e-ISSN: 2319-8753 | p-ISSN: 2347-6710

IJIRSET

International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Issue 2, February 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.118

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |

Web Based Join Framework for Data Skewness in Character Based String Similarity

Mr.C.A.Kandasamy, Haridharan K

Assistant Professor, Department of Computer Applications (MCA), K.S.R. College of Engineering (Autonomous), Tiruchengode, India

Department of Computer Applications (MCA), K.S.R. College of Engineering (Autonomous), Tiruchengode, India

ABSTRACT-Dictionary-based entity extraction identifies predefined entities from a document. A recent trend for improving extraction recall is to support approximate entity extraction, which finds all substrings in the document that approximately match entities in a given dictionary but this causes redundancy and lower its performance. To improve the performance of string matching from a document a technique called Approximate Membership Localization is used. This technique aims at locating non overlapped substring which eliminates redundancy and improves performance, efficiency of searching process. It takes more time to find the best match, since it has huge number of possible combinations. To reduce the searching time and to make the process more effective and efficient genetic algorithm can be used.

KEYWORDS: Approximate Membership Localization (AML), Approximate Membership Extraction (AME).

I. INTRODUCTION

Data mining generally categories into various function such as classification, clustering, searching. Classification in data mining is termed as collection of target categories/classes. Classification is mainly used for portioning the data in to different classes. Generally the term classification referred as the process of generalizing the data according to different instances.

The main aim is to predict the target class accurately. Classification predicts the target class accurately .classification are discrete and it doesn't have any order. The classification algorithm finds relationships between the values of the predictors and the values of the target. Different algorithm uses different technique to find these relationships.

After classifying the data the datasets are need to be grouped in to a single domain. The process of grouping data is called as clustering. The data objects that are similar will be in a same group and those are dissimilar will be in a different group.

Cluster analysis is not an automatic task but it is an iterative process. There are various classification models which uses different types of clustering models which uses different types of clustering methods. For example connectivity model uses hierarchical clustering which is mainly based on distance. There are various types of clustering methods such as

1. Strict portioning clusters
2. Strict portioning clusters with outliers
3. Overlapping clustering
4. Hierarchical clustering
5. Subspace clustering

The data mining involves various common tasks such as anomaly detection, association rule learning, clustering, classification, regression and summarization. Given a set of words of length up to n.The set of words are classified and clustered .The data can be retrieved by means of a searching process. By using the search process the best match for the given input string can be find.

The approximate membership Extraction (AME) is a dictionary based entity search process. It takes more searching time and it causes many redundancies. To overcome this problem approximate membership localization is proposed.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |

II. LITERATURE SURVEY

Agarwal et al. suggested a technique that use a combination of pre-processing and web search engine adaptations in order to implement entity search functionality at very low space and time overhead. The main tasks is to identify relevant information in a structured database using a web search query very efficiently and effectively. The search in each structured database is "soiled" in that it exclusively uses the information in the specific structured database to find matching entities. That is it matches the query terms only against the information in its own database. The result from the structured database search is therefore independent of the result from web search. The major drawback is that it takes a high processing time to search from a structured database.

Arasu et al. suggested a similarity join operation for reconciling representation of an entity. Set similarity join algorithm define that given two input collections of sets identify all pairs of set, one from each collection that are highly similar. A data collection often has various inconsistencies which have to be fixed before the data can be used for accurate data analysis. The notion of similarity is captured numerically using a string based similarity. Apart from string based similarity semantic relationship between entities can be exploited to identify different representation of the same entity. The algorithm is characterized as signature based algorithms that first generate signature for input sets, then find all pairs of sets whose signature overlap, and finally output the subset of these candidate pairs that satisfy the set-similarity predicate. The major drawback is that it just compare with minimum amount of database so that it does not give exact similarity.

Karach et al. describes an algorithm to solve the approximate dictionary matching problem. Given a list of words w, maximum distance d, fixed at pre-processing time and a query word q to retrieve all words from w that can be transformed into q with d or less edit operations. Each word is represented by a string of characters over a finite alphabet ∑. The Levenshtein distance ed (a,b) defines a metric between two words a,b and is used to compute distance between two words. The most trivial algorithm to solve the problem is scanning sequentially through the input list and noting the best match at each entry. The major drawback is that this distance computations are expensive and takes more time so processing is low.

Chan et al. describes the problem of indexing a text to support searching substrings that match a given pattern with at most errors. A naive solution either has a worst case matching time complexity or requires space. Devising a solution with better performance has been a challenge for calculating the space index that can support error matching in respect to time where occ is the number of occurrences. The major concern is how to archieve efficient matching without large amount of space for indexing, one can improve the matching time by including all possible erroneous substrings yet this seems to require o(nk)space.They are able to avoid brute force matching of patterns with a moderate increase in the index size. The major drawback is that it take long time and space complexity is high.

