Improving Wikipedia with DBpedia

Diego Torres
LIFIA, Fac. Informatica, UNLP
50 y 115, S/N
1900 La Plata, Argentina
diego.torres@lifia.info.unlp.edu.ar

Pascal Molli
LINA, Nantes University
2, rue de la Houssiniere
44322 Nantes, France
Pascal.Molli@univ-nantes.fr

Hala Skaf-Molli
LINA, Nantes University
2, rue de la Houssiniere
44322 Nantes, France
Hala.Skaf@univ-nantes.fr

Alicia Diaz
LIFIA, Fac. Informatica, UNLP
50 y 115, S/N
1900 La Plata, Argentina
alicia.diaz@lifia.info.unlp.edu.ar

ABSTRACT

DBpedia is the semantic mirror of Wikipedia. DBpedia extracts information from Wikipedia and stores it in a semantic knowledge base. This semantic feature relies in making complex queries inferring relations among articles which sometimes are missing in Wikipedia. This difference generates an information gap between DBpedia and Wikipedia. Could be improved Wikipedia with DBpedia new information to reduce this gap? How this new information should be added to Wikipedia? In this article, we propose a path indexing algorithm (PIA) who takes a data set of a DBPedia query and returns the best representative path to be applied in the Wikipedia. We evaluate the results of applying PIA to express the relation between people and their birth city.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Selection process; E.2 [**Data Storage Representation**]: Linked Data; E.1 [**Data Structures**]: Graphs and networks—*Bipartite Graph*

General Terms

Algorithms, Experimentation

Keywords

Path query, DBpedia, Wikipedia

1. INTRODUCTION

Semantic web is growing fast and bring better search and navigability on the web. It is mainly built from meta-data which is extracted from the social web. A good example is DBpedia [2], a knowledge base extracted from Wikipedia[1] infoboxes and wiki markups. DBpedia supports complex semantic queries such as ”people born in Berlin before 1900”. However, the result of this query can give names that cannot be retrieved by navigating from Berlin to these people in Wikipedia.

DBpedia organizes the information in a semantic knowledge base where concepts are related by properties. This organization makes possible to deduce new relations which are not explicit in Wikipedia. Unfortunately, it is not evident how these deduced relations can be injected in Wikipedia.

If we could know how the semantic relations of DBpedia are expressed in Wikipedia, we can understand which relations are missing in Wikipedia and what are already existing. Knowing how many missing relations gives us a quantitative idea of the improvement that can be introduced in Wikipedia. In the other hand, the existing ones are useful as a source to learn, from Wikipedia community, the convention used to represent relations. In our approach, this learning will be the manner to discover how missing relations should be added in Wikipedia . With this approach, deduced relations from DBpedia could be injected in Wikipedia and complete the existing information gap between Wikipedia and DBpedia.

In order to achieve this, first of all it is needed to understand how Wikipedia represents relationships among articles. The basic relation between one article to another is a *Wikilink* which links one article with other by means an hyperlink. Wikilinks helps to define one-to-one relations. However, one-to many relationships have a special treatment in Wikipedia; they are expressed by *Categories.* Categories are special pages which allow users to group articles behind a category name and them are organized in a tree structure. Mainly, Wikipedia community uses this hierarchical organization as a table of contents to group articles for an easier navigation and for correlating similar information. However, categorization system in Wikipedia has not a semantic purpose.

On the other hand, DBpedia relies on a consistent ontology which allows the deduction of new knowledge. This is the cause of having information in DBpedia that is missing in

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

Wikipedia. Particularly, we have detected relations which are missing in Wikipedia. In order to detect lacking information, we must first obtain from DBpedia all the articles which are connected by the relation i.e. a property like *birthPlace* or *deathPlace* and then, to discover which are existing in Wikipedia and which are missing. For example, in DBpedia, the property *birthPlace* link a *Person* to a *Place*. So it is easy to build a query called *is birthPlace of* that returns all *Person* born in a *Place*.

The case of missing results are the motivation of our work, because of we would like to help users to improve Wikipedia with this new information. But the cases where the relation already exists in Wikipedia are also useful. They helps us to learn how Wikipedia community best represent these relations in terms of navigational paths through wikilinks and categories. This learning must be used to fix the missing relations. However, it may be several paths which represents a relation. Following the example, it may exist many paths that link a particular *Place* to a *Person*. Which of them best represents the missing relations showed by the query *is birthPlace of*? Which of these path are useful to the users? Can we automatically infer paths which best represent Wikipedia community conventions?

