Record matching over duplicate records results from multiple web databases

(Dr.J.PRADEEP KUMAR)[1] (N.KIRANMAI)[2] (M.BALA JYOTHI)[3]
[1](ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, ADITYA COLLEGE OF ENGINEERING, MADANAPALLE, INDIA)
[2](STUDENT, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, ADITYA COLLEGE OF ENGINEERING, MADANAPALLE, INDIA)
[3](STUDENT, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, ADITYA COLLEGE OF ENGINEERING, MADANAPALLE, INDIA)

ABSTRACT

Data consolidation is a challenging issue in data integration. The usefulness of data increases when it is linked and fused with other data from numerous (Web) sources. The promise of Big Data hinges upon addressing several big data integration challenges, such as record linkage at scale, real-time data fusion, and integrating Deep Web. Although much work has been conducted on these problems, there is limited work on creating a uniform, standard record from a group of records corresponding to the same real-world entity. We refer to this task as record normalization. Such a record representation, coined normalized record, is important for both front-end and back-end applications. In this paper, we formalize the record normalization problem, present in-depth analysis of normalization granularity levels (e.g., record, field, and value-component) and of normalization forms (e.g., typical versus complete).

Index Terms—Record normalization, data quality, data fusion, web data integration, deep web

INTRODUCTION

The Web has evolved into a data-rich repository containing a large amount of structured content spread across millions of sources. The usefulness of Web data increases exponentially (e.g., building knowledge bases, Web-scale data analytics) when it is linked across numerous sources. Structured data on the Web resides in Web databases and Web tables. Web data integration is an important component of many applications collecting data from Web databases, such as Web data warehousing (e.g., Google and Bing Shopping; Google Scholar), data aggregation (e.g., product and service reviews), and met searching. Integration systems at Web scale need to automatically match records from different sources that refer to the same real-world entity find the true matching records among them and turn this set of records into a standard record for the consumption of users or other applications. There is a large body of work on the record matching problem and the truth discovery problem. The record matching problem is also referred to as duplicate record detection, record linkage, object identification, entity resolution or reduplication and the truth discovery problem is also called as truth finding or fact finding a key problem in data fusion. In this we assume that the tasks of record matching and truth Discoveries have been performed and that the groups of true matching records have thus been identified. Our goal is to generate a uniform, standard record for each group of true matching records for end-user consumption. We call the generated record the normalized record. We call the problem of computing the normalized record for a group of matching records the record normalization problem (RNP), and it is the focus of this work. RNP is another specific interesting problem in data fusion. Record normalization is important in many application domains. For example, in the research publication domain, although the integrator website, such as Cite seer or Google Scholar, contains records gathered from a variety of sources using automated extraction techniques, it must display a normalized record to users. Otherwise, it is unclear what can be presented to users: present the entire groups of matching records or simply present some random record from the group, to just name a couple of ad-hoc approaches. Quality of data (QoD) is a multidimensional, complex concept. Some of the significant quality dimensions are data completeness, data uniqueness, data consistency, data accuracy and data freshness.Data quality is an important property to be considered while providing access to large volume of data from alternative sources and conveying alternative query answers to end users. Due to heterogeneities in data with different data qualities, data integration has become difficult task. Comprehensive data quality knowledge is essential for data integration due to high diversity of data sources. Successful data integration mainly aims in providing better quality data to the end users. Duplicate data will affect the quality of data and reduplications is carried out to detect and remove the duplicate from resultant data. The widely used duplicate detection techniques are field matching and duplicate record detection. Character-based, token base, phonetic and numeric are popular similarity metrics in field matching techniques for identification of duplicates. Probabilistic matching models, supervised and semi supervised, active learning techniques, distance based techniques and rule based approaches are widely used for duplicate detection in detecting duplicate records method.

The contribution of this paper is to implement duplication detection and incompleteness detection and resolution approach for duplication detection and completeness of end users' data in data integration. The duplicate records are detected and resolved using Record Linkage approach and Weighted Component Similarity Summing (WCSS) approach. The incompleteness of end users' data is detected and resolved using various completeness like source completeness, tuple completeness and attribute completeness Either of these choices can lead to a frustrating experience for a user, because in the user needs to sort/browse through a potentially large number of duplicate records, and in we run the risk of presenting a record with missing or incorrect pieces of data. For example, Table 1 contains four records corresponding to the same entity (publication). They are extracted from different websites. Record R norm is constructed by hand for illustration purposes. One notices that the same publication has different representations in different websites format"last-name, first-name-initial" in the record Ra, but the values of the same field in the records Rb, Rc, and Rd use the format first-name-initial. Last-name One can also observe that the value of the field pages is absent in Ra. The field venue has incomplete values in three of the four records and has no value in Rd; it contains the abbreviations "proc", "int", "conf" to represent "proceedings", "international" and "conference", respectively, in the records Ra and Rc it contains the acronym "VLDB" to represent "Very Large Data Bases" while missing "proceedings of the 32nd international conference on" in Some values of the attributes of R norm cannot be acquired directly from the given set of matching records, such as the first names of the authors.

