pairs for the same path query s. The recommender returns a set of recommended path queries that can be used to represent the semantic property. The recommendations have to include at least one path query that can represent the semantic relation following the conventions of Wikipedia community. BlueFinder is based on the popular k-Nearest Neighbors(kNN) and Multi label kNN algorithm [23] adapted to the context of DBpedia and PIA index. The BlueFinder algorithm first identifies the k neighbors of the unconnected pair (from, to), and then applies PIA algorithm only for the k nearest neighbors. PIA results will be the prediction set.

The K-Nearest Neighbors algorithm uses a similarity measure function to select the nearest neighbors. In this work we use the well known Jaccard distance [19] to measure similarity. The Jaccard distance measures the degree of overlap of two sets and ranges from 0 (identical sets) to 1 (disjoint sets).

[7] We restrict the length for practical reasons.

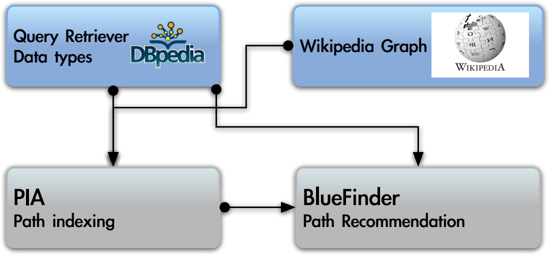

Figure 1. BlueFinder overview

In this work, we apply Jaccard distance to types of DBpedia resources.

DEFINITION 15. *Given two pairs of pages $c_1 = (a, b)$ and $c_2 = (a', b')$ and the data type set for b and b' in DBpedia defined as $B = \{t/\ b\ rdf\ :\ type\ t\ \} \in$ DBpedia and $B' = \{t'/\ b'$ rdf:type $t'\} \in$ DBpedia }. The similarity measure $jccD(c_1, c_2)$ is defined by*

$$jccD(c_1, c_2) = Jaccard \quad distance(B, B') = \frac{|B \cup B'| - |B \cap B'|}{|B \cup B'|}$$

Now, we can define the kNN [14] in our context as:

DEFINITION 16. *(KNN) Given a pair $r \in Q_p(D)$ and an integer k, the k nearest neighbors of r denotes $KNN(r, Q_p(D))$ 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 $jccD(o, r) \leq jccD(s, r)$.*

The value for an unknown rating $r_{c,s}$ for unconnected pair in Wikipedia c and a path query $s \in C_p(l)$, can be computed as:

$r_{c,s}$ =degree of s in PQ,
where $(PQ, V, I) = PIA(KNN(c, C_p(l))$

In order to compute predictions for $r_{c,s}$, we developed the BlueFinder algorithm.

BlueFinder algorithm

The BlueFinder algorithm works as a pipeline process, it takes DBpedia, Wikipedia and PIA as inputs. Figure 1 shows the relations and pipe data flow among the artefacts. DBpedia interacts with PIA and BlueFinder algorithm. The interaction with PIA is done by providing the $Q_p(D)$ result set. On the other hand, the BlueFinder algorithm asks DBpedia to obtain the type definition for specific resources. For a particular resource, DBpedia returns a set with all the types describing the resource.

The *BlueFinder* algorithm returns a set of recommended path queries that can be used to represent the semantic property between two Wikipedia articles. The recommendations have to include at least one path query that can represent the semantic relation following the conventions of Wikipedia community. This algorithm, applies PIA index algorithm to the k nearest neighbours of a unconnected pair (from,to) and retrieves a set of predicted path queries.

The BlueFinder algorithm 1 receives four inputs: (1) the number of k neighbours, (2) the maximum number $maxRecom$ of recommendations, (3) a PIA index for a $Q_p(D)$ and (4) the unconnected pair x of Wikipedia articles. For each connected pair $c \in C_p(l)$, the algorithm computes the $jccD(c, x)$ distance. Then, the k nearest pairs of x is added to the set D_x^k and the PIA index is computed for D_x^k. Before returning the recommendations, BlueFinder cleans regular-user unreachable paths (e.g. paths that include administrative categories) by means of the noiseFilter (Algorithm 2) and groups similar path queries (Algorithm 3). Finally, BlueFinder returns the $maxRecom$ best ranked path queries.

It is important to notice that $D_x^k \subseteq C_p(l)$, therefore, when PIA algorithm computes a new PIA index for the D_x^k set of connected pairs, it does not need to traverse again Wikipedia graph.

