Automatic City Knowledge Discovery from Internet Resources

Nazanin Firoozeh

Laboratoire d'Informatique de Paris-Nord, Université Paris 13
Pixalione SAS
Paris, France
Email: nazanin.firoozeh@lipn.univ-paris13.fr

Abstract—**Knowledge Discovery plays an important role in the Artificial Intelligence field. Due to the growing nature of the Web, using proper sources of extraction and reducing human intervention is an important step towards creating rich knowledge bases. Urban simulations are one type of Interactive Virtual Environment, which attempt to represent dynamic processes and interactions of urban development. Making these virtual environments closer to the human behaviour requires rich sources of knowledge. This paper presents a pattern-based approach for knowledge extraction. The goal is to extract the implicit knowledge behind any given city-related text. To achieve this goal, we make use of category names and infobox tables from Wikipedia. The system takes two inputs: 1. a text/Uniform Resource Locator (URL), 2. set of extraction patterns. Comparing to some of the proposed tools in the state-of-the-art, our system uses a simpler approach which reduces the human intervention. We tested the system with different text inputs and represented the results as both a text file and a set of triples. Manual evaluation of the system showed its good performance. According to the results, category names are a good resource of common sense knowledge when compared to infobox tables, which mostly contain basic knowledge.**

Keywords–Urban Simulation; Knowledge Discovery; Common Sense Knowledge

I. INTRODUCTION

Web pages contain a great amount of knowledge which is daily used by people or different systems. This knowledge plays an important role in Artificial Intelligence (AI) and Natural Language Processing (NLP). In fact, availability of large coverage and machine readable knowledge is an important step towards the goals of intelligent systems such as urban simulation systems. Urban simulation models are increasingly being used in city, country, and regional planning agencies to assess alternative transportation investments, land use regulations, and environmental protection policies. To have better simulations and also make the city agents more intelligent, using a rich knowledge base is essential.

One advantage of using large and high quality knowledge bases is to help agents to make better decisions and perform well, especially in real-time environments. In addition, having a dynamic knowledge extraction system, agents will be able to dynamically extract knowledge while facing different concepts of the real world. Moreover, as common sense knowledge (CS) affects human decision making process, providing a good resource of common sense knowledge is highly important. Having this type of knowledge, agents and humans will make closer decisions.

Wikipedia is a resource which is updated regularly and contains many statements in natural language. Due to the advantages of this resource, such as representing mostly unique information, it is considered as one of the good resources in knowledge extraction. In this work, we make use of Wikipedia and develop an automatic city knowledge discovery system. The goal of the system is to extract basic and common sense knowledge implicitly expressed in the content of any city-related text. To achieve this goal, semi-structured content of Wikipedia is used. Comparing to unstructured content, the advantage of using semi-structured content is to extract more reliable knowledge with higher accuracy rate.

The rest of the paper is organized as follows. In Section II, we briefly describe the works in the related domain. We then describe our proposed model and the implementation steps in Sections III. Evaluation of the system is shown in Section IV. We finally discuss the proposed model and conclude our work in Section V.

II. BACKGROUND

Extracting knowledge from Wikipedia has been one of the interesting domains of research. Some researchers believe that using Wikipedia as an extraction resource improves quality and size of knowledge bases. So far, different systems have been developed for extracting knowledge from Wikipedia. One instance of such systems is YAGO [1], an ontology which uses Wikipedia. In this project, instead of using information extraction methods, categories of Wikipedia articles are used as sources of knowledge. This system makes use of both Wikipedia articles and WordNet database in order to extract facts. The motivation behind combining these two resources is to take advantage of large number of individuals in Wikipedia as well as a clean taxonomy of concepts in WordNet. YAGO is able to detect both *is-a* and *not-is-a* relations such as *BornInYear*, *PoliticianOf*, etc. However, one limitation of YAGO is that it extracts only 14 types of relations and some relations cannot be detected using this system. As the second limitation, it is not able to extract facts from tables, such as infobox tables. In another work, authors extended the system to YAGO2 [2]. Comparing to YAGO, YAGO2 is able to represent the facts along dimensions of time and space. To achieve this, GeoNames resource is also used in addition to the existing resources, i.e., Wikipedia and WordNet.

