Text-Mining, Structured Queries, and Knowledge Management on Web Document Corpora

Hamid Mousavi
CSD, UCLA
Los Angeles, CA
hmousavi@cs.ucla.edu

Maurizio Atzori
University of Cagliari
Cagliari, Italy
atzori@unica.it

Shi Gao
CSD, UCLA
Los Angeles, CA
gaoshi@cs.ucla.edu

Carlo Zaniolo
CSD, UCLA
Los Angeles, CA
zaniolo@cs.ucla.edu

ABSTRACT

Wikipedia's InfoBoxes play a crucial role in advanced applications and provide the main knowledge source for DBpedia and the powerful structured queries it supports. However, InfoBoxes, which were created by crowdsourcing for human rather than computer consumption, suffer from incompleteness, inconsistencies, and inaccuracies. To overcome these problems, we have developed (i) the *IBminer* system that extracts InfoBox information by text-mining Wikipedia pages, (ii) the *IKBStore* system that integrates the information derived by *IBminer* with that of DBpedia, YAGO2, WikiData, WordNet, and other sources, and (iii) SWiPE and InfoBox Editor (*IBE*) that provide a user-friendly interfaces for querying and revising the knowledge base. Thus, *IBminer* uses a deep NLP-based approach to extract from text a semantic representation structure called *TextGraph* from which the system detects patterns and derives subject-attribute-value relations, as well as domain-specific synonyms for the knowledge base. *IKBStore* and *IBE* complement the powerful, user-friendly, by-example structured queries of SWiPE by supporting the validation and provenance history for the information contained in the knowledge base, along with the ability of upgrading its knowledge when this is found incomplete, incorrect, or outdated.

1. INTRODUCTION

Knowledge bases (KBs) are playing a crucial role in many systems, such as text summarization and classification, opinion mining, semantic search, and question answering systems. In recent years, several projects have been devoted to create such KBs [2, 4, 5, 10, 11, 12, 13, 23, 25]. Many of these KBs are based on the structured summaries in Wikipedia, called *InfoBoxes*; each InfoBox summarizes important properties and their values for the entity (subject) described in the Wikipedia's page containing the InfoBox.

In addition to being very valuable for human readers, InfoBoxes and KBs have shown a significant potential of bringing us closer to the realization of the *Semantic Web* vision. For instance, Figure 1 shows the InfoBox for the UCLA Law School. The information harvested from Wikipedia InfoBoxes like this has been stored into RDF-based KBs (e.g., DBpedia), which support powerful *SPARQL* queries. Thus, queries that seek law schools satisfying simple (e.g., 'Bar pass rate' $> 90\%$) can now be expressed in *SPARQL*, along with more complex queries requiring join operations (e.g., law school and business school at the same university), or decision-support aggregates (e.g., law schools with the best faculty/students ratio). This new capability paves the way to powerful Semantic Web applications, but is confronted by the two main obstacles which are discussed next.

UCLA School of Law

Motto	*Fiat lux* (Latin)
Parent school	University of California
Established	1949
School type	Public
Parent endowment	$1.88 billion (June 30, 2009) [1]
Dean	Rachel Moran
Location	Los Angeles, California, U.S.
Enrollment	1,011[2]
Faculty	116–138[2]
***USNWR* ranking**	17[3]
Bar pass rate	85%[2]
Website	www.law.ucla.edu
ABA profile	ABA Law School Data

Figure 1: InfoBox for the UCLA Law School

Ease of Access represents the first major problem, since the knowledge base is now usable only by people who can write *SPARQL* queries—thus casual users are excluded. Even expert programmers will need to spend a fair amount of time to learn DBpedia and thousands of names of entities and properties there used (e.g., names such as: `foaf:givenName` and `dbpprop:populationTotal`). Much progress has been made on the ease-of-access front with the introduction of SWiPE [8] that uses a Query-By-Example (QBE) approach on InfoBoxes treated as

input query forms. For instance, to find law schools with certain properties in Wikipedia, a user will start from the InfoBox of a familiar school (such as that in Figure 1) and replace the existing values of the desired properties with conditions that specify the query. In Section 3, we provide an overview of the enhanced SWiPE which combines more powerful structured-query capabilities (e.g., joins and aggregates) with the free-text keyword-based retrieval capabilities of Web search engines.