In this paper, we developed an algorithm called Path Index Algorithm (PIA) that, from a semantic relation in DBpedia, indexes Wikipedia paths which represent the relation starting from individuals. Paths are generalized by replacing properties of source and target articles with wildcards forming *path queries*. A path query is a general specification for many paths in Wikipedia. For example, starting from Berlin, the link *people_from_Berlin* is normalized to *people_from_[#from]*. The hypothesis is that the smallest path query in the index that maximally contain the semantic relation in DBpedia is the best expression of this semantic relation in Wikipedia. In order to validate this hypothesis, we run the proposed algorithm on DBpedia query. The algorithm has demonstrated to be able to discover recommended conventions of category tree of Wikipedia.

The paper is organised as follow; a Motivating example is introduced in Section 2. The path indexing approach and the PIA algorithm are introduced in Section 3. In Section 4 we developed an evaluation of our proposal. Section 5 describe related approaches. Finally, conclusions and further work are detailed in Section 6.

2. MOTIVATING EXAMPLE

The semantic information of DBpedia [2] is extracted from Wikipedia infoboxes and wiki markups. For example, DBpedia extract from each person in Wikipedia the infobox property *birthPlace* and creates an entry in DBpedia knowledge base which relates the person and the city with a semantic property called *birthPlace*. This allows to perform semantic queries in DBpedia such that:

```
SELECT ?city , ?person WHERE{
?person a Person .
?city a City .
?person birthplace ?city}
```

This query retrieves all the people and their born city that is contained in DBPedia. In the other hand, analysing in Wikipedia the 119097 of pairs $(city, person)$ retrieved from DBpedia by the *is birthPlace of* query, we were surprised that in only 65200 cases it was possible found a path in Wikipedia from the city to the people by means of category tree or direct wikilinks. The result is even more surprising considering that the information used by DBpedia is the one extracted from Wikipedia.

To improve the Wikipedia content with knowledge of DBpedia, we have first to learn how people express the relation *is_birthPlace_of* among cities and people in Wikipedia. In this paper, we propose to learn this analysing those paths which connect a city with a person who was born there.

Articles in Wikipedia are connected by wikilinks and are grouped in categories. A wikilink connects an article with other directly in a one-to one relation. In the other hand, *"The central goal of the category system is to provide links to all Wikipedia articles in a hierarchy of categories which readers can browse, knowing essential, defining characteristics of a topic, and quickly find sets of articles on topics that are defined by those characteristics"*[2]. Therefore, categories are used to define topics (named in singular) or set of elements (named in plural).

Consequently, a one to many relation among articles is expressed by group the relation in a category i.e. France relates its canals by the category `Cat:Canals in France`[3]. This relations are expressed using the category tree and naming conventions. In the example, the category `Cat: Canals in France` is a subcategory of the category `Cat:France` and also the subclassification is reinforced by the word *in* in the category name in order to highlight that the articles belonging to the category are *Canals* which are part of *France*.

Although Categories in Wikipedia are organized in a hierarchy, according to Suchanek et al. [7], "Wikipedia category hierarchy is barely useful for ontological purposes. For example, Zidane is in the category `Football in France`, but Zidane is a football player and not a football". However, most of the names of the categories does not denote the elements them include but names means a family of categories based on the relationships with elements they denote, for example explicit relation categories (e.g. *Villages in Brandenburg*) or class attribute categories (e.g. *Albums by Artist*) [6]. Consequently, as there is not an evident meaning among categories, it must be necessary to found and evaluate the paths within the category tree to discover the relations.

In order to analysing paths we start with an example. In the case of big cities articles, like Paris, most of the related pages to Paris are organized under the category `Cat:Paris`. The same can be observed for other cities like New York, London, Boston, etc.. In these cities, the people who was born there is represented belonging to a subcategory called `Cat: People_from_<city>` where `<city>` is the name of the city. Notice that this category represent a relations one-to-many. For example for Boston, the subcategory is `Cat:`

[2]http://en.wikipedia.org/wiki/Wikipedia:Categorization
[3]In this article we will use the prefix `Cat:` to indicate categories

`People_from_Boston` which is besides subcategory of the category *Cat:Boston*. This is a common practice that could be extracted in a general path like `Cat:<from> / Cat:People_ from_<from>` which represents a path to navigate (written with the symbol `/`) to a category and then continue to its subcategory `Cat:People_from`. This path represents, for example, the case of Boston (`Cat:Boston /Cat:People_from-_Boston`) or the case of Paris (`Cat:Paris/Cat:People_from-_Paris`) or any other city. However, there are many paths connecting Cities with People, for example `Cat:Capitals-_in_Europe / Cat:Paris / Cat:People_from_Paris`. Having several alternatives, Which of this examples is the best representation for the relation birthPlace?