Algorithm 1 BlueFinder

```
Input: $x = (s, t)$: unconnected pair, $k$: number of neighbours, $maxRecom$: maximum number of recommendation, $PiaIndex$: PIA index for a $Q_p(D)$ where $PiaIndex$ defined as $(\mathcal{PQ}(s), C_p(l) = \{(source, target)\}, I)$,
Output: $M \subseteq \mathcal{PQ}(s)$
  for all $(source, target) \in C_p(l) : \exists(pq, (source, target)) \in I$ do
      $d$= $jccD(target, t)$
      $res$ = $(pq, (source, target), d)$
  end for
  sort $res$ in ascending order
  $D_x^k \leftarrow k$ nearest pairs to $x$
  $(M, C_p(l)', I') \leftarrow PIA(D_x^k)$
  $M \leftarrow noiseFilter(M)$
  $M \leftarrow starGeneralization(M)$
  return first $maxRecom$ path queries of $M$
```

The $noiseFilter$ algorithm 2 deletes all the paths queries that are not accessible by a regular user. Wikipedia includes several administrative categories which are used by administrators. Although the list is not exhaustive, the categories deleted by $noiseFilter$ are those that their names begins with "Articles_", "All_Wikipedia_", etc. For example `Cat:Articles_to_be_merged`.

Algorithm 2 noiseFilter

```
Input: $PQ$: set of path queries
Output: 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 17. *A star path query PQ*(f,t)) is a group of similar path queries and it respects the following construction rules: (1) it starts with `#from` and ends with `#to`. (2) The * element can only be placed between of `#from` and `#to` variables. (3) The * can not be the penultimate element in the path query.*

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.*

$starGeneralization$ algorithm 3 groups path queries into star path query, if possible.

```
Algorithm 3 starGeneralization
Input: $PQ$: set of path queries
Output: $PQ^*$: 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 $PQ^*$
```

Discussion

Traditionally, the kNN-based methods have the advantage of being relatively simple to implement and adapt quickly to recent changes. When compared with other learning approaches, a relatively small number of ratings is sufficient to make a prediction of reasonable quality [13]. kNN algorithms are tolerant to a low sparsity data sets, however with high level of sparsity the kNN starts failing as it is unable to form reliable neighbourhoods [11]. In our approach, the problems related with sparsity occur when the data type definition of a resource in DBpedia is the basic one i.e. the resource has no special type definition that differentiates it from other resources. In this case, the $jccD$ similarity measure is limited in the neighbourhood discovery. In addition, as kNN is a lazy algorithm, adding a new item (path query) or a new user (pair) is not a problem. However, the addition of this new elements require PIA algorithm to re-processing data.

EXPERIMENTATIONS

To evaluate the BlueFinder algorithm and to measure the impact of the parameters,we conducted pragmatical experimentations with real data. This evaluation mainly allows us to answer the following three questions: What is the best combination of k and $maxRecom$ values to get the best behaviour of the BlueFinder algorithm? Does the algorithm retrieves path queries that can fix missing relations in Wikipedia? Does the retrieved path queries take in consideration Wikipedia articles context?

We implemented BlueFinder in Java. It interacts with a PIA index stored in a Mysql database (computed in previous work [21, 20]) using an off line version of English Wikipedia[8] and with a local version of DBpedia knowledge base. All the algorithms were executed in a MackBook Pro with 8GB RAM and a processor Intel core I7. During the evaluation, we also computed the time processing for each data set.

In the next section, we detail the data sets, metrics, methodology and the evaluation results.

Data sets

Data set	$\|PQ\|$	$\|C_p(l)\|$	$\|I\|$
$Q_{birthplace}(D)$	8,118	65,200	211,654
$Q_{city}(D)$	6,623	9,497	31,623

Table 4. PIA index data sets used in the experimentation. Columns shows the number of path queries, the number of connected pairs and the number of edges

We choose two properties of DBpedia: birthPlace and city. We run $Q_{birthPlace}$ and Q_{city} given in Listing 2. We used one PIA index for $Q_{birthPlace}(D)$ and another for $Q_{city}(D)$. Table 4 details these index. We run the BlueFinder prototype on the same data sets as in PIA.

