International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
SJIF (2024): 6.623

Fuzzy Join Algorithms in Big Data

Cong Hao Nguyen

Hue University, 3 Le Loi, Hue city, Vietnam
Email: *nchao[at]hueuni.edu.vn*

Abstract: ***Fuzzy join is a more sophisticatedly approach to data matching and has many applications in many filelds. Instead of marking records as matches or mismatches based on exact matching algorithms, fuzzy joins are used to combine two data sets that do not have exact matching keys, but are similar up to a certain threshold. The biggest challenge is finding pairs of data with similarity greater than or equal to a given threshold within a given period of time. The paper presents several algorithms with Hamming, Levenshtein, Cosine distance measures applied to the MapReduce model and propose a new algorithm based on the hedge algebra for fuzzy join on semantically ordered fuzzy datasets in big data.***

Keywords: Fuzzy join , similarity algorithms, hedge algebra, big data, MapReduce

1. Introduction

Nowadays, with the rapid increase in the quantity and diversity of data based on the platform of global digital information convergence such as e-commerce websites, social networks such as Facebook, Tiktok, Instagram ..., businesses increasingly desire more valuable values from those types of data. That means authors and researchers must find ways to cope with the storage and processing of huge volumes of data and diverse types of data. Therefore, many scientists are researching technologies and algorithms to solve the problem of storing, processing and analyzing big data in the fastest way to meet the requirements of solving practical problems such as business, market analysis ....

In data querying, joins is very important role and are used to link data sets of relations together. However, join operations on large data are very complex and potentially costly. The join problem will be more difficult if the join attribute requirements have values that are not equal or differ by a certain threshold. In this case, we call it fuzzy join. Many results have studied fuzzy joins based on MapReduce [2][6][7][12][15], fuzzy matching using deep neural networks [17], but the above fuzzy join problem is based on values that seem completely different but are semantically equal. For example, using join a 25 years old employee and a young employee is still a problem that needs to be solved.
For example, in investigating, evaluating or selecting good students in Mathematics, there are 2 groups of experts based on a number of different criteria such as GPA over semesters, academic ranking, logical reasoning ability, achievements in mathematics exams, research ability, application of mathematics in programming and some common criteria such as Age, gender. Experts want to combine these two data sets. together to form the basis for evaluating the optimal selection of excellent Math students by connecting common attributes Age, Gender. But performing the usual connection to combine exactly is not suitable for the case This is because the criteria collected in both datasets do not apply to the same student and exact age matching does not make much sense. Therefore, fuzzy join will be used in this case to match same age as a certain measurement will be more reasonable.

In this paper, we propose a fuzzy join algorithm using hedge algebra. This approach, each fuzzy set (linguistic value) is considered as an element of the hedge algebra and the manipulation and calculation are performed directly on the language. The rest of the paper is organized as follows: in addition to the introduction, section 2 present background such as hedge algebra, MapReduce, fuzzy join. Section 3 present fuzzy join algorithms, section 4 is devoted for presenting fuzzy join algorithm using hedge algebra. Some conclusions and future research directions will be given in section 5.

2. Background

2.1 MapReduce

MapReduce is a programming framework for processing and computing large amounts of data in a distributed environment. It operates on two main phases: Map and Reduce. In the first phase, the Map phase will divide the data into key pairs (key, value), then the Reduce phase will group the key pairs (key, value) together to calculate the final result.

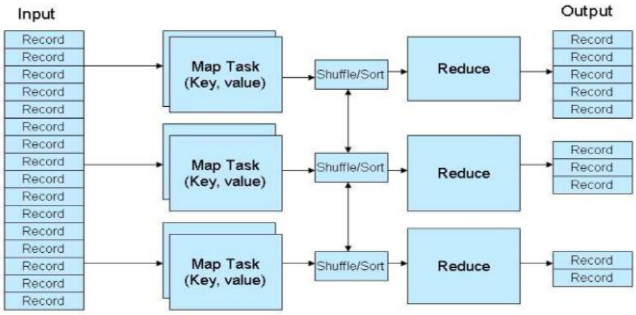

Figure 1: The Functions in the MapReduce programming model

MapReduce consists of two main functions: Map and Reduce. These are two functions defined by the user and are also two consecutive stages in the data processing process of MapReduce. These functions have the following specific main tasks:

Map function: This function is responsible for processing a key pair (key, value) to create a new key pair (keyl, valuel), at this time the key pair (keyl, valuel) will act as an intermediary. After that, the user only needs to write the data

to the hard disk and quickly notify the Reduce function so that the data goes into the Reduce input.