Chaudhuri et al. describes about the entity matching task identifies entity pairs one from a reference entity table and other from an external entity list. The task is to check whether or not a candidate string matches with member of reference table. However the challenge is that it is quite hard to obtain a large number of documents containing string unless large portion of the web is crawled and indexed as done by search engines. The approach is used to compute string similarity score between the candidate and the reference strings. The major drawback is that the quality of the id token set is low.

III. EXISTING SYSTEM

The Approximate membership Extraction (AME) is a dictionary based entity search process.AME aims at identifying all substrings approximately matching any reference. The main objective of AME guarantees a full coverage of all true matched substrings within the document. But it generates many redundant matched substrings and it also lower efficiency and accuracy. The approximation is usually con-strained by a similarity function (such as edit distance, jaccard, cosine similarity, etc.) and a threshold within [0, 1], such that slight mismatches are allowed between the sub-string and its corresponding dictionary reference. For instance, given a list of conference names like "ACM SIGMOD Conference," "VLDB Conference," "IEEE ICDE Conference" as shown on the left part of Fig. 1, the task is to find matches from the text on the right, such as "VLDB 2010 Conference" and "ICDE Conference," although they do not match the string "VLDB Conference" and "IEEE ICDE Conference" in the dictionary exactly.

The dictionary-based approximate membership checking process is now expressed by the Approximate Membership Extraction (AME), finding all substrings in a given docu-ment that can approximately match any clean references. The objective of AME guarantees a full coverage of all the true matched substrings within the document, where the true matched substring is a true mention of the clean reference semantically. On the other hand, it generates many redundant matched substrings, thus rendering AME un-suitable for real-world tasks based on entity extraction. Indeed, redundant pairs are qualified to be part of AME results, but are unlikely to be true matches in real-world

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |

situations. A pilot study with a conference dictionary and a large set of relevant web documents shows that given a reasonable similarity threshold such as 0.85, about 90 percent of substrings and reference pairs found in AME are not true matched pairs. Such a large proportion is not only expected to involve a lower efficiency of the entity extraction process, but also to deteriorate the accuracy of specific real-world application which greatly relies on the accuracy of the extracted matched pairs.

LIMITATIONS

The major limitations of AME are that causes redundancy and lower the performance efficiency. For example if there exists a dataset contains various names such as{abi,asha,rani,ram,anuskha,ramanathan...}.If the input string is "ram". The AME retrieve all the substring that match that input string. It will retrieve names such as {ram, ramkumar, anusharam, ramanathan}.The AME does not match the exact data it does not match the exact data it does not suitable for the real world entities.

These redundant substring and reference pairs (referred to as redundancies in the rest of the paper) fall into one of the two following categories.

Boundary Redundancy: In a document d ¼ fw$_{0:}$. . . . . :
w$_n$g, if the substring m ¼ fw$_i$; . . . ; w$_j$g matches with an entity r, there exist _ ¼6 0 such that m^0 ¼ fw$_{iþ_}$; . . . ; w$_j$g or m^{00} ¼ fw$_i$; . . . ; w$_{jþ_}$g also match r. However, only one of these overlapped strings is enough to characterize the presence of r in the document. For example, if d ¼ "... published in acm sigmod conference in 2009" and r ¼ "acm sigmod conference," the AME may find not only "acm sigmodconference," but also other substrings such as "in acm sigmod conference," "acm sigmod," "sigmod con-ference" and "sigmod conference in" matching with r.

Multiple-match Redundancy: In a document d ¼
fw$_0$; w$_1$; . . . ; w$_n$g, if the substring m$_1$ ¼ fw$_i$; . . . ; w$_j$g matches an entity r$_1$ there is a substring m$_2$ ¼ fw$_k$; . . . ; w$_l$g that matches an entity r$_2$ such that k _ j, and j < l. However, if m$_1$ and m$_2$ overlap with each other, there should be only one correct matched pair between ðm$_1$; r$_1$Þ and ðm$_2$; r$_2$Þ. For example, if d ¼ ". . . submitted to vldb journal..." and r$_1$ ¼ "vldb conference," r$_2$ ¼ "vldb journal," the AME may find substring "vldb" matched with r$_1$, as well as "vldb journal" matched with r$_2$.

However, to remove these redundancies from AME results we need to generate and identify the unqualified redundant matches beforehand. In order to efficiently solve the AML problem, we propose an optimized algorithm (P-Prune) which prunes a large part of overlapped redundant matched substrings before generating them (see Section 4).