The main concern of this paper is to find out how we can automatically compute the best path that represents a relation in DBpedia. This knowledge can be used to assist Wikipedians when editing pages or can be used by Wikipedia bots to maintain conventions that can be now dynamically computed.

Imagine the following interaction in Wikipedia by Bob, a Wikipedia user who access to the writer Robin Moore article[4] in Wikipedia. Then, Bob reads in the infobox summary that the writer was born in Boston, Massachusetts (Boston). When Bob goes to Boston page he founds out that is impossible to navigate to Robin Moore article through wikilinks or categories. To complete the information of Boston article, Bob could ask to the PIA algorithm for retrieving which is the best way to include in the Boston article that Robin Moore was born there. PIA will answer according to Wikipedia conventions, that the best way is by defining a path from Boston article, to Boston category, and then to People from Boston. It is the last category of the path where Robin Moore should belong to.

In next sections we introduce the path indexing approach in detail and the PIA algorithm to detect the best path representation of a semantic relation from DBpedia.

3. PATH INDEXING APPROACH

In this section we introduce the algorithm to index Wikipedia paths. As we have presented in previous sections, the relation between two articles in Wikipedia is established by navigational path. The idea of paths is introduced in the literature [4] and can be described, in terms of Wikipedia, as a sequence of pages which connects one article to another. To formalize the path concept we formulate the structure of Wikipedia as a directed graph. Hence, Wikipedia can be consider a graph formed by a set of pages (both articles and categories) which are connected by links. Therefore, it is possible to navigate from one page to another by means of links. The set of pages that are crossed form a path. Formally, a path between two pages is defined below as:

Definition 3.1 (PATH). *A path in Wikipedia is a sequence of pages $[v_1, .., v_n]$, s.t. $\forall i, j$ $1 \leqslant i < j \leqslant n$, $v_i \neq v_j$, $\forall 1 \leqslant i \leqslant n-1$, where (v_i, v_{i+1}) is a link between v_i and v_{i+1}.*

[4]http://en.wikipedia.org/wiki/Robin_Moore January 2012

For example, the expression `Boston /Cat:Boston /Cat:People_from_Boston /Robin_Moore` is a path between the articles `Boston` and `Robin_Moore`. / And also, `Turin/ Cat:Turin / People_from_Turin /Carla_Bruni` is a path from `Turin` and `Carla_Bruni`.

In the previous examples, both paths are representing the individual cases for the city Boston and Turin. However, in essence, the general structure of both paths is coincident: they have a sequence of pages where some of the pages include in its name a reference to the origin page. The same occurs for the most of important cities in Wikipedia. therefore, we introduce the *path query* concept in order to represent a set of similar paths. We introduce the definition of a path query in terms of Wikipedia following the path query definition introduced by Abiteboul et al. [1].

Definition 3.2 (PATH QUERY). *A path query is a regular expression to represent paths which uses a finite alphabet (like "*") and is posed relative to a domain page and a target page. The answer of a path query q evaluated on a domain page d is the set of all target pages reachable from d by some path that is compatible with the regular expression in Wikipedia category tree.*

Following the examples, the path query in Wikipedia category tree $Q1(c,p) = c/[Cat:c]\ /\ [Cat:People\_from\_c]/p$ is a valid path query which represents the example paths for `Boston` to `Robin_Moore`; and `Turin` to `Carla_Bruni`. The first path is compatible by evaluating *Q1(Boston, Robin_Moore)* and the second by the evaluation of *Q1(Turin, Carla_Bruni)*.

The semantic of $Q1$ is; the set of all pairs $(domain, target)$ where replacing the c variable with a domain value and p variable with a target value in each occurrence on $Q1$ is a path in Wikipedia.

