Discovering Attribute and Entity Synonyms for Knowledge Integration and Semantic Web Search

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

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

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

ABSTRACT

In this paper, we propose the *Context-aware Synonym Suggestion System* (CS^3) which learns synonyms from text by using our NLP-based text mining framework, called *SemScape*, and also from existing evidence in the current knowledge bases (KBs). Using CS^3 and our previously proposed knowledge extraction system *IBminer*, we integrate some of the publicly available knowledge bases into one of the superior quality and coverage, called *IKBstore*.

1. INTRODUCTION

The importance of knowledge bases (KBs) in semantic-web applications has motivated the endeavors of several important projects that have created the public-domain KBs shown in Table 1. The project described in this paper seeks to integrate and extend these KBs into a more complete and consistent repository named Integrated Knowledge Base Store (*IKBstore*). *IKBstore* will provide much better support for advanced web applications, and in particular for user-friendly search systems that support Faceted Search [14] and By-Example Structured Queries [5]. Our approach involves the four main tasks of:

Task A: Collecting public KBs, unifying knowledge representation format, and integrating KBs into the *IKBstore* using existing interlinks and structured information.

Task B Completing the integrated KB by extracting more facts from free text.

Task C Generating a large corpus of context-aware synonyms that can be used to resolve inconsistencies in *IKBstore* and to improve the robustness of query answering systems,

Task D Resolving incompleteness in *IKBstore* by using the synonyms generated in Task C.

Task A was greatly simplified by the fact that many projects, including DBpedia [6] and YaGo [15], represent the information derived from the structured summaries of Wikipedia (a.k.a. InfoBoxes) by RDF triples of the form $<subject, attribute, value>$, which specifies the $value$ for an $attribute$ (property) of a $subject$. This common representation facilitates the use of these KBs by a roster of semantic-web applications, including queries expressed in SPARQL, and user-friendly search interfaces [14, 5]. However, inasmuch as the coverage and consistency provided by each individual system remain limited, until we can complete and integrate these KBs at the semantic level.

Task B seeks to complete the initial KB using our knowledge extraction system called *IBminer* [18]. *IBminer* employs an NLP-based text mining framework, called SemScape, to extract *initial triples* from the text. Then using a large body of categorical information and learning from matches between the initial triples and existing InfoBox items in the current knowledge base, *IBminer* translates the initial triples into more standard InfoBox triples.

The integrated KB so obtained will (i) improve coverage, quality and consistency of the knowledge available to semantic web applications and (ii) provide a common ground for different contributors to improve the KBs in a more standard and effective way. However, a serious obstacle in achieving such desirable goal is that different systems do not adhere to a standard terminology to represent their knowledge, and instead use plethora of synonyms and polynyms.

Thus, we need to resolve *synonyms* and *polynyms* for the entity names as well as the attribute names. For example, by knowing '*Johann Sebastian Bach*' and '*J.S. Bach*' are synonyms, the KB can merge their triples and associate them with one single name. As for the polynyms, the problem is even more complex. Most of the time based on the context (or popularity), one should decide the correct polynym of a vague term such as '*JSB*' which may refer to "*Johann Sebastian Bach*", '*Japanese School of Beijing*', etc. Several efforts to find entity synonyms have been reported in recent years [9, 10, 11, 13]. However, the synonym problem for attribute names has received much less attention, although they can play a critical role in query answering. For instance, the attribute '*birthdate*' can be represented with terms such as '*date of birth*', '*wasbornindate*', and '*born*' in different (or same) KBs when used in different contexts. Unless these synonyms are known, a search for musicians born, say, in 1685 is likely to produce a dismal recall.