The main contributions of this paper are summarized below:

1. We formalize the AML problem, which aims at locating nonoverlapped clean references approxi-mately mentioned in a given document. Through identifying and removing redundancies, the matched pair results of the AML are much closer to the true matched pairs than AME results. AML is more appropriate than AME in real-world applica-tions where the accuracy of matched pairs are greatly relied on.

2. A typical real-world application of the AML, the web-based approximate join framework, is proposed. This framework can effectively solve a class of problems where sufficient join attributes are only available in web documents. It can be applied in many situations, such as joining a (product, company) dirty table to some company list to find products that are produced by each company in the list, or joining a (book, writer, price) sale record table which contains sale records from all the book stores of a country, to a writer reference list to find out whose books are the most popular ones. The advantage of AML over AME will be demonstrated on the experimental results obtained within this framework.

In this paper, we propose the web-based join, which is an approximate join using web documents collected by a web search engine. Another search-based method has been previously proposed by Chaudhuri et al. [13] where the original reference list is extended with IDTokenSets of every entity string with the help of a web search engine. Each IDTokenSet (Identifying Token Set) of an entity string is a subset of tokens present in the string, which is enough to identify the entity. For example, given s ¼ "ACM Con-ference on Information and Knowledge Management," a subset of its tokens is set$_1$ ¼ {Information, and, Knowledge, Management}. If we search the concatenation of all tokens from set$_1$ in a web search engine, and the other tokens "ACM," "Conference," and "on" appear frequently and

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |
|| Volume 12, Issue 2, February 2023 ||
| DOI:10.15680/IJIRSET.2023.1202039 |

closely in retrieved webpages, then we say that set_1 is an IDTokenSet of s. In this way, the approximate entity matching problem against the original reference table is reduced to an exact entity matching problem against the extended reference table. This approach is an extension of token-based similarity metric, and cannot step across the textual-semantic gap. There are other synonyms which do not consist of IDTokenSets such as abbreviations and typos that cannot be found.

IV. PROPOSED SYSTEM

In this section, we present the web-based join framework. As opposed to the traditional approximate join, web-based join targets a different scenario: given a list of elements T with an attribute T:X and a clean reference list R with an attribute R:A, the problem is to create from the web an intermediary table RT containing valued correlations between two attributes T:X and R:A in order to perform a join between T and R.

Given that the information available on the web can be dirty and noisy, RT shall contain the likelihood associated with its entries. Based on the hypothesis that there exist web documents containing elements of T:X that also contain the elements of R:A, we use the elements of T:X as a query for a search engine to retrieve the ranked list of documents Docs. The framework and notations are presented in Fig. 2.

For example, we input a paper title of KDD conference into the search engine and crawl the returned webpages. To find the linked conference name for the paper title, two key challenges need to be solved. First, given that the mention of conference names in webpages might not be exact, we need to locate clean references approximately mentioned in given documents (AML problem). For this challenge, our proposed method for AML is presented in Section 4. Second, besides KDD, some other conference names like SIGMOD, ICDE might also occur in the webpages. To link KDD with the paper title, one way is to use patterns of the association between paper title and its conference, which need to be learned with enough training data set. The other way is to score the correlations between elements of T:X and references in R according to the results of AML. In our approach, we prefer the second solution with an unsuper-vised approach given in the subsection below.

SCORING CALCULATION:

The values of the links to clean references of R in Docs are measured with an unsupervised approach inspired from [13]. This approach provides a score that can be used by setting a threshold (either for the value or for the number of best matched clean references) to perform the join. For a given value T:x of T:X, the three relevant parameters of the evaluation of correlations are for each document:

1. **Frequency freq:** the number of times each reference is mentioned in each document of Docs.
2. **Distance dist:** the distance between the mention of each clean reference and the position of T:x.
3. **Document importance imp(d)**: documents retrieved on the web are of different importance w.r.t. their relevance to the query, i.e., their ranks in a web search engine results.

This scoring approach requires us to determine the number of times and the locations where a clean reference is mentioned in Docs. Given that these references may be approximately mentioned in the documents, we need to find nonoverlapped substrings that can approximately match any clean reference in documents. When several reference-matched substrings overlapped in a word posi-tion of a document, only one of them should remain: the one with the largest similarity to its matched entity. For ease of presentation, we call this matched substring a bestmatch substring in this paper.