Incompleteness and inconsistency represent major issues for current KBs. These problems are largely due to ad-hoc crowdsourcing being used to generate summaries, inasmuch as standard ontologies were either unavailable or ignored during this process. Currently, more than 40% of Wikipedia pages are missing their InfoBoxes entirely, and the others contain InfoBoxes that are often incomplete. A related problem is that when the information is present, it might be represented using synonymous attributes, such as '*birth date*', '*date of birth*', and '*born*', as it is in fact the case for DBpedia and many manually created KBs.

To address these challenges, we employ text-mining techniques to extract more knowledge from unstructured data and integrate knowledge from different knowledge sources. Thus, we have developed the following integrated systems:

IBminer [19] and *OntoMiner* [20, 21] (Section 4), which respectively build structured summaries and ontologies from document corpora using text-mining,

IKBStore and CS^3 [17] (Section 5) that integrate knowledge from different sources and realign subject and attribute names to construct a general KB having the quality and coverage needed for semantic applications, and

IBE [18] (Section 6) that supports the knowledge editing and query functions needed for managing and upgrading the KB, while maintaining provenance information on each piece of knowledge entered into the system.

We will now provide a high-level overview of these systems that provide an integrated set of user-friendly tools whereby the KB can be generated, unified, searched, and also upgraded while minimizing the expertise on KB terminology and system internals required from users.

2. SYSTEM ARCHITECTURE

Figure 2 shows the three-level architecture that supports the three functions of (i) Querying the KB, (ii) Knowledge Mining and (iii) Knowledge Integration and Management.

Querying the KB. We extended the original SWiPE [8] and its user-friendly QBE-like interface to support complex queries that combine joins, aggregates, and keyword-based searches. Thus, queries entered on the InfoBox

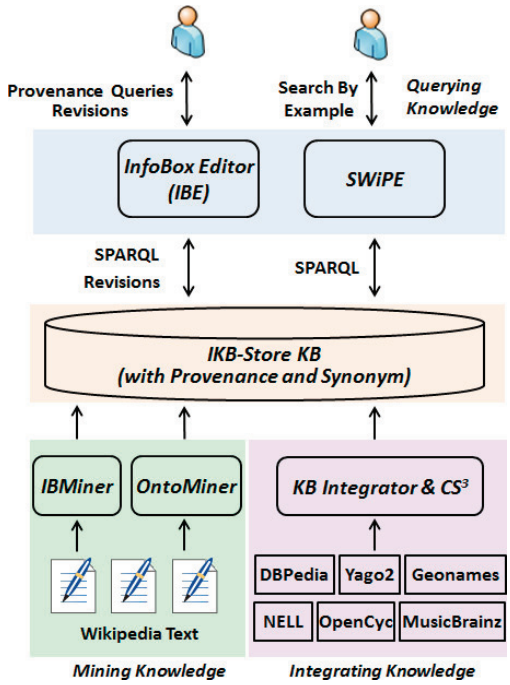

Figure 2: System Architecture

template are translated into SPARQL queries and executed on Virtuoso databases. On the other hand, *IBE* provides a simple interface for upgrading and managing the integrated KB, and also supports temporal and provenance queries on derived information.

Mining Knowledge. To improve coverage of our integrated KB, we developed the *IBminer* and the *OntoMiner* systems, which generate InfoBoxes and ontologies from free text. *IBminer* employs an NLP-based text mining engine called *SemScape* [22] to identify the morphological information in text and generate graph-based structures called TextGraphs. Then, *IBminer* extracts semantic links from TextGraphs using predefined patterns and converts semantic links to final InfoBoxes. In a fashion akin to *IBminer*, *OntoMiner* uses graph pattern rules to mine iteratively ontological information from text.