To address the aforementioned issues, we proposed our Context-aware Synonym Suggestion System (CS^3). CS^3 performs mainly tasks C and D by first extracting context-aware attribute and entity

Name	Size (MB)	Entities # (10^6)	Triples # (10^6)
ConceptNet [21]	3075	0.30	1.6
DBpedia [6]	43895	3.77	400
FreeBase [7]	85035	≈25	585
Geonames [2]	2270	8.3	90
MusicBrainz [3]	17665	18.3	≈131
NELL [8]	1369	4.34	50
OpenCyc [4]	240	0.24	2.1
YaGo2 [15]	19859	2.64	124

Table 1: Some of the publicly available Knowledge Bases

Proceedings of the VLDB Endowment, Vol. 6, No. 12

synonyms, and then using them to improve the consistency of *IKBstore*. CS^3 learns attribute synonyms by matching morphological information in free text to the existing structured information. Similar to *IBminer*, CS^3 takes advantage of a large body of categorical information available in Wikipedia, which serves as the contextual information. Then, CS^3 improves the attribute synonyms so discovered, by using triples with matching subjects and values but different attribute names. After unifying the attribute names in different KBs, CS^3 finds subjects with similar attributes and values as well as similar categorical information to suggest more entity synonyms. Through this process, CS^3 uses several heuristics and takes advantage of currently existing interlinks such as DBpedia's *alias*, *redirect*, *externalLink*, or *sameAs* links as well as the interlinks provided by other KBs. In this paper, we describe the following:

- The Context-aware Synonym Suggestion System (CS^3) which generates synonyms for both entities and attributes in existing KBs. CS^3 uses free text and existing structured data to learn patterns for suggesting attribute synonyms. It also uses several heuristics to improve existing entity synonyms.
- Novel techniques as well as both *IBminer* and CS^3 were used to integrate several pubic KBs and convert them into a general knowledge base. This improves performance of semantic search over our knowledge base, since more standard and specific terms are used for both entities and attributes.
- We performed preliminary experiments on public KBs, namely DBpedia and YaGo using Wikipedia pages. The results obtained show that CS^3 greatly improves the quality and coverage of the existing KBs.

Applications: *IKBstore* can benefit a wide variety of applications, since it covers a large number of structured summaries represented with a standard terminology. Knowledge extraction and population systems such as *IBminer* [18] and *OntoMiner* [19], knowledge browsing tools such as DBpedia Live [1] and InfoBox Knowledge-Base Browser (IBKB)[17], and semantic web search such as *Faceted Search* [14] and By-Example Structured queries [5] are three prominent examples of such applications. In particular for semantic web search, *IKBstore* improves the coverage and accuracy of structured queries due to superior quality and coverage with respect to existing KBs. Moreover, *IKBstore* can serve as a common ground for different contributors to improve the KBs in a more standard and effective way. Using multiple KBs in *IKBstore* can also be a good mean for verifying the correctness of the current structured summaries as well as those generated from the text.

2. INFOBOXES FROM TEXT

The first step in our process consists in performing the nontrivial task of generating structured data from text, we use our *IBminer* system [18]. Briefly, *IBminer*'s process can be divided into four high-level steps. The first step is to parse the sentences in text and convert them to a more machine friendly structure called TextGraphs which contain grammatical between terms and entities mentioned in the text. This step is performed by the NLP-based text mining framework *SemScape* [18]. Next as the second step, *IBminer* uses a small set of manually created SPARQL-like patterns (59 patterns) to generate semantic links between entities in the form of $<subject, link, value>$ triples. These triples are referred to as the *initial triples*. The third step is to learn a structure called *Potential Match* (*PM*). *PM* contains context-aware potential matches between semantic links in the TextGraphs and attribute names in current InfoBox items. As the forth step, PM is used to suggest the final structured summaries (InfoBoxes) from the initial triples. *IBminer* performs this part by using a large body of categorical information provided by Wikipedia.

3. CONTEXT-AWARE SYNONYMS

Synonyms are terms describing the same concept, which can be used interchangeably. According to this definition, no matter what context is used, the synonym for a term is fixed (e.g. '*birthdate*' and '*date of birth*' are always synonyms). However, polynymous and homonymous phrases are in fact much more prevalent than exact synonyms in the KBs. For example, consider the entity/subject name '*Johann Sebastian Bach*'. Due to its popularity, a general understanding is that the entity is describing the famous German classical musician. However, what if we know that for this specific entity the birthplace is in '*Berlin*'. This simple contextual information will lead us to the conclusion that the entity is refereing to the painter who was actually the grandson of the famous musician '*Johann Sebastian Bach*'. A very similar issue exists for the attribute synonyms.