ALGORITHM FOR APPROXIMATE MEMBERSHIP LOCALIZATION:

In this section, we provide algorithms for the AML problem based on the two assumptions below:

- **Assumption 1:** any approximate mention m that matched with a reference consists of consecutive words in a document, i.e., each m is a substring.
- **Assumption 2:** only substrings whose length is up to a length threshold L are of interest, so we may as well require that |m|< L.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |

In this section, we introduce two algorithms to solve the AML problem. The first algorithm is based on AME. Since the AML results are a subset of the AME results, we can do AME first, then remove the redundant pairs from the AME results second. However, this algorithm uses a lot of extra time for generating these redundant pairs and then removing them. As an alternative, a more efficient approach is to prune the potential redundant substrings before generating them, so that less time is required for generating and verifying the remaining candidate pairs.

Method 1: AME-Based AML

1. Since AME results are a superset of AML results, a straightforward way to solve AML is to remove the redundant pairs a posteriori from the large set of AME results.
2. The two corresponding constraints are defined below. The two constraints will be used as fundamental rules in pruning redundancies. The first boundary constraint was proposed in [9], while the second one is straightforward.

3. Constraint 1 (Boundary Constraint). Assume substring m ¼ $w_i w_{i\text{þ}1}$ . . . w_j is a match of entity r in the document. The boundary constraint requires that the first and last tokens of m, i.e., w_i and w_j, should be present in entity r.

4. Constraint 2 (Nonoverlapped Constraint). Assume m_1 and m_2 are two substrings from M, where m_1 matches entity r_a, m_2 matches entity r_b (r_a can be the same as r_b). If the positions of m_1 and m_2 overlap with each other in M, then only one of them should be a true match result. If simðm_1; r_aÞ > simðm_2; r_bÞ, then we remove the ðm_2; r_bÞ pair.

5. With the two constraints above, we now consider how to remove redundancy from the AME results. A straightfor-ward approach is to apply the Nonoverlapped Constraint after the AME results are generated. However, this methods implies the unnecessary generation and verification of a large part of the boundary redundancies.

6. A better alternative is to implement an algorithm based on the popular filtration-verification framework of AME. This AME-based algorithm for AML can be briefly described as:

In the filtration step, we apply the Boundary Constraint to prevent the generation of boundary redundancies.

In the verification step, we apply the Nonover-lapped Constraint to remove all the overlapped redundancies.

Method 2: Potential Redundancy Prune

The AME-based method for AML uses time resources for generating and identifying unqualified redundant matches. In order to efficiently solve AML, we propose an optimized algorithm P-Prune, which can prune potential redundant substrings before generating them. This algorithm shows a much higher efficiency than the AME-based algorithm, as will be demonstrated in the experimental study.

General idea of P-Prune. For an input document M, AML only requires best match substrings. Assuming we can divide M into subdocuments, where each subdocument is a consecutive substring of M and subdocuments may overlap with each other such that all best match substrings of M are located within these subdocuments, the problem of finding all best match substrings from M becomes a problem of finding the best match substrings from these subdocuments of M. Furthermore, if there is at most one best match substring in a subdocument, it becomes faster to judge whether there is a best match substring in each sub document.

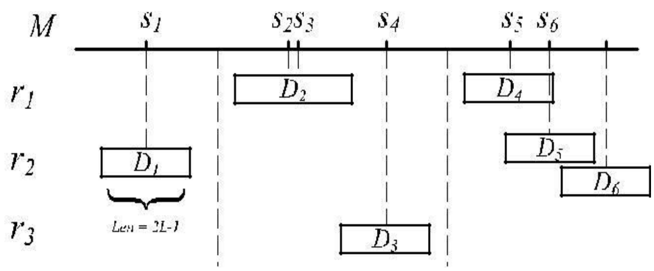
Generating Domains from Document:

To generate domains from M, all reference entities have to be considered. Here, we leverage the basic prefix signature scheme [9] to find the prefix signature set of each entity. If a substring m from M matches with an entity r, it should contain at least one word from the prefix signature set of r.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |



We first consider how to ensure all matched substrings of one reference entity r can be found within domains. Since each matched substring of r must contain at least one strong word w_s of r. according to the Assumption 2, the matched substring that contain w_s should be within a ð2L _ 1Þ length subdocument with w_s located at its centre. This subdocument is actually a domain of r. Therefore, for each strong word of r that appear in the document, we get a ð2L _ 1Þ length domain with the strong word located at its centre. In this way, we can ensure all matched substrings of r are covered by these domains.

For multiple reference entities, we find strong words of each entity, then generate the ð2L _ 1Þ length domain for each strong word in the given document. Two special cases in generating these domains are listed below:

1. If a strong word is strong in K (K is a number) different entities, then K number of ð2L _ 1Þ length domains are generated, all of which are coincident with each other;

2. When _$_{min}$ðrÞ > 1, if there are X (X > 1) strong words of entity r located consecutively (as long as they are not the same word), only one ð2L _ XÞ length domain has to be generated with the X strong words located at the center.