Integrating Knowledge. To integrate the knowledge from different sources into *IKBStore*, we use the existing interlink information from DBpedia. However, many subjects of interest are not covered by existing links, and links between corresponding attribute names (i.e., synonyms) are limited to the internal ones provided by DBpedia. Thus, the Context-Aware Synonym Suggestion System (CS^3) [17] was developed to discover context-aware synonyms for attributes and subjects. Furthermore, we built the *IBE* system which supports direct auditing and editing of the integrated KB by its curators. Thus, *IBE* was first used to address inconsistencies and other issues that surfaced during the integration of knowledge from different sources previously described,

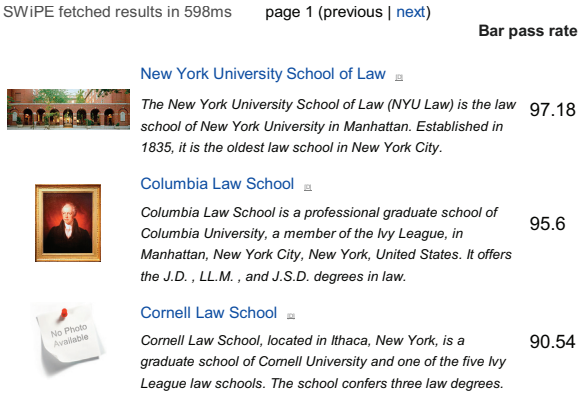

Figure 3: *SWiPE* result page

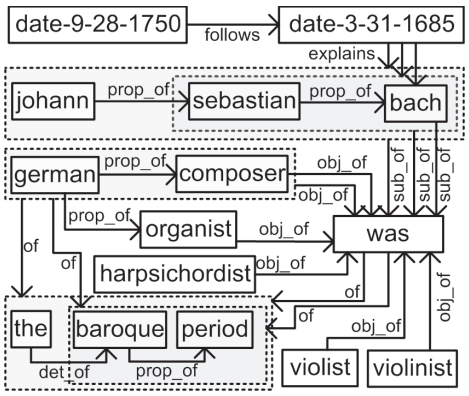

Figure 4: Part of the TextGraph for our example.

and it is currently used to manage our KB and upgrade it with information generated by crowdsourcing. Ease of access is achieved if the system can take users' queries expressed in natural language and translate them into *SPARQL* queries [14]. As discussed in [15], good results can be obtained for simple short queries; in fact, voice input can also be used in very simple ambiguity-free requests. These approaches however cannot handle complex queries, and even for simple ones the risk of misinterpretation is high.

3. BY-EXAMPLE STRUCTURED QUERY

We advocate the use of the ambiguity-free By-Example Structured Query (BEStQ) approach of *SWiPE* [8], that allows users to express with ease a large subset of the queries expressible in *SPARQL* [9]. In *SWiPE*, InfoBoxes are made active and users can enter conditions and constraints on each InfoBox item in a way similar to *Query-By-Example* (QBE). For example, to find in Wikipedia law schools with certain properties, a user starts from the InfoBox of any law school (e.g., that in Figure 1) and replaces the existing values of the desired properties with query conditions. For instance, the user could specify $>90\%$ for the '`Bar pass rate`' and '`New York`' in the `Location` field indicating the *State*. Then *SWiPE* translates this two-line BEStQ specification into the equivalent (22-line long) *SPARQL* query, and executes it on DBpedia using Virtuoso; results returned by Virtuoso are reformatted by *SWiPE* and presented to the user (Figure 3).