To take advantage of contextual information for more effectively extracting attribute synonyms, CS^3 constructs a structure called *Potential Attribute Synonyms* (*PAS*). In the generation of *PAS*, CS^3 essentially counts the number of times each pair of attributes are used between the same [set of] subjects and [set of] values and with the same corresponding semantic link in the TextGraphs. The context in this case is considered to be the categorical information for the subject and the value. These numbers are then used to compute the probability that any given two attributes in the context of their subjects and their values are synonyms.

Attribute Synonyms. Intuitively, if two attributes (say '*birthdate*' and '*dateOfBirth*') are synonyms in a specific context, they should be represented with the same (or very similar) semantic links in the TextGraphs (e.g. with semantic links such as '*was born on*', '*born on*', or '*birthdate is*'). In simpler words, we use text as the witness for our attribute synonyms. Moreover, the context, which is defined as the categories for the subjects (and for the values), should be very similar for synonymous attributes.

More formally, assume *IBminer* finds two attributes α_i and α_j that match link l in initial triple $<s, l, v>$. Let $N_{i,j}$ $(= N_{j,i})$ be the total number of times both α_i and α_j are the interpretation of the same link (in the initial triples) between category sets C_s and C_v. Also, let N_x be the total number of time α_x is used between C_s and C_v. Thus, the probability that α_i (α_j) is a synonym for α_j (α_i) can be computed by $N_{i,j}/N_j$ $(N_{i,j}/N_i)$. Obviously this is not always a symmetric relationship (e.g. '*born*' attribute is always a synonym for '*birthdate*', but not the other way around, since '*born*' may also refer to '*birthplace*' or '*birthname*' as well). In other words having N_i and $N_{i,j}$ computed, we can resolve both synonyms and polynyms for any given context (C_s and C_v).

With the above intuition in mind, the goal in *PAS* is to compute N_i and $N_{i,j}$. CS^3 constructs *PAS* in one-pass algorithm which is essential for scaling up our system. For each two records in *PM* such as $<c_s, l, c_v>$: α_i and $<c_s, l, c_v>$: α_j respectively with evidence frequency e_i and e_j $(e_i \leq e_j)$, we add records $<c_s, \alpha_i, c_v>$: α_j and $<c_s, \alpha_j, c_v>$: α_i to *PAS*, both with the same evidence frequency e_i. Note that, if the records are already in the current *PAS*, we increase their evidence frequency by e_i.

At the very same time we also count the number of times each attribute is used between a pair of categories. This is necessary for estimating N_i. Thus for the case above, we add records $<c_s, \alpha_i, c_v>$: '' (with evidence e_i) and $<c_s, \alpha_j, c_v>$: '' (with evidence e_j) to PAS.

Improving PAS with Matching InfoBox Items. Potential attribute synonyms can be also derived from different KBs which contain the same piece of knowledge, but with different (synonymous) attribute names. For instance let $<J.S.Bach, birthdate, 1685>$ and $<J.S.Bach, wasBornOnDate, 1685>$ be two InfoBox triples in different KBs indicating $bach$'s birthdate. Since the subject and value part of the two triples match, one may say $birthdate$ and $wasBornOnDate$ are synonyms. To add these types of synonyms to the *PAS* structure, we follow the exact same steps explained in the previous part.

Generating Final Attribute Synonyms. Once *PAS* structure is built, it is easy to compute attribute synonyms as described earlier. Assume we want to find best synonyms for attribute α_i in InfoBox Triple t=$<s, \alpha_i, v>$. Using *PAS*, for all possible α_j, all $c_s \in C_s$, and all $c_v \in C_v$, we aggregate the evidence frequency (e) of records such as $<c_s, \alpha_i, c_v>$: α_j in *PAS* to compute $N_{i,j}$. Similarly, we compute N_j by aggregating the evidence frequency (e) of all records in the form of $<c_s, \alpha_i, c_v>$: ''. Finally, we only accept attribute α_j as the synonym of α_j, if $N_{i,j}/N_i$ and $N_{i,j}$ are respectively above predefined thresholds τ_{sc} and τ_{se}.