For simplicity, we call these (2L _ 1) (including (2L _ X)) length domains as window domains. All window domains in a document can be classified into two categories: nonoverlapped domains and over-lapped domais.

Generating Best matches from Domains

In a domain D of entity r, each matched substring of r is a candidate bestmatch substring of the domain, with at most one of them being the bestmatch substring. In this section, we first introduce how to generate matched substrings from each domain, then we consider how to find bestmatch from the matched substrings in nonoverlapped domains and overlapped domains, respectively.

Given a domain D of entity r, a baseline approach to generate matched substrings is to generate all possible substrings from the domain first, and then compute the similarity between each substring and entity string r. Here, we propose a smarter algorithm, which can prevent from generating some unnecessary substrings that are impossible to match with r. Our algorithm is based on the following definitions and constraints.

The domain D is divided into several consecutive partitions (partitions are not overlapped with each other): segments or intervals, as defined below:

Definition 4 (Segment and interval). In a domain D of r, the consecutive words (without divided symbols between them) which are all present in r compose a segment (of r) in D; the consecutive words which are all absent from r compose an interval (of r) in D. The segment that contains the strong word of D is the strong segment (of r) in D.

Based on the definition of bestmatch substring, we have the Segment Indivisible constraint below:

Constraint 3 (Segment indivisible constraint). In a domain of r, each segment cannot be divided when matching with r, i.e., each segment should either be contained in a match substring, or nonoverlapped with that match substring. According to this indivisible segment constraint and the boundary constraint, we generate substrings with segments and intervals instead of single words.

Candidate match generation. Given a domain D of r divided into several segments and intervals, with seg_s referring to the strong segment. Intuitively, any possible match substring of r should contain seg_s, such that we only need to consider substrings containing seg_s. The detailed algorithm is described below, where cur is the substring we are processing, Active is the active substrings set , and CandSet is the candidate match substring set:

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

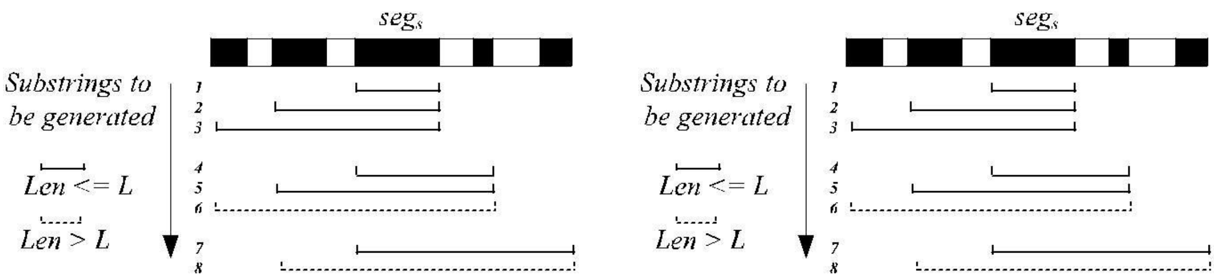
|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |

1. Each time the adjacent segment on the left of cur and the interval between them are added to cur, this new cur is put into Active. If sim(Cur; r) >, we also output cur as a candidate substring. This step repeats iteratively until there is no new segment on the left of cur.
2. We get the unprocessed segment closest to seg_s on the right. For each substring cur in Active, we refresh it by adding the new segment (and the interval adjacent to it on the left) to cur. If simðCur; rÞ > _, we output cur into CandSet. If jcurj > L, then we should remove cur from Active. This step also repeats iteratively until there is no more new segment on the right side of seg_s.

Best Match localization. Based on the candidate match generation algorithm above, now we consider how to locate best matches from nonoverlapped domains and overlapped domains, respectively. For several overlapped domains, we have to generate all match substrings for each domain, and sort them according to their similarity to the corresponding entities. The bestmatch substrings are then obtained from them by implementing the nonoverlapped constraint, that is, for overlapped candidates, only the one with the largest similarity to its matching reference will remain.