The original *SWiPE* prototype described in [8] was recently extended with more powerful structured-query capabilities based on joins, aggregates, and closure properties, and with the keyword-based retrieval capabilities of free-text search engines. The integration of BEStQ and keyword search is made possible by the fact that *SWiPE* displays the original pages from Wikipedia which contain a search box in the right top corner. Thus, in addition to the two conditions previously entered in the InfoBox, our *SWiPE* user could, e.g., enter the words '`ivy league`' in the search box. Then, *SWiPE* will return all Wikipedia pages satisfying both the conditions in the InfoBox and those in the search box. Therefore, the first entry in Figure 3 will be omitted, since 'New York University' is not in the 'Ivy League'. This simple example illustrates the dramatic improvements in precision and recall delivered by BEStQ searches and their synergy with more traditional keyword-based searches. This becomes obvious, if we try to express this query using only keywords and still get reasonable recall and precision[1]. While the integration of BEStQ and keyword search reaches new levels of precision and recall, our goal of producing high-quality answers cannot be reached unless we also tackle the incompleteness and inconsistency problem of the underlying KB. This is the focus of the rest of the paper.

4. KNOWLEDGE FROM TEXT

IBminer [19] and *OntoMiner* [20, 21], described in this section, use a deep NLP-based knowledge extraction approach to improve the completeness, consistency, and accuracy of the KB. The generation of TextGraphs and semantic links from text (Subsection 4.1), and the techniques that (i) learn patterns from TextGraphs and existing KBs and (ii) use them to generate InfoBox triples are discussed in Subsections 4.2, and 4.3. Subsection 4.4 covers *OntoMiner* and text-mining for ontological information.

4.1 TextGraphs and Semantic Links

The first step of knowledge extraction is to convert the sentences of the document into weighted graphs, which

[1]The best keyword combination we found to emulate our previous query is: `Ivy League Law Schools New York bar pass rate`; on this, Wikipedia returns a total of 46 answers. While this represents a major improvement w.r.t. the 1,040,000 results found by Google search, most of those 46 answers are irrelevant or outright ridiculous, such as "List of Batman: The Brave and the Bold Characters."

are called *TextGraphs*. This step is performed using *SemScape*, which uses Co-reference Resolution and other novel techniques to generate high-quality TextGraphs [22]. TextGraphs generated in this way provide a semantic representation of the grammatical connections between words, terms, and phrases through labeled and weighted links. For instance, Figure 4 shows the TextGraph for following sentence:

Example Sentence: "*Johann Sebastian Bach (31 March 1685 – 28 July 1750) was a German composer, organist, harpsichordist, violist, and violinist of the Baroque Period.*"

Once TextGraphs are generated, we use a set of predefined graph rules [6] to produce *Semantic Links* between concepts and terms in TextGraphs. For instance, the following *SPARQL*-like rule is used to produce <*Johann Sebastian Bach*, *was*, *composer*> and similar semantic links:

SELECT (?1 ?3 ?2)
WHERE { ?1 "subj_of" ?3. ?2 "obj_of" ?3.
NOT("not" "prop_of" ?3).
NOT("no" "det_of" ?1).
NOT("no" "det_of" ?2). }

4.2 Learning InfoBox Patterns

To map the semantic links generated in the previous subsection to the standard InfoBox triples, we learn patterns from the matching examples. For instance, consider the two semantic links <*bach*, *was*, *composer*> and <*bach*, *was*, *German*> generated from the TextGraph in Figure 4. Obviously, the link name '*was*' should be interpreted differently in these two cases, since the former is connecting a '*person*' to an '*occupation*', while the latter is between a '*person*' and a '*nationality*'. Now, consider the two InfoBox items <*bach*, *occupation*, *composer*> and <*bach*, *nationality*, *German*> which respectively match the mentioned triples from the text. These items clearly indicate that the link name '*was*' in our two triples should be interpreted respectively as '*occupation*' and '*nationality*'. Thus, from these examples, we learned the following two patterns (also called *Potential Matches* or *PMs*):

- <*cat:Person*, *was*, *cat:Occupation_in_Music*>: *occupation*
- <*cat:Person*, *was*, *cat:German*>: *nationality*

Here the PM $< c_1, l, c_2 >: \alpha$ indicates that the link named l, connecting a subject in category c_1 to a subject or value in category c_2, can be interpreted as the attribute name α. Observe that, instead of using the actual subjects and values in PMs, we used their categorical information to create more general and useful patterns. As a result, for each triple with a matching InfoBox item, we create several patterns since subjects and values usually belong to more than one (direct or indirect) categories. *IBminer* also counts the number of times each PM has occurred.