```
PREFIX o:<http://dbpedia.org/ontology/>
PREFIX p:<http://dbpedia.org/property/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>

#Qbirthplace: Cities and People born there.
SELECT ?wcity, ?wperson WHERE{
?person a o:Person.
?city a o:City.
?person p:birthPlace ?city.
?city foaf:isPrimaryTopicOf ?wcity.
?person foaf:isPrimaryTopicOf ?wperson.}

#Qcity: Cities and its universities.
SELECT ?wcity, ?wuniversity WHERE{
?university a o:University.
?city a o:City.
?university p:city ?city.
?city foaf:isPrimaryTopicOf ?wcity.
?university foaf:isPrimaryTopicOf ?wuniversity.}
```

Listing 2. Semantic Queries of the Evaluation

Metrics

We use *precision*, *recall* [7], and F_1 *score* [10] as metrics. They compute the proportion between the BlueFinder retrieved path queries (BFPQs) and the relevant or correct path queries (CPQs). The precision is the proportion of retrieved path queries that are correct with respect to connected Wikipedia articles (1), recall is the proportion of correct path queries that are retrieved (2). Finally, F_1 *score* is the combination of precision and recall (3).

$$(1)precision(CPQs, BFPQs) = \frac{|CPQs \bigcap BFPQs|}{|BFPQs|}$$

$$(2)recall(CPQs, BFPQs) = \frac{|CPQs \bigcap BFPQs|}{|CPQs|}$$

$$(3)F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Additionally, we want to measure the number of cases where BlueFinder recommends at least one correct path query that can be fix the missing links. With this value we can obtain a rate of cases where BlueFinder potentially fix the missing connection in Wikipedia. This measure is defined by the pot function:

$$pot(CPQs, BFPQs) = \begin{cases} 0 & \text{if } CPQs \bigcap BFPQs = \emptyset \\ 1 & \text{otherwise} \end{cases}$$

Methodology

To answer the questions above, we follow the steps below:

1. Evaluation setup. The first step defines the parameters to exercise BlueFinder algorithm. We define different values of k from 1 to 10 and different values of $maxRecom$=1;3;5,

[8] http://dumps.wikimedia.org/enwiki/20111007/

	K	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PIA	$precision_k$	0.587									
	$recall_k$	0.518									
	$F1_k$	0.550									
	pot_k	0.585									
maxRecom=1	$precision_k$	0.487	0.515	0.520	0.527	0.541	0.525	0.551	0.558	0.562	**0.571**
	$recall_k$	**0.453**	0.483	0.487	0.5	0.506	0.510	0.516	0.524	0.527	0.535
	$F1_k$	0.453	0.483	0.487	0.500	0.506	0.510	0.516	0.524	0.527	0.535
	pot_k	**0.488**	0.516	0.522	0.534	0.541	0.545	0.552	0.558	0.562	0.570
maxRecom=3	$precision_k$	0.479	0.456	0.423	0.393	0.376	0.358	0.346	0.337	0.326	0.321
	$recall_k$	0.462	0.596	0.653	0.678	0.697	0.697	0.695	0.693	0.695	0.698
	$F1_k$	0.453	0.483	0.487	0.500	0.506	0.510	0.516	0.524	0.527	0.535
	pot_k	0.517	0.650	0.704	0.728	0.748	0.748	0.747	0.744	0.747	0.753
maxRecom=5	$precision_k$	0.479	0.455	0.421	0.385	0.362	0.339	0.323	0.307	0.289	**0.280**
	$recall_k$	0.601	0.708	0.758	0.785	0.807	0.820	0.828	**0.832**	0.828	0.828
	$F1_k$	0.533	0.554	0.541	0.517	0.500	0.480	0.465	0.449	0.429	0.417
	pot_k	0.517	0.652	0.709	0.747	0.775	0.793	0.809	**0.811**	0.809	**0.811**

Table 2. $Q_{birthplace}(D)$ **data set experiment values.**