In another work, WordNet database is enriched using new relations extracted from Wikipedia [3]. The work consists of four steps, which are all automatic. 1) As a pre-processing step, each Wikipedia entity is assigned to the corresponding WordNet synset. 2) First, the system looks for words which are connected to the entity through hyperlinks. WordNet is then used to see if there is any relation between the entity and any of the found words. In case of having a relation, the context is analysed and a pattern is extracted for that relation. 3) In the next step, similar patterns are generalized. 4) Finally, the patterns are applied for finding new relations, which do not exist in WordNet database. Evaluation of this system shows that the precision rate of the extracted relations is not good enough, 0.61 to 0.69. Hence, one disadvantage of this system is that it produces some unreliable facts. In addition, to extract relations between different concepts, the system needs to go through definition of each entity in Wikipedia to look for hyperlinks. This increases the time complexity of the system.

Category names of Wikipedia articles have been also noted in other works. Large scale taxonomy was built from Wikipedia [4]. In this work, semantic relations between categories are found using a connectivity network and lexico-syntactic matching. This method is able to extract both *is-a* and *not-is-a* facts including relations such as of, with, contain, etc. However, although this method extracts relation between category names, it cannot find relation between a specific concept and its corresponding category names.

DBpedia [5], [6], [7] is another knowledge extraction system, which is now available on the World Wide Web. The goal of DBpedia is first to extract structured information from Wikipedia and then to allow users to ask queries against it. It also links different data sets on the Web to Wikipedia data. In the extraction process, authors make use of MediaWiki, which is a wiki software behind Wikipedia. MediaWiki enables authors to represent structured information in an “attribute-value” notation.

One drawback of the initial version of DBpedia was that its data could be based on several months old data. This problem however was solved using DBpedia-live [8]. DBpedia-live provides a live synchronization method based on the update stream of Wikipedia.

Wikipedia articles have been also used for extracting relations between different concepts of Wikipedia [9]. In this approach, an unsupervised approach is used along with linguistic analysis with web frequency information. The goal of using this analysis is to improve unsupervised classification performance. Unlike our approach, which focuses on semi-structured content of Wikipedia, here, unstructured content of articles is taken into consideration.

Similar to the described systems, we also make use of Wikipedia articles as a rich source of information. However, unlike YAGO, YAGO2 and the work done by Ruiz-Casado et al, the system does not use WordNet database. Hence, the complexity of the system is reduced. In addition, while the work done by Ponzetto et al, tries to find types of relations between different category names, our system finds relations between any given concept and its related category names. As a result, it provides various information about a specific concept rather than generating a network of category names.

DBpedia can be considered as the closest knowledge base to our proposed one. Comparing to DBpedia, our proposed approach is a simpler and less structured approach. This work is however an initial effort on knowledge extraction in order to propose a simple but efficient approach for extracting knowledge from Wikipedia. As will be seen later, as a future work, we are going to compare our extracted knowledge with DBpedia knowledge. Comparing the results, we will be then able to see if the proposed method can be considered as a complementary tool for DBpedia. In this case, using our system, some new relations can be added to DBpedia knowledge base. Specifically, in case that a concept does not have a corresponding entry in DBpedia, our system will be able to extract its basic and common sense knowledge in real time. It should be noted that although many works make use of knowledge extracted by DBpedia [10], [11], [12], according to [13] there is still a room for adding more entities and knowledge to this knowledge base.

III. A MODEL OF EXTRACTION

In this section, we explain in detail the different steps of our proposed approach.

A. *Methodology*

Extracting good amount of basic and common sense knowledge and using it in interactive applications is an important step in agents decision making. In particular, providing a comprehensive source of common sense knowledge enables machines to reason about everyday life. Meanwhile, developing automatic systems is highly important, as due to the growing nature of the information on the Web, it is almost impossible to manually create rich and up-to-date knowledge bases.

In this work, we make use of Wikipedia as a source of knowledge. Although Wikipedia has both unstructured and semi-structured content, we focus only on semi-structured content. The main motivation behind this choice is that common sense knowledge is a kind of knowledge that is merely expressed in unstructured content. However, as it is seen in the next sections, having extraction patterns, we will be able to extract this kind of knowledge from semi-structured content.

The two semi-structured sources used in our work are category names and infobox tables. Statistics show that from 2,390,513 available articles in Wikipedia in 2008, 1,057,563 articles (44.2%) contain infobox tables, while 1,927,525 articles (80.6%) have category names [14].

B. *Model*

In this section, the proposed algorithm is described. In general, our model consists of six main steps: 1. getting the Hyper Text Markup Language (HTML) source code, 2. getting the plain text of the source code, 3. parsing the plain text, 4. extracting category-based facts, 5. extracting infobox-based facts, 6. mapping the result into both a text file and a triple format. It should be noted that by category-based and infobox-based facts we respectively mean facts that are extracted from category names and infobox tables. Figure 1 illustrates the steps of the algorithm. Following is the detailed description of each step.