They could be obtained by mining external sources, such as a search engine. In this paper, we focus on the best effort record normalization: we compute R norm from the set of matching records and do not explore external sources. Furthermore, this paper only focuses on the normalization of text data, and we will leave the normalization of data involving numeric and more complex values as future work. Record level assumes that the values of the fields within a record are governed by some hidden criterion and that together create a cohesive unit that is user-friendly. As a consequence, this normalization favors building the normalized record from entire records among the set of matching records rather than piecing it together from field values of different records. Thus, any of the matching records (ideally, that has no missing values) can be the normalized record. Using our running example in the record Rc is a possible choice for the normalized record with this level of normalization granularity.

Proposed System

We proposed four single-strategy approaches: frequency, length, centroid, and feature-based to select the normalized record or the normalized field value. For multi strategy approach, we used result merging models inspired from Meta searching to combine the results from a number of single strategies. We analyzed the record and field level normalization in the typical normalization. In the complete normalization, we focused on field values and proposed algorithms for acronym expansion and value component mining to produce much improved normalized field values. We implemented a prototype and tested it on a real-world dataset.

ALGORITHM

Mining Abbreviation-Definition Pairs

We use a number of heuristics to determine whether given two value components s and t, s is an abbreviation of t. In this section, a value component is a word (or term). As we mentioned in this we consider only fields with the string data type. We define the neighboring context of a word w within the set of values of a field fj as the set of pairs (leaf t neighbor word, right neighbor word) with the property that the substring leaf t neighbor word w right neighbor word is a substring of a value of fj in some record in Re. If w is the beginning word of a field value, we use a special start-symbol "⟨s⟩" to mark leaf t neighbor word. If it is the last word in the field value, we use the special end-symbol "⟨/s⟩" to mark right neighbor word. For example, the words "proceedings" and "proc" occur many times in the field venue, and they share a good fraction of their neighboring contexts, such as (in, of), (⟨s⟩, of), (in, acm). "Proc" is also the prefix of "proceedings", so we become increasingly confident that "proc" is a possible abbreviation of "proceedings". The algorithm for finding pairs of the form (s, t), where s is an abbreviation of t, is given in starts with initializing two sets: c words and AW P, where c words stores the words that are likely to have abbreviations and AW P stores the final abbreviation word pairs. In line 2, the function tokenizes segments all field values in value into individual words and stores them into p words. In line the function unique looks for unique words and stores them into u words. In lines the words in u words whose lengths are larger than a threshold and there are less than a threshold become candidate words with abbreviations. They are stored into c words and are empirically set the function get Words By Same Context looks for the possible abbreviated words for each u word in u words. It accomplishes this task by measuring the size of the intersection of the neighboring contexts of u word and c word. Then it sorts the words in descending order of the size of the intersection with the neighboring context of u word, retains only the top npos of them and returns them in the set pa words. In line the function get Abbreviations finds the words in pa words that are prefixes of c word. It returns them in abbreviations. For each abbreviation in abbreviations, the pair (abbreviation, c word) is inserted into AW P. Finally, the algorithm returns AW P.

Mining Template Collocations and Sub collocations

Let an n-collocation be a template collocation and a k collocation kc be its sub collocation (k < n). We observe that a number of rules govern the expansion process of kc to tc.

Rule 1: If kc is a sub collocation of a (k+1)-collocation k1c, and the extra word in k1c is a preposition (e.g., "in" and "on") or an article (e.g., "the", "a", and "an"), we can expand kc to k1c.

Consider kc = "proceedings of" and k1c = "proceedings of the", k = 2. kc is a sub collocation of k1c and "the" is a preposition. Thus, "proceedings of" can be expanded to "proceedings of the." In another example, "conference on" is the sub collocation of "international conference on" and the distinct word "international" is neither a preposition nor an article, so we cannot expand "conference on" to "international conference on". Not every venue has the word "international," which suggests that this expansion is infeasible in practice.

Rule 2 (Transitivity): If a k-collocation can be expanded to a (k+1)-collocation k1c and k1c can be expanded to a (k+2)-collocation k2c, then can be expanded to k2c.