4.3 Generating Structured Summaries

To extract new structured summaries using PMs, for a given semantic link, say $<s, l, v>$, *IBminer* finds all potential matches such as $<c_s, l, c_v>: \alpha_i$. The resulting set of potential matches are then grouped by the InfoBox attribute names, α_i's. For each group, *IBminer* computes the aggregate frequency of the matches (called *evidence count*). At this point, *IBminer* removes infrequent potential matches using a threshold on normalized frequency, and then applies a type-checking to eliminate implausible matches [16]. Finally the matches remaining in the list are *ranked* according to their evidence count. The match with the largest evidence count, pm, is selected as a new InfoBox tuple $<t_n.s, pm.a_i, t_n.v>$ with confidence $t_n.c \times pm.c$ and evidence $t_n.e$. More details are provided in [16, 19].

4.4 Generating Ontological Information

The *OntoMiner* system uses a successive refinement approach to generate ontological information from text. It performs successive passes, where each pass consists of (i) a relation extraction phase which generates ontological relations between the existing terms and (ii) the concept extraction phase that detects new concepts and aliases using the generated relations. At each pass *OntoMiner* transforms each sentence into a TextGraph assuming the current ontology. New ontological relations between nodes are extracted using predefined graph rules (similar with the ones for generating semantic links). Then, it combines the generated relations, creating a list of ontological relations between terms with their weight and frequency. In the concept extraction phase, *OntoMiner* uses ontological relations between concepts and non-concept terms to detect new concepts and aliases that can be used to enhance the current ontologies. This step can also be performed under the supervision of a human who decides if another cycle of this process is needed or we can stop with the new and improved ontologies.

5. KNOWLEDGE INTEGRATION

To construct a comprehensive KB, we take the knowledge gathered from various public sources and that harvested by our knowledge extraction systems, and integrate these inputs into a KB seeking to achieve superior quality and coverage. A serious obstacle that limits the quality of the result is that different systems do not use a standard terminology, and often describe the same concept or attribute by different names. In this section, we introduce the Context-Aware Synonym Suggestion System (CS^3) which generates synonyms for both subjects and attributes whereby KBs are combined using these synonyms along with other interlinks.

5.1 Generating Synonyms

Different KBs use different terminologies for naming their attributes, and non-unique attribute names might also be used in the same KB. For instance in *DBpedia*, the attribute names '*birthdate*', '*data of birth*', '*born*' are all used to indicate the birthdate of a person, whereas *YAGO2* typically uses '*wasBornOnDate*' to refer to birthdate. Moreover, the attribute name '*born*' is used for both birthdate and birth-place in many systems. Indeed, synonyms are the source of significant ambiguities and inconsistencies.

To address this problem, our systems learn synonym patterns from TextGraph triples [16, 17]. Formally, if TextGraph triple $<s, l, v>$ matches patterns $<c_s, l, c_v>$:α_1 and $<c_s, l, c_v>$:α_2 respectively with evidence frequency f_1 and f_2 ($f_1 \geq f_2$), we create the following *Potential Attribute Synonym* (*PAS*) pattern: $<c_s, \alpha_1, c_v>$: α_2.

This synonym pattern indicates that attribute name α_1 from subject category c_s to value category c_v is also known as α_2. Again, less frequent PAS patterns and those with low support are filtered out and the rest is used to suggest attribute interlinks for existing or newly generated InfoBoxes. Similar techniques are used for learning subject synonyms [16, 17].

5.2 Combining Knowledge Bases

Our first step in building IKBstore consisted in integrating the knowledge extracted from domain-specific KBs, such as MusicBrainz [4] and Geonames [2], and from several general-purpose ones, such as DBpedia [10] and YAGO2 [13]. Every piece of information in our system is represented by an RDF triple <*subject*, *attribute*, *value*>.