	K	1	2	3	4	5	6	7	8	9	10
PIA	$precision_k$	0.636									
	$recall_k$	0.418									
	$F1_k$	0.505									
	pot_k	0.642									
maxRecom=1	$precision_k$	0.282	0.229	0.191	0.191	0.249	0.269	0.286	0.303	0.316	0.333
	$recall_k$	0.19	0.15	**0.123**	0.133	0.171	0.181	0.191	0,205	0,214	0.224
	$F1_k$	0.228	0.181	0.15	0.157	0.202	0.217	0.229	0.244	0.255	0.268
	pot_k	0.284	0.232	0.194	0.211	0.253	0.281	0.298	0.312	0.326	0.343
maxRecom=3	$precision_k$	**0.344**	0.308	0.274	0.260	0.261	0.257	0.257	0.259	0.257	0.249
	$recall_k$	0.345	0.431	0.451	0.454	0.459	0.460	0.469	0.473	0.470	0.461
	$F1_k$	0.344	0.359	0.341	0.331	0.333	0.330	0.332	0.335	0.333	0.324
	pot_k	0.486	0.583	0.604	0.607	0.604	0.611	0.621	0.625	0.625	0.618
maxRecom=5	$precision_k$	**0.344**	0.311	0.268	0.241	0.229	0.217	0.203	0.199	0.197	0.201
	$recall_k$	0.470	0.542	0.579	0.593	0.615	0.618	0.614	0.604	0.606	**0.636**
	$F1_k$	0.397	0.395	0.366	0.343	0.334	0.322	0.305	0.3	0.297	0.306
	pot_k	0.486	0.607	0.645	0.652	0.666	0.670	0.652	0.642	0.659	**0.708**

Table 3. $Q_{city}(D)$ **data set experiment values.**

N is the set of connected pairs in Wikipedia to evaluate BlueFinder and the PIA index for each data set.

2. Exercise the BlueFinder on known cases. We eliminate links that already exist in the PIA index and then we observe if BlueFinder is able to recreate them. Concretely, for each connected pair in N, we generate a mock PIA index without the path queries of the pair i.e. we *"disconnect"* the pair. After that, we exercise BlueFinder to "fix" the pair by using the mock PIA index. Then, we compare the recommendations with the path queries of PIA.

Consequently, the $recall$, $precision$ and pot for the BlueFinder using k neighbours and N is calculated with the following functions:

$recall_k = \frac{1}{N}\sum_{i=1}^{N} recall[i,k]$ where $recall[i,k]$ is the recall value for the i^{th} pair extracted by BlueFinder exercised with k-nearest neighbours.

$precision_k = \frac{1}{N}\sum_{i=1}^{N} precision[i,k]$ where $precision[i,k]$ is the precision for the i^{th} pair extracted by BlueFinder exercised with k-nearest neighbours.

$pot_k = \frac{1}{N}\sum_{i=1}^{N} pot[i,k]$ is the pot value for the i^{th} pair extracted by BlueFinder exercised with k-nearest neighbours.

For $Q_{birthplace}(D)$ data set, we fix $N = 1774$ and for $Q_{city}(D)$ data set, we fix $N = 288$, in both cases the pairs were randomly selected. We run BlueFinder and PIA on the same datasets and we compare both approaches.

Results

In this section, we detail and compare the results obtained by PIA and by BlueFinder.

For the data set $Q_{birthplace}(D)$, the recall of PIA is 0.518, the recommended path query is `#from/*/Cat:People_from_#from/ #to`. This means that around % 51 of pairs are connected by the`#from/*/Cat:People_from_#from/ #to`. The precision is 0.587 and the $F1$ score was 0.55.

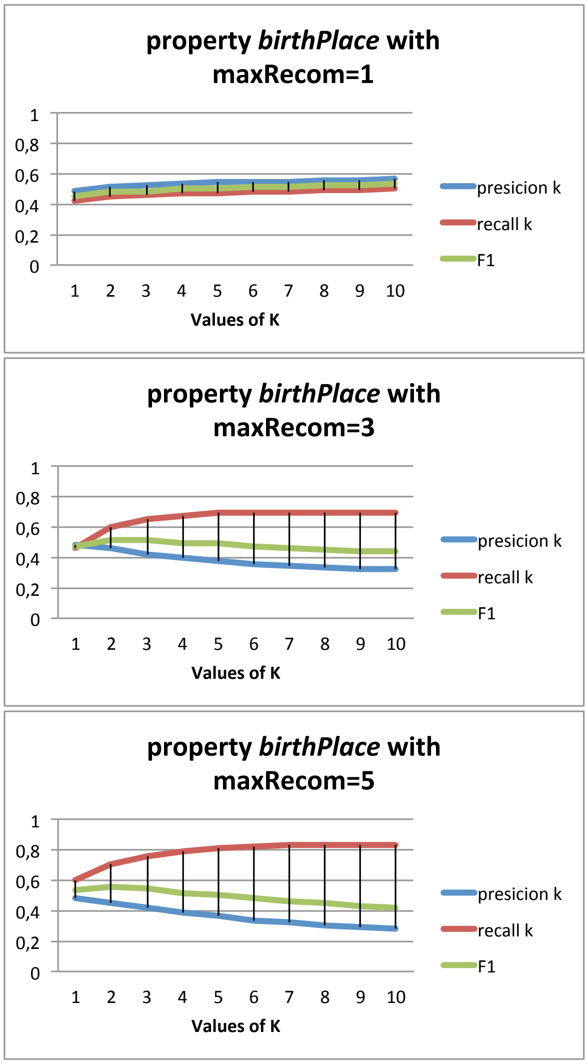

Figure 2. Evaluation for $Q_{birthplace}(D)$ data set using different values of K.