1) Getting the HTML Source Code: The input of the system can be either a city-related plain text or Uniform Resource Locator (URL). In case of having a plain text as an input, the system starts from the third step, i.e., parsing. Otherwise, the HTML source code of the given URL is retrieved for further processing.

```
Data: Text or URL of a web page, extraction patterns
Result: Extracted basic and common sense facts

initialization;
if input is a plain text then
    go to the next step;
else
    get plain text of the web page;
end
parse the text and generate concepts (based-on
  Wikipedia concepts);
while concept exists for each text do
    extract all category names from the category
      section of the corresponding Wikipedia article;
    while category name exists do
        apply extraction patterns and find the
          corresponding facts;
        if category-based facts != null then
            store the result into a text file;
            store the result as a triple
              format;
        else
            continue with the next
              category value;
        end
    end
    get the Wikitext of the concept;
    if Wikitext contains infobox table then
        apply the patterns on infobox table
          and extract the corresponding facts;
    else
        continue with the next concept;
    end
    if infobox-based facts != null then
        store the result into a text file;
        store the result as a triple format;
    else
        continue with the next concept;
    end
end
```

Figure 1. Algorithm of the developed knowledge extraction system.

2) Getting the Plain Text of the Source Code: Not all tags in the HTML code contain informative content. Hence, we apply a filtering procedure on the code to reduce its noisy content. Headers, Footers, Style, and Script tags in HTML code are examples of such noisy tags. Having the clean HTML source code, we then extract its content.

3) Parsing: The next step of the algorithm is to detect concepts of the studied text for which we want to extract facts. This is done through Parsing step. In this step, we customize the Stanford Part-Of-Speech (POS) tagger. The tagger uses Penn Treebank tag set for representing tags [15]. It makes use of different sets of models as training set of the tagging procedure. The model we use in our system is english-left3words-distsim.tagger, which is the widely used one in applications. Accuracy rate of the model is also 96.97%.

Concepts are generated by means of both POS tagger and Wikipedia articles. In fact, we assume that each Wikipedia article corresponds to one concept in the real world. Some concepts however have different representations. For these concepts, Wikipedia redirects users to the same article and shows the same page for different representations. Using the redirection feature of Wikipedia, we avoid generating concepts with the same meaning but different formulations. As an example, both UK and United Kingdom are redirected to the same URL in Wikipedia. Hence, only one of them is taken into consideration for further processing.

4) Extracting Category-Based Facts: Most of the Wikipedia articles have a category section where navigational links to other Wikipedia pages are provided. Using categories of Wikipedia, users are able to quickly find sets of pages related to any Wikipedia article. In order to reveal the semantics encoded in category names of Wikipedia articles, we develop an extraction algorithm which consists of three steps. The goal is to extract a set of triples as {*concept1, relation, concept2*}, *where concept1 is a detected concept in the given text, concept2 is a concept found in category section of Wikipedia, and relation shows the semantic relation between the two concepts.* Following is the detailed description of each step:

a) Extracting category names from HTML source code: In the first step, for any detected concept, source code of the corresponding Wikipedia article is extracted. Category section of the article is then retrieved and its category names are extracted.

b) Discarding uninformative names: Not all the extracted category names are informative. There are some general category names such as “Disambiguation pages” that appear in some of Wikipedia articles. We ignore all the categories under Wikipedia administration.

c) Extracting the facts: In order to extract the knowledge behind category names, we propose 23 extraction patterns based on structures of different category names. To generate the patterns, different criteria such as type and position of the prepositions as well as occurrences of some keywords like Type, Establishment, Disestablishment, etc. are taken into consideration. Table I shows the proposed extraction patterns. As it is seen, one or more facts along with their corresponding triples are assigned to each pattern. It should be noted that the triples represent the relation between only two concepts. Hence, in case of having more than two concepts in the extraction pattern, no triple is assigned.

For each category name, the system checks if it matches any of the patterns. If so, the associated fact is extracted. It is important to mention that in Table I, *X* refers to *concept1*, and both *Y* and *Z* refer to *concept2*. In addition, *YEAR* simply indicates a year and *Xs* shows the plural form of *X*. *X1*, *X2* and *Y1*, *Y2* also show respectively the sub-terms of *X* and *Y*. All these symbols are considered as concepts in our extraction patterns.

5) Extracting Infobox-Based Facts: Infobox is another resource that we use for the purpose of fact extraction. The triples extracted from infobox tables are in the form of {*concept, attribute, value*} which in fact represent the value of a specific attribute for the studied concept. For each detected concept in the parsing step, we do the following steps to extract infobox-based facts:

TABLE I. EXTRACTION PATTERNS AND THEIR ASSOCIATED FACTS AND TRIPLES FOR CATEGORY VALUES OF WIKIPEDIA.