Reduce function: This function is responsible for receiving intermediate key pairs and values corresponding to that number of keys (keyl, valuel) to create a different set of keys by combining them. These key/value pairs will be fed into the Reduce function through a position pointer. This process will help programmers easily manage a large number of lists as well as allocate values suitable for system memory.

In addition, between Map and Reduce there is another intermediate step called Shuffle. After Map completes its task, Shuffle will continue to collect and synthesize the intermediate key/value pairs created by the previous Map and transfer it to Reduce for further processing.

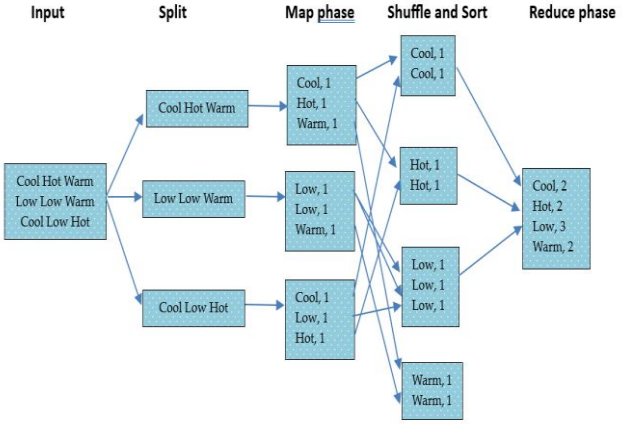

Figure 2: Overall MapReduce workflow

2.2. Fuzzy join

Fuzzy joins are a technique in data processing and data science that links tables together based on approximate similarities between values, rather than requiring exact matches as in a regular join. In a regular join (inner join, left join...), two tables are joined if the values in the key column are exactly the same. Fuzzy joins allow records to be joined based on the approximate similarity of strings, numbers, or other characteristics [4].

Given two input data sets R and L, the fuzzy join operation will return all the combined output records x ∈ R and y ∈ L such that Sim (x, y) > = θ, where Sim is the similarity function and θ is a user-specified threshold. The similarity functions are used such as include Hamming, Levenshtein, Longest common sequence (LCS), Jaccard, Jaro, Jaro – Winkler…. Given two input data sets R and L (as shown in Table 1), the fuzzy join operation with a similarity threshold of 1 provides the result set shown in Table 2.

Table 1: Dataset R and S

R		S	
Join key R	Value R	Join key S	Value S
CS1_H1	1	CS1_H1	S1
CS2_H1	2	CS2_H1	S2
CS1_H2	3	CS1_H2	S3
CS2_H3	4		

In Table 2, the result set consists of 5 records, of which the records in row 1, row 4, and row 5 are the results of the exact match operation. The remaining two records are the results of the fuzzy join with a threshold of 1. When performing the fuzzy join of two datasets based on the MapReduce model, the mapping phase will divide the input datasets of R and S for the mappers to process (Figure 3).

Table 2: Join results of R and S with a fuzzy threshold of 1

Join key R	Value R	Join key S	Value S
CS1_H1	1	CS1_H1	S1
CS2_H1	2	CS1_H1	S1
CS1_H1	1	CS2_H1	S2
CS2_H1	2	CS2_H1	S2
CS1_H2	3	CS1_H2	S3

For example, the datasets are divided into four Mappers, where Map1 is supposed to process the first two records of dataset R, Map2 processes the remaining two records of dataset R, Map3 processes two records of dataset S, and Map4 processes the remaining one record of dataset S. Then, the tuples with the same join key will be sorted and shuffled into the same reducer before performing the join operation, and the result is returned after performing the join at the Reducer.