The naively integrated KB so obtained contains many synonymous terms and duplicate triples. Thus we utilize the interlinks provided by existing KBs and synonyms generated by CS^3 to improve the consistency of this initial integrated KB, using the techniques described next.

Interlinking Subjects: Fortunately, DBpedia has linked its subjects with other public KBs on the Web. We exploit existing interlinks in DBpedia to combine the information on the same subject from difference sources. For the cases lacking such interlinking information (e.g., NELL [12]), in addition to exact matching, we use synonym and context matching. Synonyms can be obtained from *redirect* and *sameAs* links in DBpedia, WordNet [24], CS^3, and OntoMiner. As for the context, we view identical attributes and values for different subjects as indication of possible matchings.

Interlinking Categories: In addition to exact matching, we compute the similarity of the categories in different KBs on the basis of their instances. Consider two categories c_1 and c_2, and let $S(c)$ be the set of subjects in category c. The similarity function for categories interlink is defined as $Sim(c_1, c_2) = |S(c_1) \cap S(c_2)|/|S(c_1) \cup S(c_2)|$. If the $Sim(c_1, c_2)$ is greater than a certain threshold, we consider c_1 and c_2 as aliases of each other.

After semantic integration, IKBstore contains 9.2 million English subjects and 105.4 million triples. All triples in our integrated KB are assigned accuracy, confidence, and frequency values, as explained in [19]. The sources from which the triples are generated are also stored to support provenance auditing in our KB.

6. INFOBOX EDITOR (IBE)

IBE [18] provides a user-friendly interface to manage InfoBox information while maintaining the provenance of this information. Therefore, *IBE* supports several important functions that streamline the crowdsourcing and management of InfoBox information, including:

1. Resolving inconsistencies in the knowledge collected from various KBs,
2. Extracting knowledge from user provided text by using *IBminer*,
3. Revising the InfoBoxes generated and adding new InfoBoxes to existing subjects,
4. Retrieving and revising meta-level information, such as ontologies and synonyms,
5. Recording the provenance history of InfoBoxes,
6. Supporting simple queries on current KB and its provenance history, along with transaction time queries to flash-back to the past.

While *IBE* is still a work-in-progress (only functions 1–4 from the list are currently implemented), it outlines a new level of functionality required for curated Web corpora to take a central role in advanced applications. Thus the new responsibility of curators will go beyond that of enabling the creation of textual documents and supervising their contents. They will also be responsible for promoting and supervising the process of knowledge creation and integration, crucial for the many applications that rely on the KBs created from Web document corpora. Recent developments, including Wikidata [7], underscore the significance of this trend.

7. EXPERIMENTS AND RESULTS

Extensive experiments were conducted to evaluate the effectiveness of our systems [6, 16, 18, 19, 21, 9]. In this section, we present the results we obtained by applying *IBminer* onto the text of the entire English Wikipedia, which is a corpus containing 4.4 Million subjects each described by 18.2 sentences on average.

The experiments took several weeks on the Hoffman2 cluster at UCLA [3] using up to 256 cores each with 8GB of main memory.