	Extraction pattern	*Fact*	*Triple*
1	Y of X	Y is an attribute of X	{Y, is_attribute, X}
2	Type(s) of Y	X has a type of Y	{X, has_type, Y}
3	Y in X	X has Y	{X, has, Y}

4	Y in Z (Y contains just letters)	X is a Y in Z
		X is a Y
		X is in Z
			
			{X, is_a, Y}
			{X, is_in, Z}

5	Y of Z	X is a Y of Z (Singular Y) X is one of the Y of Z (Plural Y)	
6	Y in YEAR	In year YEAR, there was a Y of X	
7	YEAR introductions	X was introduced in YEAR	{X, was_introduced, YEAR}
8	X in YEAR	X was in YEAR	{X, happened, YEAR}

9	Y established in YEAR	X has been established in YEAR
		X is a Y (Y singular)
		X is one of the Y (Y plural)
			{X, was_established, YEAR}
			{X, is_a, Y}

10	YEAR establishment(s)	X has been established in YEAR	{X, was_established, YEAR}

11	Y establishment(s) in Z	X has been established in Y (if Y contains digit)
		X is in Z
			{X, was_established, Y}
			{X, is_in, Z}

12	Y disestablished in YEAR	X has been disestablished in YEAR
		X was a Y (Y singular)
		X was one of the Y (Y plural)
			{X, was_disestablished, YEAR}
			{X, was_a, Y}

13	YEAR disestablishment(s)	X has been disestablished in YEAR	{X, was_disestablished, YEAR}

14	Y disestablishment(s) in Z	X has been disestablished in Y (if Y contains digit)
		X was in Z
			{X, was_disestablished, Y}
			{X, was_in, Z}

15	YX (one concept)	YX is one form of X	{X, has_form, YX}
16	Y=(Y1 Y2) & X=(X1 X2) (if Y2 = X2)	X is a Y (Y is singular) X is one of the Y (Y is plural)	{X, is_a, Y}
17	Y by type	X has a type of Y	{X, has_type, Y}
18	Y invention	X is a Y invention	{X, was_invented, Y}
19	Y format(s)	Format of X is Y	{X, has_format, Y}
20	YEAR birth	X was born in YEAR	{X, was_born, YEAR}
21	YEAR death	X died in YEAR	{X, died, YEAR}
22	Xs	X is a subset of Xs	{X, is_subset, Xs}
23	If none of the above relations	X R Y (relates)	{X, relates, Y}

a) Getting the Wikitext: Wikitext is a markup language used for writing the content of wiki websites. This language is in fact a simplified alternative to HTML. As the first step of extracting infobox-based facts, we get this raw data of each Wikipedia article.

b) Finding the infobox template: As not all the Wikipedia articles contain an infobox table, for each detected concept, we should check for the existence of the infobox template. The template starts with "{{Infobox" and ends with corresponding "}}". This section is called infobox template and is used for further processing. If Wikitext of a concept does not contain this template, we stop extracting infobox-based facts for that concept.

c) Extracting attributes and values: In case of having an infobox template in the Wikipedia page, we extract the concept's attributes along with their corresponding values. However, not all content of an infobox template produces interesting facts. Hence, we first discard useless attributes including image, caption, logo, alt, coat, footnote, etc. We then define different patterns as regular expressions for extracting attribute names and values. It is important to mention that we just keep the attributes which have at least one informative value.

d) Refining the extracted facts: In some cases, in infobox template, one attribute is related to the previous one. In our work, we try to relate the dependent attributes. As an example of such an attribute we can refer to "date". In some cases, this attribute by itself represents no meaningful fact and instead shows the corresponding date of the previous attribute.

6) Writing the Facts as Text and Triple Formats: The triple format has a format of {*concept1, relation, concept2* }, which represents the relation between any two concepts. As the last step of our work, in addition of representing the results in a human-readable format, i.e., a text file, we write the facts in the triple format. As a future work, these triples can be converted into the Resource Description Framework (RDF) triples in order to make them machine-readable and applicable for further use in real systems.