With this definitions, we are now able to relate path queries in Wikipedia with those DBpedia queries that also retrieve sets of pairs. These particular DBpedia queries are what we called *binary semantic queries*. The query in the section 2 is an example of them.

But the issue here is to discover the path query that better fit a binary semantic query. In order to have the best representation of a DBpedia binary query in Wikipedia, we will detect the path query which best represent the DBpedia query as follow:

Q=birthPlace(c,p):-{?p,<birthPlace>,?c} a valid result set RS_q of evaluating Q could be $RS_Q = \{(Paris, Bob)$, *(Boston, Alan)*, $(NewYork\ , John)\}$. And let RS_{qw}, the result set of a representation of Q in Wikipedia.

Let Q a DBpedia query, RS_q the result set of pairs *(domain, target)* of evaluating Q; for example, for *Q=birthPlace(c,p):-{?p,<birthPlace>,?c}* a valid result set RS_q of evaluating Q could be $RS_Q = \{(Paris, Bob), (Boston, Alan), (NewYork\ , John)\}$. And let RS_{qw}, the result set of a representation of Q in Wikipedia. We suppose that the shortest path query that maximally contains the semantic relation expressed by

Q(d,t)	Path	Path Query
(City1, Bob)	[City1/ Cat:City1/Bob]	[#from/ Cat:#from/#to]
	[City1/ Cat:City1/Cat:Actor_from_City1/ Bob]	[#from/ Cat:#from/Cat:Actor_from_ #from/ #to]
(City2, Alan)	[City2/ Cat:City2/Cat:Actor_from_City2/ Alan]	[#from/ Cat:#from/Cat:Actor_from_#from/ #to]

Table 1: Path Index Algorithm example

the binary semantic query in DBpedia is the best expression of this semantic relation in Wikipedia. We formally define the smallest maximally contain path query (SMCPQ) with length up to a constant $maxL$ for RS_{qw} as:

DEFINITION 3.3. *(SMCPQ) Let a path query $Q' \subseteq RS_{qw}$, where $\forall p \in Q', length(p) \leqslant maxL$. Q' is the SMCPQ if and only if $\nexists$ a path query $Q'' \subseteq RS_{qw}$ such that $Q' \subset Q'' \subseteq Q_w$*

To compute the SMCPQ we developed Path Index Algorithm (PIA). The PIA algorithm defined in Algorithm 1 computes SMCPQ for a given semantic query Q. It takes as input the RS_q result set and $maxL$ value. The *buildIndex* function performs a Deep First Search up to $maxL$ starting from any node contained in the *domain* of RS_q. For each path reaching the *target*, it normalizes the path into a path query and thus builds the path index as a bipartite graph linking path queries to elements of RS_q. The SMCPQ will be the first ranked path query in the index.

Algorithm 1 PIA algorithm

```
Input: $RS_Q$: set of (domain,target), k for maxL: int
 1: $index \leftarrow buildIndex(RS_Q, k)$
 2: return $firstRanked(index)$
```

The index as a bipartite graph is formalized as $index = (PQ, V, E)$ where PQ is a set of path queries, V is a set of pairs $(domain, target)$ and E is a set or pairs (PQ, V). The *pathQueryGen* function (Algorithm 2) transforms a path into a path query. It replaces occurrences of domain and target by specific wildcards. The *pathQueryGen* is in charge of the path normalization.

Algorithm 2 pathQueryGen

```
Input: path: [v_1,...,v_n], domain, target: WP article
Output: A path query
 1: for all $v$ in path do
 2:     StringReplace($v$, domain, target)
 3: end for
 4: return path
```

Applying *pathQueryGen* on `path = [Paris / Cat:Paris, Cat:People_from_Paris/ Pierre Curie]` produces `[#from/ Cat:#from/ Cat:People_from_#from/ #to]`.

To explain the complete functionality of the algorithm, we describe a short test case. Suppose we need to obtain the SMCPQ for the query Q of DBpedia using the Wikipedia Category Tree. The RS_Q includes two cases which are enumerated in the first column of the table 1 ((City1, Bob), (City2, Alan)). With that result set, we run PIA obtaining for each case the path and its corresponding path query which are listed in the second and third column respectively.