We run BlueFinder with the same data set. The first thing we could observe was that BlueFinder recommend the `#from/*/Cat: Sportspeople_from_Boston/#to`, `#from/*/#from_F.C._players/#to` and `#from/*/#from_F.C._non-playing_staff/#to` star path queries to the case of *Chris Lawler* from *Liverpool*. This case is the example used in the Introduction of this article and the three BlueFinder recommendations were correct for the case.

Table 2 details BlueFinder precision, recall, F1 and pot for the same data set. For instance, the best value of recall of BlueFinder is 0.832 with $k = 8$ and $maxRecom = 5$, the best value of pot is 0.811 with $k = 8, 10$ and $maxRecom = 5$ and the best value of precision is 0.571 $k = 10$ and $maxRecom = 1$.

The worst value of recall is 0.453 with $k = 1$ and $maxRecom = 1$, the worst value of precision is 0.28 with $k = 10$ and $maxRecom = 5$ and the worst value of pot is

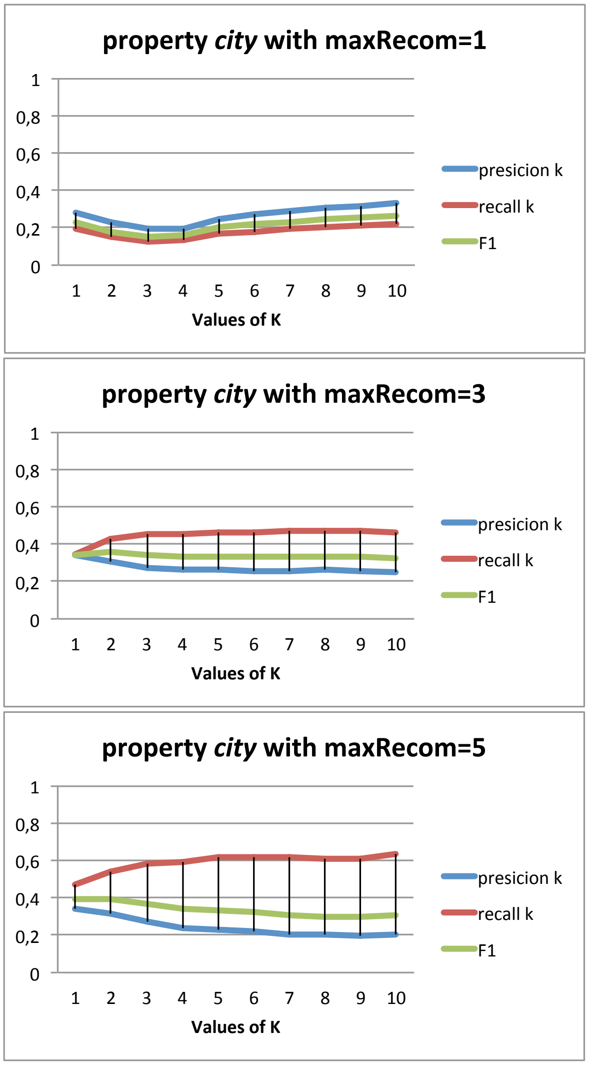

Figure 3. Evaluation for $Q_{city}(D)$ data set using different values of K.

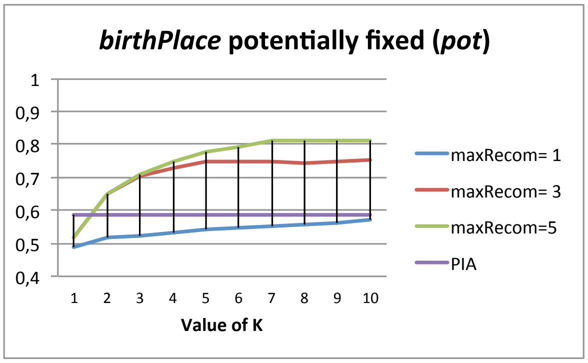

Figure 4. Comparison of pot values with the different configuration of BlueFinder and PIA in $Q_{birthplace}(D)$ data set.

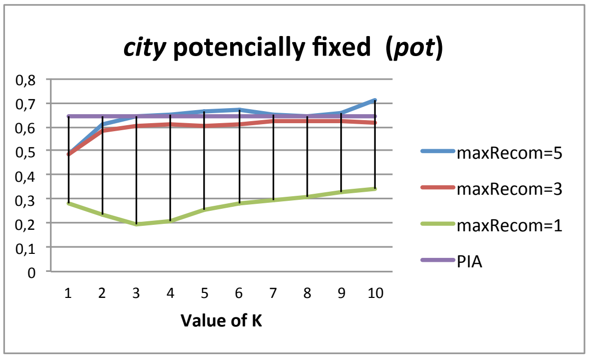

Figure 5. Comparison of pot values with the different configuration of BlueFinder and PIA in $Q_{city}(D)$ data set.

0.488 with $k = 1$ and $maxRecom = 1$.

Figure 2 shows the curve of recall, precision and F1 for $maxRecom = 1; 3; 5$ and different value of K. Figure 4 shows pot function for the previous values. We can observe that for k between 7 and 10, the BlueFinder could fix near 80 % of pairs with $maxRecom = 5$ and nearly 75 % of pairs with $maxRecom = 3$.

In comparison with BlueFinder, the PIA approach showed a rate of potentially under 60 % fixed pairs. Finally, for all k, the BlueFinder with $maxRecom = 1$ is under the potential fixed rate than PIA approach.

For the data set $Q_{city}(D)$, the recall of PIA is 0.418, the recommended path query is `#from/ #to`. This means that around % 41 of pairs are connected by the `#from/ #to`. This path represents a direct link from the city page to the university page. The precision is 0.636 and the $F1$ score was 0.505.

Table 3 details BlueFinder results for the same data set. For instance, the best value of recall of BlueFinder is 0.636 and the best value of pot is 0.708 with $k = 10$ and $maxRecom = 5$ and the best value of precision is 0.344 with $k = 1$ and $maxRecom = 3; 3$.

The worst value of recall is 0.123 with $k = 4$ and $maxRecom = 1$, the worst value of precision is 0.197 with $k = 9$ and $maxRecom = 5$ and the worst value of pot is 0.194 with $k = 3$ and $maxRecom = 1$.

Figure 3 shows the curve of recall, precision and F1 for $maxRecom = 1; 3; 5$ and different value of K. Figure 5 shows pot function for the previous values. For $maxRecom = 5$, the potential fixed rate is nearly 65 % for k between 3 and 9 and then climbs to 70 % for $k = 10$. In the second position appears the PIA approach with a rate of 64 %.