IV. EVALUATION

Accuracy and reliability of the extracted knowledge is one of the main steps in knowledge discovery. Using incorrect knowledge in different applications decreases their performance. In this work, we evaluate the system by defining three labels that indicate the quality of the extracted facts. The labels are correct, incorrect and ambiguous. A correct fact is a fact with a correct meaning and a proper formulation. Incorrect fact, on the other hand, refers to the fact with an incorrect meaning. Among the extracted facts, some have correct meanings but wrong formulations. For now, these facts are labeled as ambiguous. By differently labeling these facts, we aim to differentiate them from the incorrect ones, since we believe that as a very first step of the future work, we can refine the system to get correct formulations for these cases. Hence in this work, we exclude ambiguous facts from the correct ones in the evaluation step.

In this work, we did a shallow evaluation in order to see the initial performance of the system. This evaluation was done manually by scanning all the facts and finding the ratio of the correct, incorrect and ambiguous facts. To do this, 10 students were asked to label the extracted facts and the final label was assigned based on the majority vote.

Evaluation of the system is an important step as it helps us with further improvements. In this section, performance of each extraction pattern, used for extracting category-based facts, is also evaluated. The following subsection is the description of the experiment step. After, we present the results and analyse them to show the efficiency of our system.

A. Experiment

We tested the system with different city-related web pages. As a shallow and initial evaluation, we took into account the facts extracted from 10 input texts. These texts are either from news websites or city-related web pages. The extracted facts were evaluated in terms of both meaning and formulation.

It should be noted that although it is possible to run the system for any other domain, we evaluated its performance over the city-related web pages, as the global aim of our work is to apply the developed system into interactive city applications. In the following subsection, the obtained results are presented and analysed in order to show the initial performance of the system.

B. Result and Analysis

As mentioned in the previous sections, extracted facts are stored in a text file. Going through the extracted facts, we calculated the ratio of the correct, incorrect and ambiguous facts. Tables II and III show examples of the extracted facts for different labels.

Although most of the extracted facts are considered as basic knowledge, some others can be considered as common sense knowledge. As an example, “Eiffel tower is in Paris” is considered as common sense knowledge since almost all people know it. This means that while saying “I am travelling to visit the Eiffel Tower”, we are implicitly saying that “I am travelling to Paris”.

In case of having category-based facts, the extracted facts are in two types; “R-specific” that explicitly specifies types of relations and “R-generic” that just indicates that the concepts are related without explicitly showing type of the relation. In Table I, patterns 1 to 22 generate R-specific facts, whereas R-generic facts are extracted using pattern 23.

Figure 2 compares the average rates of accuracy, error and ambiguity for both category-based and infobox-based facts and over all the evaluated examples. For the former case, the evaluation metrics are calculated over R-specific facts, since this type shows the performance of the extraction patterns. Hence, in this step, by “total number of the facts” we mean total number of the R-specific facts. In calculations, accuracy, error, and ambiguity rates are obtained by respectively dividing the number of correct, incorrect, and ambiguous facts to the total number of the facts. Result of the evaluation can be also shown as a precision metric. In our evaluation, in order to calculate the precision of the system, we discard ambiguous facts by assuming that they equally affect the positive and negative examples. Having this assumption, the obtained precision values for category-based facts and infobox-based facts are respectively 90% and 91%. It should be noted that these values are related to both basic and common sense

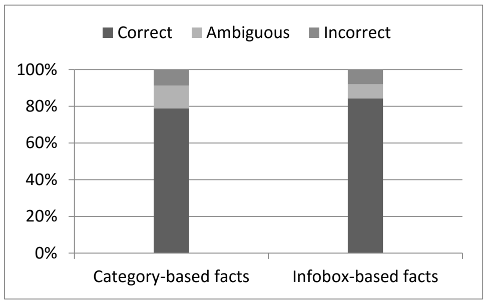

Figure 2. Comparing the average rates of correct, incorrect and ambiguous facts (common sense and basic) over 10 evaluated input texts.

facts. According to the results, the system has an acceptable precision. As an important step in the future work, we are going to use a gold standard for calculating the recall value. In our case, the gold standard may contain a set of facts extracted manually from infobox tables and category names of corresponding Wikipedia articles.

As mentioned before, in the infobox template, in order to extract value(s) of each attribute, patterns are defined using regular expressions. Due to the complexity of the infobox templates, complex patterns should be applied and this increases the complexity of the system. To overcome this issue, we tried to have a trade-off between accuracy and complexity, meaning that instead of extracting all the values, just majority of them are extracted to avoid increasing the complexity. One case that we are not able to capture in our system is the date format containing date, month and year, e.g., 13/09/1988. Although this is one of the limitations of our system, as in some cases dates can be captured using category names, we miss less information. According to our evaluations, half of the incorrect facts are related to the date format.