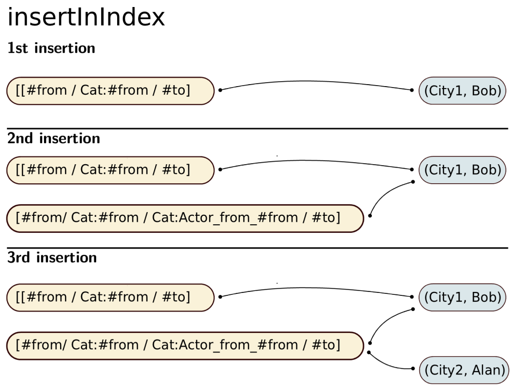
insertInIndex
1st insertion
[[#from / Cat:#from / #to]
(City1, Bob)
2nd insertion
[[#from / Cat:#from / #to]
(City1, Bob)
[#from/ Cat:#from / Cat:Actor_from_#from / #to]
3rd insertion
[[#from / Cat:#from / #to]
(City1, Bob)
[#from/ Cat:#from / Cat:Actor_from_#from / #to]
(City2, Alan)

Figure 1: Insertion example

As it was described above, for each path query, the PIA makes an insertion in the index. Each insertion updates the bipartite graph by adding the new path queries and the correct edge. In our example, we have to insert three path query, two for (City1, Bob) and one for (City2, Alan). In Figure 1 we describe the state of the index after each insertion. The first insertion, insert the node corresponding to the path query `[#from/ Cat:#from/ #to]`, the corresponding case where the path query is valid (in this case (City1,Bob)) and an edge between the path query and - (City1,Bob).

For the second insertion, the algorithm only insert the path query `[#from/Cat:#from/Cat:Actor_from_#from/ #to]` and the edge between this and the pair (City1,Bob) because the last one already exist in the index. Finally, PIA inserts the path query corresponding to the case of (City2,Alan). As the path query is present in the index, the algorithm only insert the pair (City2,Alan) and the edge between the path query `[#from/ Cat:#from/ Cat:Actor_from_#from/ #to]` which was in the index and the new inserted pair.

After the PIA algorithm finishing, the state of the path query index is the last one showed in Figure 1. In terms of the bipartite graph, the most representative path query is the one with the highest degree. In this case, the degree of `[#from/ Cat: #from/Cat:Actor_from_ #from/ #to]` is 2, because it has and edge to (City1, Bob) and other to (City2, Alan), in the other hand the degree of the path query `[#from/ Cat:#from/#to]` is 1, because it has only one edge.

In conclusion, the PIA returns the SMCPQ from of pairs (domain,source) of a DBpedia query Q. In the next section we run an evaluation of the path generation algorithm.

4. EXPERIMENTATION

In this section we will show the result of doing some experimentation with the PIA algorithm. In order to achieve this,

Domain	Range	Cases	Found	Not Found	Errors
City	Person	119097	65200	49899	3998
City	Philosopher	171	103	61	7
France	Philosopher	21	21	0	0

Table 2: Evaluation datasets for "is birthPlace of" using different Domain and Range

Pos	Path Query	#
1	`[#from]/ [Cat:from]/ [Cat:People_from_#from/ [#to]`	34008
2	`[#from]/[#to]`	16312
3	`[#from]/[Cat:Wikipedia_semi-protected_pages]/ [Cat:Wikipedia semi-protected categories]/ [Cat:Living_people]/ [#to]`	3188
4	`[#from]/ [Cat:Articles_containing-_Japanese_language_text]/ [#to]`	2422
5	`[#from]/ [Cat:Cities_in_New_York/ [Cat:#from]/ [Cat:People_from_#from]/ [#to]`	2104
6	`[#from]/ [Cat:Capitals_in_Europe]/ [Cat:#from]/ [People_from_#from]/ [#to]`	2028

Table 3: Generated Index results case 1: is birthPlace of (City, People)

Pos	Path Query	#
1	`[#from] /[Cat:from]/[Cat:People_from_#from/ [#to]`	60
2	`[#from] /[Cat:Capitals_in_Europe]/ [Cat:#from]/[People_from_#from]/[#to]`	15
3	`[#from] /[#to]`	14

Table 4: Index results case 2: is birthPlace of (City, Philosopher)

Pos	Query Path	#
1	`[#from]/ [Cat:#from]/[Cat:French people]/ [Cat:#French people by occupation]/ [French philosophers]/ [#to]`	21
2	`[#from]/ [Cat:#from]/ [Cat:French people]/ [Cat:#French people by occupation]/ [French sociologists]/ [#to]`	3
3	`[#from]/ [Cat:#from]/ [Cat:French people]/ [Cat:#French people by religion]/ [French atheists]/ [#to]`	2