The results show us that BlueFinder can correct more pairs than the PIA approach. By analysing the *pot* values, we can see that for both datasets BlueFinder with $maxRecom = 5$ had a better performance than PIA. By analysing precision, recall and F_1 score, we can see that $maxRecom$ has more impact than the value of k. However, if we compare precision and recall values with pot values, we can see that a bigger value of k increases the number of potentially fixed pairs. This means that a value of k between 7 and 10 and a value of $maxRecom = 5$ ensures for both datasets a high recall and a high potentially fixed rate. However, the increasing number of recommendations decreases the precision value. Finally, when the $maxRecom = 1$ and the value of k is getting bigger, the BlueFinder behaves like PIA approach. This is a natural consequence, because when k is getting close to the total of connected pairs, the set of neighbors will be similar to the complete N set and it gives the most general path query.

In terms of processing time, the algorithm was executed quickly. It took 50 seconds for the $Q_{birthplace}(D)$ dataset and it took 2 seconds for the second dataset. The main bottleneck in the computation is computing the PIA indexes. For that, BlueFinder uses pre-computed PIA indexes.

CONCLUSIONS AND FURTHER WORK

In this article, we have introduced the BlueFinder algorithm which is an improvement of the previous PIA algorithm. PIA algorithm finds a path query that links two articles in Wikipedia related by a property in DBpedia. Although PIA respects Wikipedia conventions it only predicts the most general path query. Instead, BlueFinder fixes this PIA's drawback by paying special attention into the specificity of the resources. BlueFinder algorithm filters the results from PIA to recommend better representations for a DBpedia semantic property in Wikipedia. It follows a collaborative filtering strategy to recommend a the best path query between a given couple of Wikipedia pages according to a given DBpedia semantic property.

BlueFinder is based on the popular k-Nearest Neighbors(kNN) and Multi label kNN algorithm [25] adapted to the context of DBpedia and PIA index. It uses a similarity function called jccD. jjcD is an adaptation of Jaccard distance which measure similarity between Wikipedia pages by using a DBpedia semantic property.

To validate BlueFinder, we have run an evaluation to compare the new approach with the previous one. The analysis of the evaluation showed us that BlueFinder algorithm obtains path queries with more specificity to connect in a right way the Wikipedia articles. This implies that BlueFinder has a better recall than PIA.

Although this good levels of recall are an important help to improve the Wikipedia content, it is necessary to continue working to improve the filter steps in the recommendation algorithm in order to improve the precision values in the recommendations. We plan to analyse other similarity measures like using semantic properties. In other research line, we plan to adapt this approach in combination with other languages version of Wikipedia. Finally, we plan to extend the approach to any property in DBpedia.

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.

This work was also funded by: the PAE 37279-PICT 02203 which is sponsored by the ANPCyT, Argentina.

REFERENCES