The next point is that in a very few cases, the extracted category-based facts are almost the same. In fact, these facts are extracted from different category names, which are close to each other. In our system, we tried to reduce the number of similar facts. As a result, in the evaluated examples, we had either no repetitive facts or just one or two cases. For instance, considering the facts “Paris is a capital in Europe” and “Paris is one of the capitals of Europe”, we represent only one of them in the final result.

According to Figure 2, the average rate of accuracy for the extracted infobox-based facts is higher than the one for the category-based facts. However, difference of their error rates is minor. This indicates that most of the infobox-based facts are labeled as correct and incorrect, whereas a higher number of the category-based facts have a degree of correctness and cannot be labeled explicitly as correct or incorrect.

Average rates of R-specific and R-generic category-based facts are shown in Figure 3. As expected, more facts are categorized as R-generic. The reason is that many of the category names have no specific structure. Retrieving facts from these category names though might contain correct facts, increases the error rate. Hence, ignoring such names is more efficient. Figure 4 also compares the amount of common sense and basic knowledge in both category-based and infobox-based facts.

TABLE II. EXAMPLES OF THE EXTRACTED FACTS HAVING CORRECT, INCORRECT AND AMBIGUOUS LABELS - CATEGORY-BASED FACTS.

	Extracted category-based fact	*Evaluated as*	*Basic/CS knowledge*
1	Paris is a capital in Europe	Correct	CS
2	Versailles is an art museum and gallery	Ambiguous	–
3	Eiffel Tower is a Michelin Guide	Incorrect	–

TABLE III. EXAMPLES OF THE EXTRACTED FACTS HAVING CORRECT, INCORRECT AND AMBIGUOUS LABELS - INFOBOX-BASED FACTS.

	Extracted infobox-based fact	*Evaluated as*	*Basic/CS knowledge*
1	Latitude of Paris is 48.8567	Correct	Basic
2	Roof of Eiffel Tower is, abbr=on	Incorrect	–
3	Gini year of Spain is 2005	Ambiguous	–

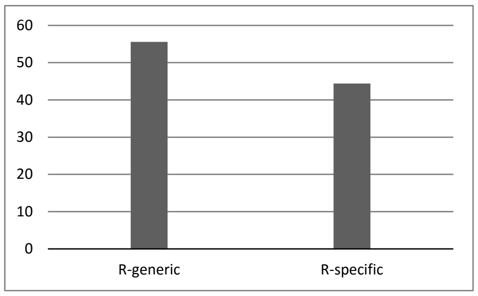

Figure 3. Comparing R-generic and R-Specific rates in category-based facts.

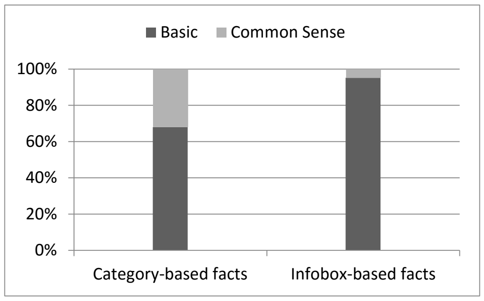

Figure 4. Comparing types of the extracted facts (Common sense vs. Basic).

According to the figure, it can be concluded that infobox template is not a good resource for extracting common sense knowledge. Instead, basic knowledge can be extracted using this table. This is due to the fact that Wikipedia has infobox tables for proper nouns such as Paris and not for general concepts such as shopping. As a result, the extracted facts in most cases cannot be considered as common sense knowledge. On the other hand, category names seem to be a better resource for extracting common sense knowledge. Results show that using category names, we are able to extract both common sense and basic knowledge.

One of the important steps in evaluating the system is to evaluate the performance of the proposed extraction patterns. In this step, pattern 23 from Table 1 is not considered, as it generates R-generic relations, while we are interested to see how efficient the extraction patterns are in extracting R-specific relations. To do this, we focused on half of the input texts (5 texts) and studied the total number of the facts extracted using each pattern (Figure 5). Performance of each pattern is also shown in Figure 6.

According to the figure, patterns 3, 10, 17, 19, 20, and 21 from Table 1 have the highest accuracy rate. However, these patterns extract a few numbers of facts. Considering the patterns with high rates of extraction, i.e., patterns 4, 5 and 22, it can be seen that pattern 22 outperforms the other two patterns due to the high rate of accuracy (96.06%). On the other hand, the result shows that pattern 15 has a poor performance when applied on the input texts, as it mostly generates incorrect facts. Hence, this pattern should be removed while improving the system.