Entity Synonyms. There are several techniques to find entity synonyms [20, 22, 12, 13, 23, 19]. Although performing very well on suggesting context-independent synonyms, they do not explicitly consider the contextual information for suggesting more appropriate synonyms and resolving polynyms and homonyms. To define context-aware entity synonyms, for each entity name, CS^3 uses the categorical information of the entity as well as all the InfoBox triples of the entity as the contextual information for that entity. In other words, two entities that have many similar categories and attribute/value pairs in common are more probable to be synonyms. Thus to complete the exiting entity synonym suggestion techniques, for any suggested pair of synonymous entities, we compute entities context similarity to verify the correctness of the suggested synonym. We should note that this approach should be used as a complementary technique over the existing ones for practicality issues. In this work, we use the OntoHarvester system [19] in addition to simple string matching techniques.

4. COMBINING KNOWLEDGE BASES

We are currently in the process of integrating KBs listed in Table 1. For all KBs, we convert their knowledge into RDF triples and store them in *IKBstore* which is implemented over Apache Cassandra. We currently recognize three main types of information in *IKBstore*:

i) *InfoBox triples* which provide information on a known subject ($subject$) in the $<subject, attribute, value>$ format (e.g. *<J.S. Bach, PlaceOfBirth, Eisenach>* which indicates the birthplace of the subject *J.S.Bach* is *Eisenach*.), **ii) *Subject/Category triples*** which provide the categories that a subject belongs to in the form of *subject/link/category* where, *link* represents a taxonomical relation (e.g. *<J.S.Bach, is in, Cat:Composers>* which indicates the subject *J.S.Bach* belongs to the category *Cat:Composers*.), and **iii) *Category/Category triples*** that represent taxonomical links between categories (e.g. *<Cat:Composers, is in, Cat:Musicians>* which indicates the category *Cat:Composers* is a sub-category of *Cat:Musicians*.).

IKBstore also preserves the provenance of each piece of knowledge. In other words, for every fact in the integrated knowledge base, we can track its data source. For the facts derived from text, we record the article ID as the provenance. In fact, we also annotate each fact with accuracy confidence and frequency values, based on the provenance of the fact [18] [16].

Data set Name	Subjects	InfoBox Triples	Subjects with Abstract	Subjects with InfoBox	Sentences per Abstract
Musicians	65835	687184	65476	52339	8.4
Actors	52710	670296	52594	50212	6.2
Institutes	86163	952283	84690	54861	5.9

Table 2: Description of the data sets used in our experiments.

Knowledge Integration. During the initial knowledge integration phase we discover interlinks between i) subjects, ii) attributes, and iii) categories from the various knowledge sources to eliminate duplication, align attributes, and reduce inconsistency. Such information is partially provided for subjects by some of the KBs through interlinks to DBpedia. However, for attributes and categories it is completely missing. For these, we use simple matching techniques as explained in [16]. Using these interlinking techniques, we create an initial KB by naively integrating all the existing ones. The provenance information for each piece of knowledge is also stored along with the triples.

Once the initial KB is ready, we employ *IBminer* to extract more structured data from accompanying text and then utilize CS^3 to resolve synonyms, improve the inconsistency, and create the final knowledge base. More specifically, we perform the following steps in order to complete and integrate the final knowledge base:

Improving Knowledge Base Coverage: As described in Section 2, *IBminer* derives InfoBox triples from free text using the initial knowledge base. Adding these triples to *IKBstore* will greatly improve the coverage. For each generated triple, we also update the confidence and evidence frequency in *IKBstore*.

Realigning Attributes: Next, we employ CS^3 to discover synonyms for attribute names and expand the initial KB with more common and standard attribute names.

Matching Entity Synonyms: This step merges the entities base on the entity synonyms suggested by CS^3 (Section 3). For the suggested synonymous entities such as s_1, s_2, we aggregate their triples and use one common entity name, say s_1. The other subject (s_2) is considered as a possible alias for s_1, which can be represented by RDF triple $<s_1, alias, s_2>$.

Integrating Categorical Information: Since we have merged subjects based on entity synonyms, the similarity score of the categories may change and thus we need to rerun the category integration described in [16].