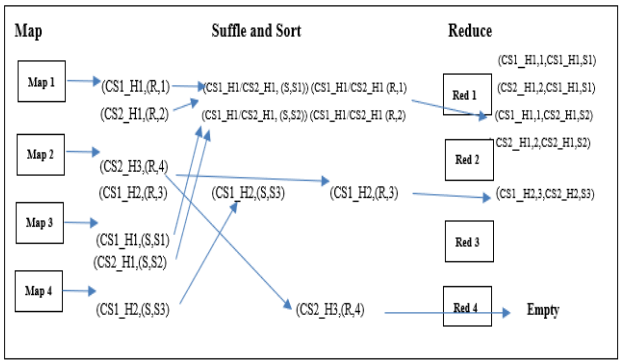

Figure 3: Fuzzy join based on MapReduce model

2.3. Hedge algebra

Hedge algebra is one approach to detecting algebraic structure of the value domain of the linguistic variable. In view of algebra, each value domain of the linguistic variable X can be interpreted as an algebra $AX = (X, G, H, \preceq)$, in which $Dom(X) = X$ is the terms domain of linguistic variable X is generated from a set of primary generators $G = \{c^-, c^+\}$ by the impact of the hedges $H = H^- \cup H^+$; W is a neutral element; ≤ is an semantically ordering relation on X, it is induced from the natural qualitative meaning of terms. Order structure induced directly so is the difference compared to other approaches. When we add some special elements, then hedge algebra become an abstract algebra $\underline{\boldsymbol{X}} = (\boldsymbol{X}, G, H, \Sigma, \Phi, \preceq)$, which Σ, Φ are two operators taking the limit of the set terms is generated when affected by the hedges in H. Alternatively, if the symbol $H(x) = \{h_1 \ldots h_p x / h_1, \ldots h_p \in H\}$, then $\Phi x = infimumH(x)$ and $\Sigma x = supremumH(x)$. Thus, hedge algebra $\underline{\boldsymbol{X}}$ is built on foundation of Hedge algebra $AX = (X, G, H, \preceq)$, where $X = H(G)$, Σ and Φ are two additional operators. Then X= X∪Lim(G) with Lim (G) is the set of elements limited: $\forall x \in Lim(G), \exists u \in X: x = \Phi u$ or $x = \Sigma u$. The limitation elements are added to hedge algebra $\underline{\boldsymbol{X}}$ to make the

new calculation meant and so $\underline{\boldsymbol{X}} = (\boldsymbol{X}, G, H, \Sigma, \Phi, \leq)$ called complete hedge algebra [11].

Definition 2.1. A function $f : \boldsymbol{X} \to [0; 1]$ is called a quantitative semantic function: if $\forall h , k \in H^+$ or $\forall h , k \in H^-$ and $\forall x , yX \in$, we have :

$$\left|\frac{f(hx) - f(x)}{f(kx) - f(x)}\right| = \left|\frac{f(hy) - f(y)}{f(ky) - f(y)}\right|$$

Definition 2.2. Given a quantitative semantic function f of $\boldsymbol{X}$. For any $x \in \boldsymbol{X}$, the fuzziness of x, denoted $I(x)$ and measured by the diameter of the set $f(H(x))$.

Definition 2.3. A function fm: $\boldsymbol{X} \to [0; 1]$ is called a fuzziness measure of $\boldsymbol{X}$ if it satisfies the following axioms:

i) fm is the complete fuzzy measure on $\boldsymbol{X}$, i.e. $\sum_{i=-q, i\neq 0}^{p} fm(h_i u) = fm(u)$ for every $u \in \boldsymbol{X}$.

ii) If x is a crisp value, $H(x) = \{x\}$ then $fm(x) = 0$. Hence, $fm(\boldsymbol{0}) = fm(W) = fm(\boldsymbol{1}) = 0$.

iii) For all $x , y \in \boldsymbol{X}$ and $h \in H$, we have $\frac{fm(hx)}{fm(x)} = \frac{fm(hy)}{fm(y)}$, this ratio does not depend on x and y , denoted by $\mu(h)$ is called the fuzziness measure *of* the hedge h.