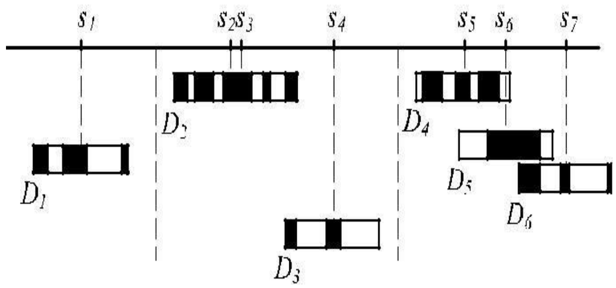


For a nonoverlapped domain, since it has no overlap with other domains, we only need to judge bestmatch within the domain. Let str be a matched substring of r in a nonoverlapped domain, if there is no other matched sub-strings that overlapped with str within the domain, then str will be a bestmatch substring of r. Otherwise, we assume the overlapped one is str^0. If simðstr; rÞ > simðstr^0; rÞ, then str can still be the bestmatch substring of r, or else str^0 will be the bestmatch substring of r. Therefore, we have

Lemma 1. In a nonoverlapped domain of r, once we have a matched substring of r, there must be a bestmatch substring of r in this domain.

According to Lemma 1, once we find a substring with a similarity to r larger we can infer that there is a bestmatch substring of r in this domain We output the position of the strong segment in this domain as the position of the bestmatch string.

Pruning Window Domains



Domain windows before prune

The window domains we generated in Section 4.2.1 are raw divisions which only guarantee that all bestmatch substrings can be found within. For very large reference lists, the number of these window domains, especially overlapped ones, can become an issue. In this section, we introduce how to prune domains that impossible to contain a bestmatch substring, and how to minimize the size of the remaining domains.

Prune 1 (Weight Pruning): A domain D of r should be removed, if the sum weight of all segments in D is smaller than wt(r).

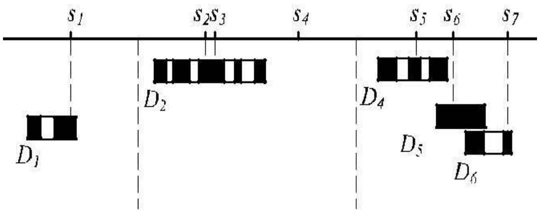
|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

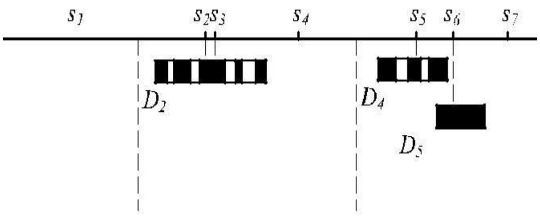
| DOI:10.15680/IJIRSET.2023.1202039 |

Prune 2 (Interval Pruning): In a domain of r, if there is an interval t whose weight is larger than 1– – _ wtðrÞ on the left (right) side of the strong segment, then this interval and other segments and intervals on the left (right) side of t should be removed from the domain.

Prune 3 (Boundary Pruning): The leftmost and rightmost partitions of a domain of r should be two segments of r.



(b). Temporal Domains being Pruned



(c). Final Domains after Prune

V. EXPERIMENTAL STUDY

Webpages versus Snippets

The web-based join framework requires to retrieve web documents. Intuitively a web document is the webpage returned by a web search engine. Another type of light-weight web document is the snippet, which is a short relevant

To measure the effectiveness of using snippets instead of original webpage, we compare the experimental results of using the first 100 WebPages and the first 100 snippets retrieved by Google's web search engine (Google returning at most 100 snippets at once for a query) for 100 records randomly selected from the 6k labeled publication records. The results presented in Fig. 7a evidence that using snippets always yields better performances than using web pages

AML versus AME

In order to demonstrate the appeal for AML versus AME in our search-based method, we compare the join precision and recall of the search-based method with using AME results or AML results as the locations of the references in the retrieved documents, respectively.

For doing AME, we use the EvIter algorithm based on an SIL index [31], which has proven to be state of the art in terms of efficiency. Best performance is reached for a substring length threshold set at L ¼ 10 and the "compression rate" parameter of EvIter at k ¼ 3.

The results presented by Fig. 7b show that WebJoinþ AME cannot perform as well as either WebJoinþAML (AME-based) or WebJoinþAML(P-Prune). The best preci-sion and recall of web-based join with the AML results can be as high as 0.873 and 0.831, respectively, which is much better than those with the AME results (0.5 and 0.8, respectively). Although WebJoinþAME may reach a higher recall than WebJoinþAML (AME-based) and WebJoinþ AML (P-Prune) (since the AME results are superset of AML results), plenty of redundancy within the AME results will greatly deteriorate the join precision.