5. EXPERIMENTAL RESULTS

To evaluate our system, we create an initial KB using subjects listed in Wikipedia for three specific domains (Musicians, Actors, and Institutes) shown in Table 2. These data sets do not share any subject, and in total they cover around 7.9% of Wikipedia subjects. For these subjects, we add their related structured data from DBpedia and YaGo2 to our initial KBs. As for the text, we use Wikipedia's long abstracts for the mentioned subjects. All the experiments are performed in a single machine running Ubuntu12 with 16 cores of 2.27GHz and 16GB of main memory.

Although our initial KB contains three data sets and we have performed all four tasks on these data sets, we only report the evaluation results for the Musicians data set due to space issue. Very similar results are achieved for the other data sets.

Completing Knowledge by IBminer. Using the Musicians data set we trained the Potential Match (*PM*) structure using *IBminer* system and generate the final InfoBox triples without setting any confidence and evidence frequency threshold (i.e. $\tau_e = 0$ and $\tau_c = 0.0$). To estimate the accuracy of the final triples, we randomly select $20K$ of the generated triples and carefully grade them by matching against their abstracts. As for the recall, we investigate

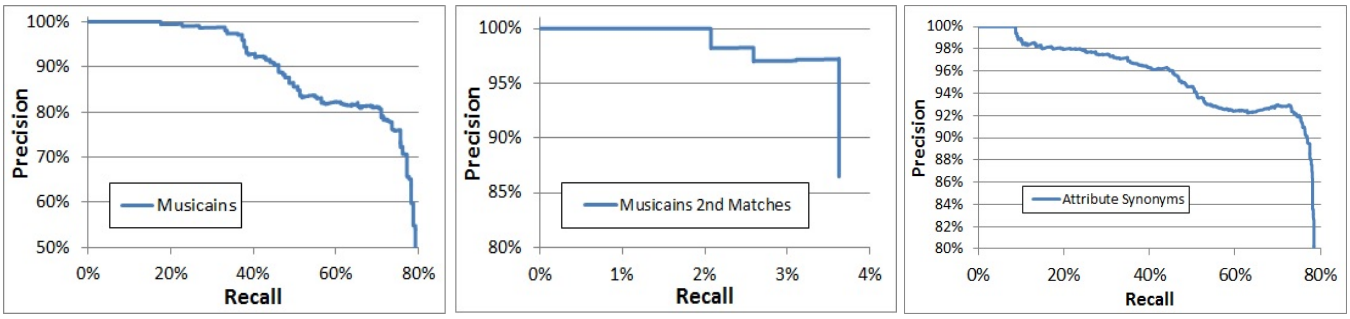

Figure 1: Results for Musicians data set: a) Precision/Recall diagram for best matches, b) Precision/Recall diagram for attribute synonyms, and c) Precision/Recall diagram for the attribute synonyms generated for existing InfoBoxes in the Musicians data set.

existing InfoBoxes and compute what portion of them is also generated by *IBminer*. This gives only a pessimistic estimation of the recall ability, since we do not know what portion of the InfoBoxes in Wikipedia are covered or mentioned in the text (long abstract for our case). To have a better estimation for recall, we only used those InfoBox triples which match at least to one of our initial triples. In this way, we estimate based on InfoBox triples which are most likely mentioned in the text.

Best Matches: To ease our experiments, we combine our two thresholds (τ_e and τ_c) by multiplying them together and create a single threshold called $\tau(=\tau_e \times \tau_c)$. Part a) in Figure 1 depicts the precision/recall diagram for different thresholds on τ in Musicians data set. As can be seen, for the first 15% of coverage, *IBminer* is generating only correct information. For these cases τ is very high. According to this diagram, to reach 97% precision which is higher than DBpedia's precision, one should set τ to 6,300. For this case, *IBminer* generates around $96.7K$ triples with 33.6% recall.

Secondary Matches: For each best match found in the previous step, say $t = <s, \alpha_i, v>$, we generate attribute synonyms for α_i using CS^3. If any of the reported attribute synonyms are not in the list of possible matches for t, we ignore the synonym. Considering τ_e=.12$|T_m|$, precision and recall of the remaining synonyms are computed similar to the best match case and depicted in part b) of Figure 1 (while the potential attribute synonym evidence count (τ_{se}) decreases from right to left). As the figure indicates, by setting accuracy to 97%, we improve the recall of the best matches by 3.6%. Although this seems to be only an small improvement, the number of correct new triples that we generate is $53.6K$ which is quite comparable to those generated as the best matches.