Proposition 2.1. Given a fuzziness measure fm on **X** and fuzziness measure μ(h) of hedges, the following statements hold:

i) $fm(hx) = \mu(h) fm(x), \forall x \in \underline{\boldsymbol{X}}$

ii) $fm(c^-) + fm(c^+) = 1$

iii) $\sum_{-q\leq i\leq p, i\neq 0} fm(h_i c) = fm(c)$, *where* $c \in \{c^-, c^+\}$

iv) $\sum_{-q\leq i\leq p, i\neq 0} fm(h_i x) = fm(x)$, $x \in \underline{\boldsymbol{X}}$.

v) $\sum_{-q\leq i\leq -1} \mu(h_i) = \alpha$ *and* $\sum_{1\leq i\leq p} \mu(h_i) = \beta$, *where* $\alpha, \beta > 0$ *and* $\alpha + \beta = 1$.

Example 2.1: Consider the domain of the attribute Age as a hedge algebra, then we choose the structure of the hedge algebra as follows: $\underline{\boldsymbol{X}} = (\boldsymbol{X}, G, H, \leq)$, with $\boldsymbol{X}$ *is* Age, G = {0, "*young*", W, "*old*", 1}, $H^- = \{less, possibly\}$, $H^+ = \{more, very\}$, We given $W = 0.6$, α= 0.6 and β = 0.4. We have : $fm(young) = 0.6$, $fm(old) = 0.4$, $\mu(possibly) = 0.4$, $\mu(less) = 0.2$, $\mu(more) = 0.25$ and $\mu(very) = 0.15$. Suppose that D_{Age} =[0..100]. Then, we partition the interval [0,100] into 5 intervals similar of level 1: $S(0)$, $S(young)$, $S(W)$, $S(old)$ and $S(1)$. We have: $fm(very\ old) \times 100 = 0.15 \times 0.4 \times 100 = 6$. So $S(1) \times 100 = (94 , 100]$; $(fm(possibility\ old) + fm(more\ old)) \times 100 = (0.4 \times 0.4 + 0.25 \times 0.4) \times 100 = 26$ and $S(old) \times 100 = (68, 94]$; $(fm(less\ young) + fm(less\ old)) \times 100 = (0.2 \times 0.6 + 0.2 \times 0.4) \times 100 = 20$ and $S(W) \times 100 = (48, 68]$; $(fm(possibly\ young) + fm(more\ young)) \times 100 = (0.4 \times 0.6 + 0.25 \times 0.6) \times 100 = 39$ and $S(young) \times 100 = (9, 48]$, $S(0) \times 100 = [0, 9]$.

3. Fuzzy Join Algorithms

3.1 Hamming distance

The Hamming distance is the number of places where two strings of the same length differ. It measures how different the strings are, and is used in many fields such as information theory and computer science. This difference is determined by comparing each corresponding character in the two strings and counting how many characters differ. The larger the Hamming distance, the more different the two strings are.

Algorithm 1: **Hamming_Distance**
Input: Two strings s and t have the same length n
Output: Hamming distance between s and t

Function Hamming_ Distance(s, t)
1. distance ← 0
2. for i ← 0 to length(s) - 1 do
3. if s[i] ≠ t[i] then
4 . distance ← distance + 1
5. return distance
The complexity of the algorithm is O(max(|s|,|t|)).

3.2. Levenshtein distance

The Levenshtein distance is a metric in computer science that measures the differences between two strings. It calculates the minimum number of edits (insertions, deletions, or substitutions) required to transform one string into the other. The smaller this distance, the more similar the two strings are. This algorithm is commonly used in applications such as spell checking, speech recognition, and biology to compare DNA sequences.