In the meantime, we also observe that the join perfor-mance of WebJoinþAML (P-Prune) is no worse than WebJoinþAML (AME-based), which proves that the approx-imation brought by P-Prune has little effect on the results.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

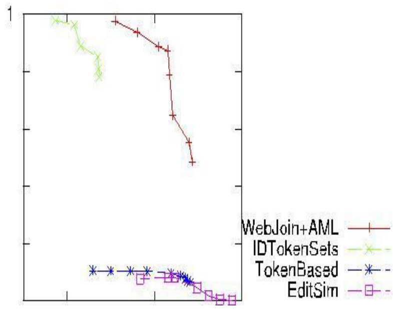
| DOI:10.15680/IJIRSET.2023.1202039 |

However, the P-Prune
algorithm is much more efficient than the AME-based algorithm

Web-Based Join versus Other Join Methods

Now, we compare the join results of WebJoinþAML with several join methods below:

1. IDTokenSets: This is the search-based method proposed by Chaudhuri et al. in [1],[13], which expands the given reference list with IDTokenSets of each of its entities.
2. Token-based: Token-based similarity metric usually takes a string as a token set (words or n-grams), then calculate similarity between two token sets. Here, we use the idf metric to assign weights to words, then calculate the cosine-similarity between word sets as the similarity between two strings.
3. Edit-distance: This is a state-of-the-art similarity metric, which measures the similarity between strings according to the minimum number of edits needed to transform one string into the other. Here, we compare to the Monge-Elkan version of edit-distance similarity metric [16], [37] that can weaken the influence of word gaps in the strings, thus favoring the retrieval of acronyms.



Comparison between three methods

Here, the P-Prune algorithm is used for the AML component and reaches the best performance when the substring length threshold is set to L ¼ 10 and the similarity threshold to _ ¼ 0:85. In the link evaluation process, we set B ¼ 10 and reach the best results with w_a ¼ 0:7. The IDTokenSets method reaches its best performance with _ ¼ 0:85. Snippets of the first 100 search results retrieved by Google's web search engine are used for both web-based Join and IDTokenSets methods.

Since all four methods use a threshold to determine qualified matched pairs as the matching results, they inevitably produce both false-positives and false-negatives. Based on the gold standard, we measure the precision and recall of the entity matching results of the four methods for various thresholds. The results of this evaluation , demonstrate that our web-based join method reaches its highest recall and precision (recall ¼ 0:831, precision ¼ 0:873), which is far above all other methods.

VI. CONCLUSION

Formalizing the AML problem and propose to solve it with an efficient P-Prune algorithm. Prune is proved to be several times faster, sometimes even tens or hundreds of times faster, than simply adapting formerly existing AME methods. To inspect the improvement of AML over AME here apply both approaches within our proposed web-based join framework, which is a typical real-world application that greatly relies on the results of membership checking. The results prove that the precision and recall of web-based join with the AML results can be as good as 0.873 and 0.831, respectively, largely outperforming AME (where results are 0.5 and 0.8, respectively). Also apply the web-based join framework in joining publication titles with venue names from the ERA conference and journal list, thus demonstrating that our method can reach a higher precision and recall than the previous search-based one proposed and previous textual-based similarity metrics that use a unique join attribute. AML-targeted solutions are more appropriate than the AME-targeted solutions for this type of real-world applications, since the matched pair results of the AML are much closer to the true matched pairs than AME results.

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |

REFERENCES

[1] S. Agrawal, K. Chakrabarti, S. Chaudhuri, V. Ganti, A. Konig, and D. Xin, "Exploiting Web Search Engines to Search Structured Databases," Proc. 18th WWW Int'l Conf. World Wide Web, pp. 501-510, 2009.

[2] A. Aho and M. Corasick, "Efficient String Matching: an Aid to Bibliographic Search," Comm. ACM, vol. 18, no. 6, pp. 333-340, 1975.

[3] A. Arasu, V. Ganti, and R. Kaushik, "Efficient Exact Set-Similarity Joins," Proc. 32nd VLDB Int'l Conf. Very Large Data Bases, pp. 918-929, 2006.

[4] R. Bayardo, Y. Ma, and R. Srikant, "Scaling Up All Pairs Similarity Search," Proc. 16th WWW Int'l Conf. World Wide Web, pp. 131-140, 2007.

[5] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allow-able Errors," Comm. ACM, vol. 13, no. 7, pp. 422-426, 1970.

[6] B. Bocek, E. Hunt, and B. Stiller, "Fast Similarity Search in Large Dictionaries," Technical Report ifi-2007.02, Dept. of Informatics Univ. of Zurich, 2007.

[7] A. Borthwick, "A Maximum Entropy Approach to Named Entity Recognition," PhD thesis, New York Univ., 1999.

[8] H. Chan, T. Lam, W. Sung, S. Tam, and S. Wong, "A Linear Size Index for Approximate Pattern Matching," Proc. 17th Ann. Symp. Combinatorial Pattern Matching, pp. 49-59, 2006.