Figure 5 also shows that patterns 11, 12, 13, and 14 extract no fact in the mentioned examples. In fact, they extract some facts but as the produced facts were similar to the previously extracted facts by the other patterns, we removed them from the final result. However, it is important to keep these patterns, as due to their structure, in some cases it might be possible to extract new facts from these patterns.

V. DISCUSSION & CONCLUSION

This paper addresses the problem of automatically extracting basic and common sense knowledge with the goal of providing rich city knowledge bases. These knowledge bases can be then used in interactive city applications to help agents to make decisions. In this work, category names and infobox tables of Wikipedia articles are used as resources of knowledge extraction. Unlike many of the systems in the state-of-the-art, our proposed model is a simple approach which reduces the human intervention. Our system just makes use of the proposed extraction patterns without having the effort of using some thesauri or ontologies. In addition, it enables agents to dynamically extract knowledge when they receive a new input text. Knowledge extracted using this approach could be considered as complementary knowledge of DBpedia.

We generated 23 extraction patterns for extracting facts from category names. Also, some complex patterns were defined to extract attribute and value pairs from infobox tables. Results of the system on 10 input texts show the average rates of correct, incorrect and ambiguous facts as 78.80%, 8.69% and 12.49% for category-based facts and as 84.36%, 8.002% and 7.62% for infobox-based facts. In terms of precision, for category-based facts and infobox-based facts values of 90%

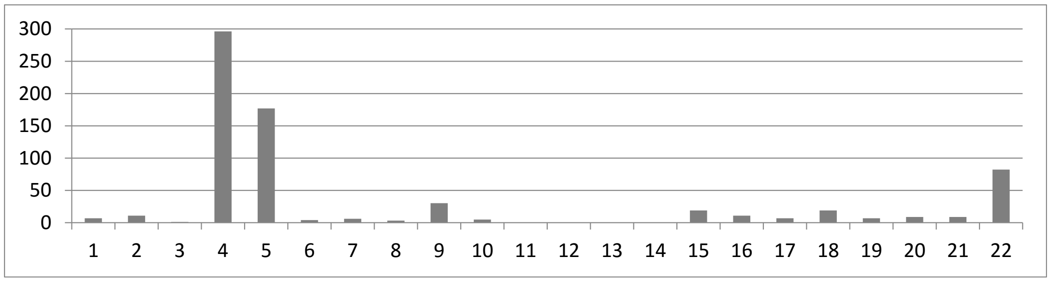

Figure 5. Total number of the facts extracted by each pattern over five input texts.

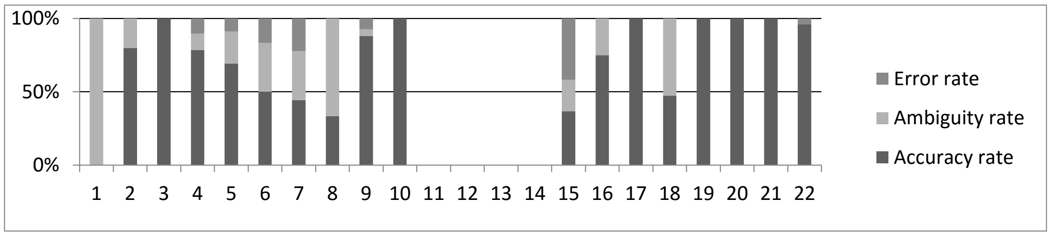

Figure 6. Performance of each extraction pattern over five input texts.

and 91% were respectively obtained. Results also show that category names are better resources for extracting common sense knowledge, while infobox tables mostly extract basic facts. According to the results, among all the category-based extraction patterns, pattern 22 has a better performance with a high accuracy rate of 96.06%.

The first step of the future work is to refine the system in order to get correct formulations for ambiguous facts. As other steps, we can make the system more automated using bootstrapping approach [16], which makes use of a training set, including pairs from infobox tables and the extracted facts, and does a recursive self-improvement. After having an automatic evaluation phase, the next step is to compare our results with the ones obtained from the previous works. To have a better evaluation, value of recall should be also calculated. Gold standard can be then generated manually. As an alternative approach, one could use the facts extracted using DBpedia. As one of the objectives of our system is to extract facts which do not exist in DBpedia, we cannot use this knowledge base for calculating the recall value.

The next step in future work is to convert the extracted triples into RDF triples in order to make them machine-readable for further use in real systems.

ACKNOWLEDGMENT

We would like to thank the Laboratoire d'informatique de Paris 6 (LIP6), where this work was conducted.

REFERENCES

[1] 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), 2007, Banff, Alberta, Canada. ACM, 2007, pp. 697–706, ISBN: 978-1-59593-654-7, URL: http://doi.acm.org/10.1145/1242572.1242667.