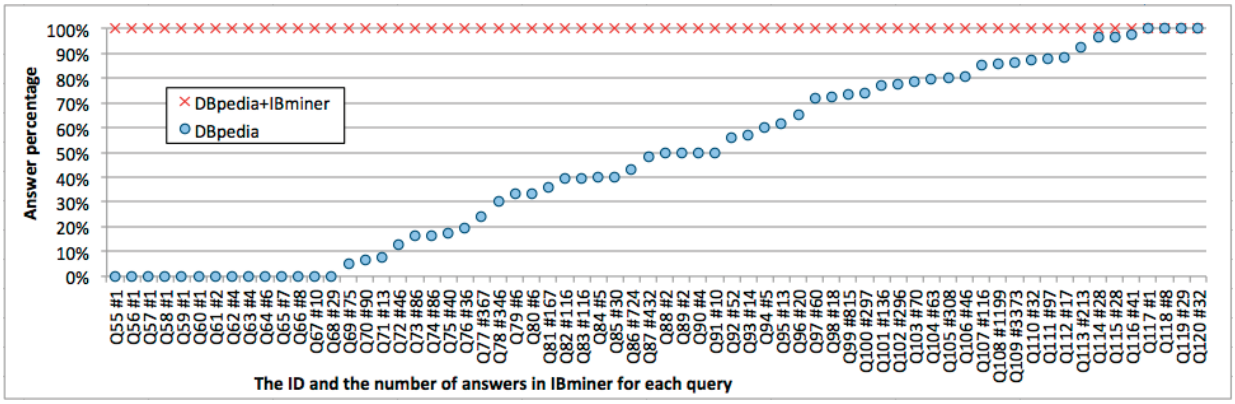

Figure 5: Number of results generated for popular queries using DBpedia and *IBminer*+DBpedia.

The use of *IBminer* on the entire English Wikipedia produced 251 Millions of links where subjects match their page titles. We ranked these links according to their confidence values that measure their supporting evidence (whereas many other links where the subject does not match the page title were simply discharged). In order to decide on the minimum confidence required for the links to be added to our KB, and produce InfoBox triples, we executed the following manual evaluation. We selected 50,000 of such triples and matched them against the text in the page. Therefore for our recall evaluation, we only used those InfoBox triples which match at least one of our semantic links, and measured the recall by computing how many of them are also generated by *IBminer*, whereas precision is measured by the percentage of such triples that are correct. As we lower the confidence threshold required, we obtain a larger recall but a worse precision. For instance, *IBminer* was able to generated 3.9 Million triples with 95% precision. When the threshold was lowered to 90% precision, we are able to recall 7.1 millions of new InfoBox triples.

A more meaningful way to evaluate the usefulness and completeness of a KB is to evaluate the degree of recall it entails for frequently used queries. The next experiment, akin to those performed in [14] and [15], attempts to evaluate the improvement obtained for popular queries on musicians and actors, when the 7.1 Million triples generated by *IBminer* are used.

Popular Queries: In order to create a set of popular queries for our evaluation, we used Google Search AutoComplete system, and found around 150 keyword queries suggested by this system to complete two phrases: "*musicians who*" and "*actors who*". We were able to translate 120 of these keyword-based queries to *SPARQL*. The remaining keyword queries, such as "*Actors who are tall*", "*Musicians who married normal people*", are too vague for a precise translation and quantification and were thus ignored.

Knowledge Bases: Two different KBs are used in this evaluation. As for the baseline KB, we use DBpedia's InfoBox triples. Since the goal is to measure how much *IBminer*'s result improves DBpedia, we combine the triples in DBpedia and *IBminer* into our second KB called IBminer+DBpedia.

After preparing the queries and the KBs, we employed Apache Jena [1], and ran the queries using the two KBs. For more than 44% of the queries, no answer is found from any of the KBs. This very clearly illustrates the incompleteness of the current KBs. Nevertheless, for the remaining queries, *IBminer*+DBpedia produces more answers than the baseline for all queries except four queries in which the additional triples of *IBminer* produced no additional result. Figure 5 shows what portion of the answers found using IBminer+DBpedia are found using only DBpedia's KB. The queries producing no answer are omitted from the figure, where the others are shown sorted by increasing percentages. The number of results found using IBminer+DBpedia is included in the horizontal axis under the query ID [6]. Figure 5 shows that for 11.6% of the queries for which DBpedia is not able to provide any answer, *IBminer* is actually able to find between 1 to 29 answers. Therefore while *IBminer* improves DBpedia's size (coverage) by 21.3%, it improves the completeness of the answers for popular queries by 53.3%.