Algorithm 2. Levenshtein_Distance
Input: Two strings s and t
Output: Levenshtein distance between s and t

Function Levenshtein_ Distance (s, t)
1. m ← length(s)
2. n ← length(t)
3. Create matrix D[0..m][0.. n]
4. for i ← 0 to m do D [i][0] ← i
5. for j ← 0 to n do D[0][j] ← j
6. for i ← 1 to m do
7. for j ← 1 to n do
8. if s[i-1] = t[j-1] cost ← 0 else cost ←1
9. D[i][j] ← min(D[i- 1][j] + 1, D[i][j-1] + 1, D[i- 1][j-1] + cost)
10. return D[m][n]
The complexity of the algorithm is O(|s|*|t|).

3.3. Longest common subsequence

The longest common substring (LCS) of two strings is the longest substring that appears in both strings in the correct order but not necessarily consecutively. This is an important problem in dynamic programming and has applications in text comparison, DNA sequence analysis, etc.

Algorithm 3. Longest common subsequence

Input: Two strings s and t

Output: Length of longest common subsequence of s and t

Function LCS(s, t)
1) m ← length(s)
2) n ← length(t)
3) Create table L[0..m][0.. n]
4) for i ← 0 to m do L[i][0] ← 0
5) for j ← 0 to n do L[0][j] ← 0
6) for i ← 1 to m do
7) for j ← 1 to n do
8) if s[i-1] = t[j-1] then
9) L[i][j] ← L[i- 1][j-1] + 1
10) else
11) L[i][j] ← max(L[i- 1][j], L[i][j-1])
12) return L[m][n]

The complexity of the algorithm is O(|s|*|t|).

4. Fuzzy join algorithm using hedge algebra

For the algorithms in section 3 used to join data from two sets when the linking keys do not match exactly (fuzzy matching). It works by calculating the distance or similarity between strings, allowing for searching and join records that are approximately similar. For example, fuzzy matching can find the combination of 'John Smith' and 'Jon Smythe' even if they are not an exact match. This technique is useful when working with data that is imperfect, erroneous, or has a lot of variation.

However, in practice, there are many cases where we want to join two fuzzy data sets, for example, two Age data sets. The Age attribute is also called a fuzzy attribute because its domain allows for fuzzy values. In that case, a 25 years old employee is considered “young” or 45 years old employee is considered “middle-aged”, which cannot be handled by the algorithms in section 3. Therefore, in this section, we propose a new fuzzy join algorithm based on hedge algebra to processing these types of fuzzy data in big data.

Algorithm 4. Fuzzy_join_Hedge Algebra

Input : Two values x, y; where x ∈Dom(X), y ∈Dom(Y); X, Y are fuzzy attribute

Output : True or False

Function Fuzzy_ join(x, y)
1) Building two hedge algebras $\underline{\boldsymbol{X}}$ and $\underline{\boldsymbol{Y}}$ based on Dom(X) and Dom(Y)
2) Let Dom(X)= $D_X \cup LD_X$ and Dom(Y)= $D_Y \cup LD_Y$
3) Partition D_X and D_Y into similar intervals of level 1, level 2, level 3,… level k. Let Ω_k (y) = S_k(y) is the neighborhood of level *k* of y.
4) if x ∈D_X and y∈ D_Y then
5) if x = y then check_function = True else check_function = False
6) if x ∈D_X and y∈ LD_Y then
   if x∈ Ω_k(y)*|D_Y| check_function = True else check_function = False
7) if x ∈LD_X and y∈ D_Y then
   if y∈Ω_k(x)*|D_X| check_function = True else check_function = Talse
8) if x ∈LD_X and y ∈LD_Y then
   if Ω_k(x)=Ω_k(y) then check_function = True else check_function= False
9) return check_function

The complexity of the algorithm is O(n+m), where n and m are the number of elements in Dom(X) and Dom(Y) respectively.