For example, “proceedings of” can be expanded to “in proceedings of the” via “proceedings of the”. The transitive property is an immediate consequence of Rule 1. Thus, we can use it to expand kc to tc.

Rule 3: All one word collocations (i.e., k = 1) are nouns. We use POS tagger in NLTK to get the part of speeches of the words in the experimental studies. Using the above analysis, we aim to find all template collocations and their sub collocations. The template collocations become the candidates with which we can expand (replace) the sub collocations. They will be used to generate the normalized component values for a field. The algorithm of finding template collocations and its sub collocations is given in the input of the algorithm CV al(fj) is the updated version of V al(fj), the collection of all values of the field fj , where the abbreviations are extended by the output is a set of pairs T CSP. A pair (tc, Stc) in T CSP denotes a template collocation tc and its set of sub collocations Stc. We will use the output to replace the occurrence of an element in Stc in some value of the field fj with tc when we build the normalized record.

CONCLUSION

We studied the problem of record normalization over a set of matching records that refer to the same real-world entity. We presented three levels of normalization granularities (record-level, field-level and value component level) and two forms of normalization (typical normalization and complete normalization). For each form of normalization, we proposed a computational framework that includes both single-strategy and multi-strategy approaches. In the future, we plan to extend our research as follows. First, conduct additional experiments using more diverse and larger datasets. The lack of appropriate datasets currently has made this difficult. Second, investigate how to add an effective human-in-the-loop component into the current solution as automated solutions alone will not be able to achieve perfect accuracy. Third, develop solutions that handle numeric or more complex values.

REFERENCES

[1] K. C.-C. Chang and J. Cho, “Accessing the web: From search to integration,” in SIGMOD, 2006, pp. 804–805.

[2] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, “Webtables: Exploring the power of tables on the web,” PVLDB, vol. 1, no. 1, pp. 538–549, 2008.

[3] W. Meng and C. Yu, Advanced Metasearch Engine Technology. Morgan & Claypool Publishers, 2010. [4] A. Gruenheid, X. L. Dong, and D. Srivastava, “Incremental record linkage,” PVLDB, vol. 7, no. 9, pp. 697–708, May 2014.

[5] E. K. Rezig, E. C. Dragut, M. Ouzzani, and A. K. Elmagarmid, “Query-time record linkage and fusion over web databases,” in ICDE, 2015, pp. 42–53.

[6] W. Su, J. Wang, and F. Lochovsky, “Record matching over query results from multiple web databases,” TKDE, vol. 22, no. 4, 2010.

[7] H. Kopcke and E. Rahm, "Frameworks for entity matching: A ¨ comparison," DKE, vol. 69, no. 2, pp. 197–210, 2010.

[8] X. Yin, J. Han, and S. Y. Philip, "Truth discovery with multiple conflicting information providers on the web," ICDE, 2008.

[9] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," TKDE, vol. 19, no. 1, pp. 1–16, 2007.

[10] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," TKDE, vol. 24, no. 9, 2012.

[11] S. Tejada, C. A. Knoblock, and S. Minton, "Learning object identification rules for information integration," Inf. Sys., vol. 26, no. 8, pp. 607–633, 2001.

[12] L. Shu, A. Chen, M. Xiong, and W. Meng, "Efficient spectral neighborhood blocking for entity resolution," in ICDE, 2011.

[13] Y. Jiang, C. Lin, W. Meng, C. Yu, A. M. Cohen, and N. R. Smalheiser, "Rule-based deduplication of article records from bibliographic databases," Database, vol. 2014, 2014.

[14] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava, "Truth finding on the deep web: Is the problem solved?" in PVLDB, vol. 6, no. 2, 2012, pp. 97–108.

[15] J. Pasternack and D. Roth, "Making better informed trust decisions with generalized fact-finding," in IJCAI, 2011, pp. 2324–2329.

[16] X. L. Dong and F. Naumann, "Data fusion: resolving data conflicts for integration," PVLDB, vol. 2, no. 2, pp. 1654–1655, 2009.

[17] E. K. Rezig, E. C. Dragut, M. Ouzzani, A. K. Elmagarmid, and W. G. Aref, "ORLF: A flexible framework for online record linkage and fusion," in ICDE, 2016, pp. 1378–1381.

[18] X. Wang, X. L. Dong, and A. Meliou, "Data x-ray: A diagnostic tool for data errors," in SIGMOD, 2015, pp. 1231–1245.

[19] G. R. D. Patrick AV Hall, "Approximate string matching," ACM Computing Surveys, vol. 12, no. 4, pp. 381–402, 1980.

[20] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string metrics for matching names and records," in KDD workshop on data cleaning and object consolidation, 2003, pp. 73–78.