8. CURRENT WORK & CONCLUSION

In this paper, we have described a set of tools for distilling Web corpora into knowledge bases that will enable structured queries on text documents, making it possible to support more advanced Semantic-Web applications. It is therefore clear that curators of Web corpora must take on responsibilities that go well beyond enabling access to Web documents and supervising their contents. Indeed curator groups must take on the role of KB managers who are responsible for promoting and

supervising the creation of computer-processable document summaries that will support sophisticated Semantic-Web applications and powerful structured queries, such as those provided by *SWiPE* [8].

The importance of curated or semi-curated Web corpora, featuring integrated, well-managed knowledge bases, is underscored by the success of Wikipedia and Wikidata [7] which is revising, completing, and improving current InfoBoxes with interlingual links. These developments represent great news for our project: in fact according to our plan of future work, *IKBStore* will be extended and improved with knowledge taken from Wikidata and Freebase, and *SWiPE* will be extended with multilingual query capabilities. Moreover, we plan to apply our tools and approach to document corpora other than Wikipedia — e.g., medical and technical encyclopedias. For many of these applications, the massive crowdsourcing approach of Wikipedia will not be cost-effective, and our approach that relies on text-mining and various semi-automatic tools will be more desirable.

9. ACKNOWLEDGEMENT

The authors would like to thank the reviewers and the editors for several suggested improvements. This work was partially supported by NSF Grant IIIS-1118107, RAS project CRP-17615 *DENIS: Dataspaces Enhancing Next Internet in Sardinia*, and MIUR PRIN 2010-11 project *Security Horizons*.

10. REFERENCES

[1] Apache Jena. http://jena.apache.org/.
[2] Geonames. http://www.geonames.org/.
[3] Hoffman2 Cluster, UCLA. http://hpc.ucla.edu/hoffman2/.
[4] Musicbrainz. http://musicbrainz.org/.
[5] Opencyc. http://www.cyc.com/platform/opencyc .
[6] Semantic web information management system (swims). http://semscape.cs.ucla.edu/.
[7] Wikidata. http://www.wikidata.org.
[8] M. Atzori and C. Zaniolo. Swipe: searching wikipedia by example. In *WWW (Companion Volume)*, pages 309–312, 2012.
[9] M. Atzori and C. Zaniolo. Expressivity and accuracy of by-example structure queries on wikipedia. CSD Technical Report #140017, UCLA, 2014.
[10] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009.
[11] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.
[12] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
[13] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, 2013.
[14] R. Huang and L. Zou. Natural language question answering over rdf data. In *SIGMOD Conference*, pages 1289–1290, 2013.
[15] Lei Zou et al. Natural language question answering over rdf: a graph data driven approach. In *SIGMOD Conference*, pages 313–324, 2014.
[16] H. Mousavi. *Summarizing Massive Information for Querying Web Sources and Data Streams*. PhD thesis, UCLA, 2014.
[17] H. Mousavi, S. Gao, and C. Zaniolo. Discovering attribute and entity synonyms for knowledge integration and semantic web search. *3rd International Workshop on Semantic Search over The Web*, 2013.
[18] H. Mousavi, S. Gao, and C. Zaniolo. Ibminer: A text mining tool for constructing and populating infobox databases and knowledge bases. *PVLDB*, 6(12):1330–1333, 2013.
[19] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Deducing infoboxes from unstructured text in wikipedia pages. In *CSD Technical Report #130001), UCLA*, 2013.
[20] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Ontoharvester: An unsupervised ontology generator from free text. In *CSD Technical Report #130003), UCLA*, 2013.
[21] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Harvesting domain specific ontologies from text. In *ICSC*, 2014.
[22] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Mining semantic structures from syntactic structures in free text documents. In *ICSC*, 2014.
[23] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. Open mind common sense: Knowledge acquisition from the general public. In *Confederated International Conferences DOA, CoopIS and ODBASE*, London, UK, 2002.
[24] M. M. Stark and R. F. Riesenfeld. Wordnet: An electronic lexical database. In *Proceedings of 11th Eurographics Workshop on Rendering*. MIT Press, 1998.
[25] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*, 2012.