Table 5: Index results case 3: is birthPlace of (France, Philosopher)

we implemented the PIA algorithm and run it considering the following three queries:

```
SELECT ?city , ?person WHERE{
?person a Person.
?city a City.
?person birthplace ?city}
```

```
SELECT ?city , ?philosopher WHERE{
?philosopher a Philosopher.
?city a City.
?philosopher birthplace ?city}
```

```
SELECT France, ?philosopher WHERE{
?philosopher a Philosopher.
?philosopher birthplace France}
```

The queries have been executed on a DBpedia snapshot produced in January 2012. The PIA algorithm has been executed on a copy of the English version of Wikipedia produced in October 2011[5].

Table 2 shows how many pairs are produced by each query on DBpedia (Cases column) and, for each pair *(domain, target)*, evaluates the existence of paths between the domain and the target in Wikipedia: If it exists then, the pair is computed as found path (Found column), if not it is computed as not found path (Not Found column). The *Errors* column shows the cases where elements of DBpedia are not present in Wikipedia dump. This comes mainly from desynchronization between Wikipedia and DBpedia.

In this evaluation, we have also analysed the results obtained by the execution of the algorithm for the datasets. The path index for each dataset is showed in tables 3, 4 and 5. The first column of each table point out the rank position, in the second column appears the path query and in the last column, it is indicated the number of elements of the path query result set. The index is sorted and the first path query is the SMCPQ returned by PIA . Below, we explain the information of the index result tables.

Table 3 shows the path index for the semantic query which relates cities and people. The SMCPQ is the first path query showed in table 3. This SMCPQ describes a path built by a category with the same name of the city and a subcategory of this called `Cat: People_from_#from` where #from is a wildcare which replace city name. For example, for Boston to Bob the path is `[Boston / Cat:Boston / Cat:People_from_Boston / Bob`. Additionally, table 3 includes the administrative category *Wikipedia semi-protected categories* in the third path query. Administrative categories are not reachable by regular Wikipedia user and then this cases must be pruned from the index.

The index path obtained for the second data set showed in table 4 promoted as SMCPQ the same than in the former case between cities and people. The main difference in this index path is that we have pruned administrative categories from the index ranking. Finally, in the last semantic query dataset, which relates the country France with Philosophers,

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

we observe in table 5 that the SMCPQ is representative for all the cases from the dataset.

Concluding the evaluation, we have collected relevant information. First of all, the SMCPQ for the first and second cases was the same even having in account that the second case was more specific than the first one. This path queries show us that according to Wikipedia Community the general and most used path to express the relation birth place among cities and people is by the query path `[#from]/ [Cat:from]/ [Cat:People_from_#from/ [#to]`. Returning to the motivating example at the end of section 2, at this moment we are able to answer Bob's question: the best path to represent the relation between Boston and Robin_Moore is `Boston / Cat:Boston/ Cat:People_from_Boston/ Robin-_Moore` based on 34008 paths which represents the same relation in Wikipedia. Finally, the SMCPQ which best represent the relation of a philosopher who born in France is `[#from/ Cat:#from/ Cat:French_people/ Cat:French_people_by-_occupation/ Cat: French_philosophers/ #to]` . This path query differ from the previous mainly because the domain of the query is other. Instead of cities this query relates the country France with philosophers. The SMCPQ obtained for this case shows the concordance with that in Wikipedia people is usually categorized by their nationality and occupation.

5. RELATED WORK

There are some previous works that studied different approaches to improve the content of Wikipedia by using external resources. Wigipedia is on of them. Wigipedia [3] fallows regular WikipediaÂt's users to introduce semantic relations from DBpedia into Wikipedia. This is done by means of an interface which can be embed in a Wikipedia article page. The Wigipedia approach generates from the current article a graph of relations with no more than 2 hops. The graph is conformed by semantic relations, best ranked outgoing links and related infobox with the visualized article. By inspecting this graph, the user would be able to repair eventual missing relations. However, our approach is more general in the study of any semantic query made in DBpedia and not only those related with a particular article. Besides, Wigipedia does not take in consideration the extraction community convention in a whole manner as our approach. The study introduced in this article also explores what is the best way to add missing relations according to the Wikipedia Category Tree usage while Wigipedia mainly fix by adding direct wikilinks for a specific article.