[2] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "Yago2: A spatially and temporally enhanced knowledge base from wikipedia," 2010.

[3] M. Ruiz-Casado, E. Alfonseca, and P. Castells, "Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia," in Proceedings of the 10th International Conference on Natural Language Processing and Information Systems (NLDB), 2005, Alicante, Spain. Springer-Verlag, 2005, pp. 67–79, ISBN: 3-540-26031-5, 978-3-540-26031-8, URL: http://dx.doi.org/10.1007/11428817_7.

[4] S. P. Ponzetto and M. Strube, "Deriving a large scale taxonomy from wikipedia," in Proceedings of the 22Nd National Conference on Artificial Intelligence (AAAI), 2007, Vancouver, British Columbia, Canada. AAAI Press, 2007, pp. 1440–1445, ISBN: 978-1-57735-323-2, URL: http://dl.acm.org/citation.cfm?id=1619797.1619876.

[5] S. Auer and J. Lehmann, "What have innsbruck and leipzig in common? extracting semantics from wiki content," in Proceedings of the 4th European Conference on The Semantic Web: Research and Applications (ESWC), 2007, Innsbruck, Austria. Springer-Verlag, 2007, pp. 503–517, ISBN: 978-3-540-72666-1, URL: http://dx.doi.org/10.1007/978-3-540-72667-8_36.

[6] C. Bizer et al., "DBpedia - A Crystallization Point for the Web of Data," Web Semant., vol. 7, 2009, pp. 154–165, ISSN: 1570-8268, URL: http://dx.doi.org/10.1016/j.websem.2009.07.002.

[7] J. Lehmann et al., "DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia," Semantic Web Journal, vol. 6, 2015, pp. 167–195.

[8] M. Morsey, J. Lehmann, S. Auer, C. Stadler, and S. Hellmann, "DBpedia and the Live Extraction of Structured Data from Wikipedia," Program: electronic library and information systems, vol. 46, 2012, p. 27.

[9] Y. Yan, N. Okazaki, Y. Matsuo, Z. Yang, and M. Ishizuka, "Unsupervised relation extraction by mining wikipedia texts using information from the web," in Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2009, Suntec, Singapore. Association for Computational Linguistics, 2009, pp. 1021–1029, ISBN: 978-1-932432-46-6, URL: http://dl.acm.org/citation.cfm?id=1690219.1690289.

[10] M. Héder and P. N. Mendes, "Round-trip semantics with sztakipedia and dbpedia spotlight," in Proceedings of the 21st In-

ternational Conference on World Wide Web (WWW), 2012, Lyon, France. ACM, 2012, pp. 357–360, ISBN: 978-1-4503-1230-1, URL: http://doi.acm.org/10.1145/2187980.2188048.

[11] M. Szczuka, A. Janusz, and K. Herba, “Clustering of rough set related documents with use of knowledge from dbpedia,” in Proceedings of the 6th International Conference on Rough Sets and Knowledge Technology (RSKT), 2011, Banff, Canada. Springer-Verlag, 2011, pp. 394–403, ISBN: 978-3-642-24424-7, URL: http://dl.acm.org/citation.cfm?id=2050461.2050520.

[12] I. Hulpus, C. Hayes, M. Karnstedt, and D. Greene, “Unsupervised graph-based topic labelling using dbpedia,” in Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM), 2013, Rome, Italy. ACM, 2013, pp. 465–474, ISBN: 978-1-4503-1869-3, URL: http://doi.acm.org/10.1145/2433396.2433454.

[13] G. Quercini and C. Reynaud, “Entity discovery and annotation in tables,” in Proceedings of the 16th International Conference on Extending Database Technology (EDBT), 2013, Genoa, Italy. ACM, 2013, pp. 693–704, ISBN: 978-1-4503-1597-5, URL: http://doi.acm.org/10.1145/2452376.2452457.

[14] Q. Liu et al., “Catriple: Extracting triples from wikipedia categories,” in The Semantic Web, 2008, URL: http://dx.doi.org/10.1007/978-3-540-89704-0_23.

[15] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of english: The penn treebank,” Comput. Linguist., vol. 19, 1993, pp. 313–330, ISSN: 0891-2017, URL: http://dl.acm.org/citation.cfm?id=972470.972475.

[16] S. Zhao and J. Betz, “Corroborate and learn facts from the web,” in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2007, San Jose, California, USA. ACM, 2007, pp. 995–1003, ISBN: 978-1-59593-609-7, URL: http://doi.acm.org/10.1145/1281192.1281299.