1. Wikipedia. The Free Encyclopædia that Anyone Can Edit. *Online http://www.wikipedia.org/* (2006).
2. Abiteboul, S., and Vianu, V. Regular path queries with constraints. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '97, ACM (New York, NY, USA, 1997), 122–133.
3. Adafre, S. F., and de Rijke, M. Discovering missing links in wikipedia. In *Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05, ACM (New York, NY, USA, 2005), 90–97.
4. Adomavicius, G., and Tuzhilin, A. Toward 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.
5. Alkhateeb, F., Baget, J.-F., and 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).
6. Arenas, M., Conca, S., and 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, ACM (New York, NY, USA, 2012), 629–638.
7. Billsus, D., and Pazzani, M. J. Learning collaborative information filters. In *ICML*, J. W. Shavlik, Ed., Morgan Kaufmann (1998), 46–54.
8. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web 7*, 3 (2009), 154 – 165.
9. Breese, J., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc. (1998), 43–52.
10. Chan, K., Chen, T., and Towey, D. Restricted random testing. *Software QualityECSQ 2002* (2006), 321–330.
11. Grčar, M., Fortuna, B., Mladenič, D., and Grobelnik, M. knn versus svm in the collaborative filtering framework. *Data Science and Classification* (2006), 251–260.
12. Hoffmann, R., Amershi, S., Patel, K., Wu, F., Fogarty, J., and 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, ACM (New York, NY, USA, 2009), 1849–1858.
13. Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. *Recommender Systems: An Introduction*, 1 ed. Cambridge University Press, September 2010.
14. Lu, W., Shen, Y., Chen, S., and Ooi, B. Efficient processing of k nearest neighbor joins using mapreduce. *Proceedings of the VLDB Endowment 5*, 10 (2012), 1016–1027.
15. Panchenko, A., Adeykin, S., Romanov, A., and 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.
16. Pérez, J., Arenas, M., and Gutierrez, C. Semantics and complexity of sparql. *ACM Trans. Database Syst. 34*, 3 (September 2009), 16:1–16:45.
17. Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, ACM (New York, NY, USA, 2007), 697–706.
18. Sunercan, O., and Birturk, A. Wikipedia missing link discovery: A comparative study. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, AAAI (2010).
19. Toldo, R., and Fusiello, A. Robust multiple structures estimation with j-linkage. *Computer Vision–ECCV 2008* (2008), 537–547.
20. Torres, D., Molli, P., Skaf-Molli, H., and 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).
21. Torres, D., Molli, P., Skaf-Molli, H., and Díaz, A. Improving wikipedia with dbpedia. In *WWW (Companion Volume)*, A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, Eds., ACM (2012), 1107–1112.
22. Wang, Y., Wang, H., Zhu, H., and Yu, Y. Exploit semantic information for category annotation recommendation in wikipedia. *Natural Language Processing and Information Systems* (2007), 48–60.
23. Zhang, M., and Zhou, Z. A k-nearest neighbor based algorithm for multi-label classification. In *Granular Computing, 2005 IEEE International Conference on*, vol. 2, IEEE (2005), 718–721.