In the other hand, Hoffman et al. [5] developed human-machine collaborative system to complete infobox information taking sentences from the current Wikipage by using Kylin [9]. Kylin is an information extraction system which for the work in [5] extracts training data analysing Wikipedia infoboxes and collects the best sentences in the article body that represents a field in the infobox. After that, Kylin detect missing infobox properties which appears in the body of other articles. Finally, the system makes suggestions to the user for adding missing fields in the infobox. As in our approach, the machine efforts are used to detect, from Wikipedia information, missing relations which are not evident. By contrast, we are able to fix relations among articles which does not have sentences in their text body, based on DBpedia information and following the community usage in the developing of the Wikipedia category tree.

Less related with our approach is the Madwiki [4] work. Madwiki stores information in structured databases using slots pairing attribute/values. Views of this databases are generated as wikipages where users can add and manage information. When changes are generated, Madwiki take the control an synchronize users information with the structured database fixing inconsistencies. Finally, Madwiki uses the idea of path view language to navigate from a node to another in the database representation. We were inspired in the idea of Madwiki navigational path to define the navigational path in Wikipedia, however the main difference with our approach falls in the absence of a semantic base like DBpedia.

Related with the algorithmic field there are several works in link and patterns predictions. For example, Thor et al. in [8] developed a link prediction framework for a controlled vocabulary ontology using tripartite and bipartite graph approach for biological datasets. In their work, they developed a prediction link framework for an annotated graph using topological shortest path. They used graph summarization to transform dense graph into simpler to analyze graphs. However, the idea of generates shortest path in a topological order in a graph is similar, but the main approach of their work is far from our.

6. CONCLUSIONS AND FURTHER WORK

In this article we introduced a novel approach to detect the best expression in Wikipedia of a DBpedia Query. For that, we have developed an algorithm that index path queries to $maxL$ length called Path Index Algorithm . In the article we also analyse a preliminary evaluation of our approach that confirms our hypothesis.

During the analysis of the evaluation, we have noticed that the path query are not always coincident with Wikipedia categorization usage. In some cases because the analysed DBpedia query has not a directly categorization rule which involves it, for example *birthplace* between Cities and People instead of Countries and People with Occupation. A combination with the obtained path query and the confidence level bring us important information to determine de evolution in Wikipedia Community of that path query. However, this discrepancies trigger different challenge to work on them. We are working on the improvement and analysis of new normalization strategies involving the use of semantic properties and Wikipedia relations in the path query building, including the early detection of administrative categories.

7. 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.

8. REFERENCES

[1] S. Abiteboul and V. Vianu. Regular path queries with constraints. In *Proceedings of the sixteenth ACM*

SIGACT-SIGMOD-SIGART symposium on Principles of database systems, PODS '97, pages 122–133, New York, NY, USA, 1997. ACM.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165, 2009.

[3] S. Bostandjiev, J. O'Donovan, C. Hall, B. Gretarsson, and T. Höllerer. Wigipedia: A tool for improving structured data in wikipedia. In *ICSC*, pages 328–335. IEEE, 2011.

[4] P. DeRose, X. Chai, B. J. Gao, W. Shen, A. Doan, P. Bohannon, and X. Zhu. Building community wikipedias: A machine-human partnership approach. In G. Alonso, J. A. Blakeley, and A. L. P. Chen, editors, *ICDE*, pages 646–655. IEEE, 2008.

[5] R. Hoffmann, S. Amershi, K. Patel, F. Wu, J. Fogarty, and D. S. Weld. Amplifying community content creation with mixed initiative information extraction. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1849–1858, New York, NY, USA, 2009. ACM.

[6] V. Nastase and M. Strube. Decoding wikipedia categories for knowledge acquisition. In D. Fox and C. P. Gomes, editors, *AAAI*, pages 1219–1224. AAAI Press, 2008.

[7] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[8] A. Thor, P. Anderson, L. Raschid, S. Navlakha, B. Saha, S. Khuller, and X.-N. Zhang. Link prediction for annotation graphs using graph summarization. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy, and E. Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 714–729. Springer, 2011.

[9] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, and A. O. Falcão, editors, *CIKM*, pages 41–50. ACM, 2007.