By aggregating the above two cases, we can reach up to 97% accuracy while the total number of generated results is more than $150K$ ($96.7K + 53.6K$). This indicates 28.7% improvement in the coverage of our Initial KB while the accuracy is above 97%.

Completing Knowledge by Attribute Synonyms. In order to evaluate the attribute synonyms generated by CS^3, we use the Musicians data set and construct the PAS structure. Using PAS, we generated synonyms for $10,000$ InfoBox items in our initial knowledge base. We should mention that these synonyms are for already existing InfoBoxes which differentiate them from secondary matches discussed in Subsection 5. This has generated more than $14,900$ synonym items. Among the $10,000$ InfoBox items, 1263 attribute synonyms were listed and our technique generated 994 of them. We used these matches to estimate the recall value of our technique for different frequency thresholds (τ_{cs}) as shown in part c) of Figure 1. As for the precision estimation, we manually graded the generated synonyms. As can be seen in the figure, CS^3 is able to find more than 74% of the possible synonyms with more than 92% accuracy. In fact, this is a very big step in improving structured query results, since it increase the coverage of the *IKBstore* by at least 88.3%. This to some extents improves the consistency of the KBs terminology by providing more synonymous InfoBox triples. In aggregate with the improvement we achieved by *IBminer*, we can state that our *IKBstore* doubles the size of the current KBs while preserving their precision (if not improving) and significantly improving their consistency.

6. REFERENCES

[1] Dbpedia live. `http://live.dbpedia.org/`.

[2] Geonames. `http://www.geonames.org`.

[3] Musicbrainz. `http://musicbrainz.org`.

[4] Opencyc. `http://www.cyc.com/platform/opencyc`.

[5] M. Atzori and C. Zaniolo. Swipe: searching wikipedia by example. In *WWW (Companion Volume)*, pages 309–312, 2012.

[6] 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.

[7] 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.

[8] 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.

[9] K. Chakrabarti, S. Chaudhuri, T. Cheng, and D. Xin. A framework for robust discovery of entity synonyms. In *Proceedings of the 18th ACM SIGKDD*, KDD '12, pages 1384–1392. ACM, 2012.

[10] S. Chaudhuri, V. Ganti, and D. Xin. Exploiting web search to generate synonyms for entities. In *WWW*, pages 151–160. ACM, 2009.

[11] S. Chaudhuri, V. Ganti, and D. Xin. Mining document collections to facilitate accurate approximate entity matching. *Proc. VLDB Endow.*, 2(1):395–406, Aug. 2009.

[12] T. Cheng, H. W. Lauw, and S. Paparizos. Fuzzy matching of web queries to structured data. In *ICDE*, pages 713–716, 2010.

[13] T. Cheng, H. W. Lauw, and S. Paparizos. Entity synonyms for structured web search. *IEEE Trans. Knowl. Data Eng.*, 24(10):1862–1875, 2012.

[14] R. Hahn, C. Bizer, C. Sahnwaldt, C. Herta, S. Robinson, M. Bürgle, H. Düwiger, and U. Scheel. Faceted wikipedia search. In *BIS*, pages 1–11, 2010.

[15] 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.

[16] H. Mousavi, S. Gao, and C. Zaniolo. Discovering attribute and entity synonyms for knowledge integration and semantic web search. In *CSD Technical Report #130013), UCLA*, 2013.

[17] H. Mousavi, S. Gao, and C. Zaniolo. Ibminer: A text mining tool for constructing and populating infobox databases and knowledge bases. *VLDB (demo track)*, 2013.

[18] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Deducing infoboxes from unstructured text in wikipedia pages. In *CSD Technical Report #130001), UCLA*, 2013.

[19] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Ontoharvester: An unsupervised ontology generator from free text. In *CSD Technical Report #130003), UCLA*, 2013.

[20] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, Mar. 2001.

[21] 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.

[22] M. M. Stark and R. F. Riesenfeld. Wordnet: An electronic lexical database. In *Proceedings of 11th Eurographics Workshop on Rendering*. MIT Press, 1998.

[23] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of EMCL*, pages 491–502, London, UK, UK, 2001. Springer-Verlag.