5. Conclusion

In this paper, we present four fuzzy join algorithms, in which the fourth algorithm is a new proposal to solve the semantics of fuzzy data that the algorithms 1 to the algorithms 3 have not solved. The Algorithms 1 to algorithms 3 have been experimentally implemented by the authors [4] based on the standard data set created in reference [1]. Fuzzy join algorithms in big data processing bring many important meanings, especially in fields such as data mining, image processing, clustering, and machine learning. In the big data environment, determining the relationship between data elements is no longer a simple problem due to the uncertainty, noise, and heterogeneity of the data. Fuzzy join algorithms are developed to flexibly model imprecise or undefined relationships, helping to improve the performance of analysis and decision making in artificial intelligence and machine learning systems. Some experimental implementations of th algorithm 4 will be implemented in the future works.

References

[1] F. Ahmad, S. Lee, M. Thottethodi, and T. Vijaykumar, “Puma: Purdue mapreduce benchmarks suite,” Electrical and Computer Engineering, Purdue University, Tech. Rep., 2012.
[2] FN Afrati, AD Sarma, D. Menestrina, A. Parameswaran, and JD Ullman, “Fuzzy Joins Using MapReduce”, presented at the 2012 IEEE 28th International Conference on Data Engineering, 2012, 498-509.
[3] DB Bisandu, R. Prasad, and MM Liman, “Data clustering using efficient similarity measures,” *Journal of Statistics and Management Systems,* Vol. 22, No. 5, 2019, 901–922.
[4] A.C. Phan, TC Phan, “Similarity algorithms for fuzzy join computation in big data processing environment”, Journal of Computer science and Cybernatic, Vol. 39, No. 2, 2023, 101-124.
[5] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification”, in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* , Vol. 1, 2005, 539- 546.
[6] D. Deng, G. Li, S. Hao, J. Wang, and J. Feng, “Massjoin: A mapreduce-based method for scal- able string similarity joins,” print *2014 IEEE 30th International Conference on Data Engineering.* IEEE, 2014, 340–351.
[7] A. Das Sarma, Y. He, and S. Chaudhuri, “Clusterjoin:

A similarity joins framework using map- reduce," *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1059–1070, 2014.

[8] M. Henzinger, "Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms", in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2006, 284-291.

[9] J. Wang, G. Li, and J. Fe, "Fast-join: An efficient method for fuzzy token matching based on string similarity join", in *2011 IEEE 27th International Conference on Data Engineering*, 2011, 458-469.

[10] J. Wang, G. Li, and J. Feng, "Extending String Similarity Join to Tolerant Fuzzy Token Matching", *ACM Trans. Database Syst.*, Vol. 39, No. 1, 2014.

[11] Nguyen Cat Ho, Le Xuan Vinh, Nguyen Cong Hao, *"Unify data and establish similarity relation in linguistic databases upon hedge algebra based"*, Journal of computer science and Cybernetic, Vol.25, No.4, 2009, 314-332.

[12] B. Kimmett, A. Thomo, and V. Srinivasan, "Fuzzy joins in MapReduce: Edit and Jaccard distance", in *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)* , 2016,1-6.

[13] IE Agbehadji, H. Yang, S. Fong, and R. Millham, "The comparative analysis of smith- waterman algorithm with jaro-winkler algorithm for the detection of duplicate health related records," in *2018 International Conference on Advances print Big Data, Computing and Data Communication Systems (icABCD)* . IEEE, 2018, 1 –10.

[14] J. M. Keil, "Efficient bound jaro winkler similarity-based search," *BTW 2019*, 2019

[15] B. Kimmett, A. Thomo, and V. Srinivasan, "Fuzzy joins in mapreduce: Edit and jaccard dis- tance," print *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE, 2016, 1–6.

[16] D. Shapiro, N. Japkowicz, M. Lemay, and M. Bolic, "Fuzzy String Matching with a Deep Neural Network", *Applied Artificial Intelligence*, Vol. 32, No. 1, 2018, 1-12.

Author Profile

Cong Hao Nguyen received his **Ph.D** degree in Computer Science from Institute of Technology Information, VAST in 2008. At present, he is a lecturer of Hue University. His main research interests are fuzzy databases, fuzzy logic and approximate reasoning methods.