[11] S. Chaudhuri, V. Ganti, and R. Kaushik, "A Primitive Operator for Similarity Joins in Data Cleaning," Proc. 22nd Int'l Conf. Data Eng.,
p. 5, 2006.

[12] S. Chaudhuri, V. Ganti, and D. Xin, "Exploiting Web Search to Generate Synonyms for Entities," Proc. 18th Int'l Conf. World Wide Web (WWW), pp. 151-160, 2009.

[13] H. Chieu and H. Ng, "Named Entity Recognition: A Maximum Entropy Approach Using Global Information," Proc. 19th Int'l Conf. Computational Linguistics, p. 7, 2002.

[14] W. Cohen, "Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity,"Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 201-212, 1998.

[15] W. Cohen, P. Ravikumar, and S. Fienberg, "A Comparison of String Distance Metrics for Name-Matching Tasks," Proc. IJCAI '03 Workshop Information Integration on the Web (IIWeb '03), pp. 9-10, 2003.

[16] W. Cohen and S. Sarawagi, "Exploiting Dictionaries in Named Entity Extraction: Combining Semi-Markov Extraction Processes and Data Integration Methods," Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 89-98, 2004.

[17] I. Dagan, S. Marcus, and S. Markovitch, "Contextual word Similarity and Estimation from Sparse Data," Proc. 31st Ann. Meeting on Assoc. for Computational Linguistics, pp. 164-171, 1993.

[18] A. Elmagarmid, P. Ipeirotis, and V. Verykios, "Duplicate Record Detection: A Survey," IEEE Trans. Knowledge and Data Eng., vol. 19, no. 1, pp. 1-16, Jan. 2007.

[19] L. Getoor and C. Diehl, "Link Mining: A Survey," ACM SIGKDD Explorations Newsletter, vol. 7, no. 2, pp. 3-12, 2005.

[20] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," Proc. 25th VLDB Int'l Conf. Very Large Data Bases, pp. 518-529, 1999.

[21] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrish-nan, and D. Srivastava, "Approximate String Joins in a Database (Almost) for Free," Proc. 27th VLDB Int'l Conf. Very Large Data Bases, pp. 491-500, 2001.

[22] D. Gusfield, Algorithms on Strings Trees and Sequences: Computer Science and Computational Biology. Cambridge Univ. Press, 1997.

[23] W. Hon, T. Lam, R. Shah, S. Tam, and J. Vitter, "Cache-Oblivious Index for Approximate String Matching," Theoretical Computer Science, vol. 412, pp. 3579-3588, 2011.

[24] M. Jaro, "Probabilistic Linkage of Large Public Health Data Files," Statistics in Medicine, vol. 14, pp. 491-491, 1995.

[25] K. Jarvelin and J. Kekalainen, "Cumulated Gain-Based Evaluation of IR Techniques," ACM Trans. Information Systems, vol. 20, no. 4,
pp. 422-446, 2002.

[27] D. Karch, D. Luxen, and P. Sanders, "Improved Fast Similarity Search in Dictionaries," Proc. 17th Int'l Conf. String Processing and Information Retrieval, pp. 173-178, 2010.

[28] N. Koudas, S. Sarawagi, and D. Srivastava, "Record linkage: Similarity Measures and Algorithms," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 802-803, 2006.

[29] Z. Li, L. Sitbon, L. Wang, X. Zhou, and X. Du, "Approximate Membership Localization (AML) for Web-Based

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.118| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 2, February 2023 ||

| DOI:10.15680/IJIRSET.2023.1202039 |

Join," Proc. 19th CIKM Int'l Conf. Information and Knowledge Management, 2010.

[30] D. Lin, "Automatic Retrieval and Clustering of Similar Words," Proc. Ann. Meeting Assoc. for Computation Linguistics, vol. 36, pp. 768-774, 1998.

[31] J. Lu, J. Han, and X. Meng, "Efficient Algorithms for Approximate Member Extraction Using Signature-Based Inverted Lists," Proc. 18th CIKM ACM Conf. Information and Knowledge Management,

[36] 315-324, 2009. A. Mikheev, M. Moens, and C. Grover, "Named Entity Recogni-tion without Gazetteers," Proc. Ninth Conf. European Chapter of the Assoc. for Computational Linguistics, pp. 1-8, 1999.

[37] A. Monge and C. Elkan, "The Field Matching Problem: Algo-rithms and Applications," Proc. Second Int'l Conf. Knowledge Discovery and Data Mining, pp. 267-270, 1996.

[38] G. Navarro, "A Guided Tour to Approximate String Matching," ACM Computing Surveys, vol. 33, no. 1, pp. 31-88, 2001.

.