A Novel SSPS Framework for String Similarity Join

P. Selvaramalakshmi
Research Scholar,
Department of Computer Science
Bishop Heber College,
Trichy, TamilNadu, India.

S. Hari Ganesh, PhD
Assistant Professor,
Department of Computer Science,
H. H. The Rajah's College.
Pudukottai, TamilNadu, India

Florence Tushabe, PhD
Associate Professor,
UTAMU, Kampala,
Uganda, East Africa.

ABSTRACT

As the enormous growth of information challenges the existing string analysis techniques for processing huge volume of data, there always seem to be a hope for newer inventions. Moreover, the problems encountered with the traditional methods such as low pruning power, increased false positives and poor scalability should be addressed with the appropriate solutions that cater to the need for improvement. Hence, this paper aims at proposing an improved similarity joins using SSPS MapReduce Framework that consists of a novel PSS Stemming algorithm along with three newly proposed filtering techniques such as SSize, SPositional and UI(Union –Intersection) that could effectively process large scale data by concerning the limitations of the traditional filtering methods. The experimentation shows that the framework reduces the false positives and run time cost with increased scalability than the existing frameworks.

Keywords

similarity joins, Hadoop, MapReduce, filtering and Verification

1. INTRODUCTION

Similarity Join is considered as one of the vital tasks in data cleansing and integration that is intended to find the similar pairs of strings from two sets or collections of documents. Thus, offers wide range applications including duplicate detection [1] [2] [3] [4], data cleaning [5] [6], plagiarism detection [7], record linkage [8] and string searching [9] [10]. The traditional methods of string similarity employ a well-known filter –verification framework that embraces two essential steps of filter and verification. The filter extracts the candidate pairs by pruning the large number of dissimilar pairs and the verification retrieves the original similarity of documents by thoroughly evaluating each candidate pair in which the filter requires an intensive care it plays a vital role in the framework.

The typical way of classifying the string similarity is either character or token based metrics [11]. As the intension of this research work is to propose several filtering approaches to process. The token-based filtering approaches have been studied. The metric first converts the strings into token sets and applies the set-based similarity such as Jaccard and Cosine Similarity measures to quantify the similarity [12]. The filtering techniques are also classified according to the types of similarity measures. The state of the art of string similarity join lies under effective modification of filtering techniques w.r.t. similarity metrics which is the influencing factor of this research. Hence, the preceding section presents the recently proposed filtering techniques and their merits and demerits.

The remain sections of the paper is organized as follows: Section 2 deals with the recent literature on filtering techniques, section 3 discusses the SSPS framework and the research contributions of the paper, section 4 describes the experimentation and result discussions and finally, section 5 concludes the findings of the paper.

2. LITERATURE REVIEW ON FILTERING TECHNIQUES

2.1 Count Filtering (CF)

The basic notion of CF is that if two strings are similar, if and only if they share at least C common signatures which implies that the number of shared signatures between two strings which is smaller than C is the string pair that can be pruned. The method takes each token as signature and sets an overlap threshold as γ common signatures C=γ. Two strings 'j' and 'm' are similar w.r.t the overlap similarity can be denoted using the equation

$$C=(|j\cap m|)/(|j|+|m|-|j\cap m|)\geq\gamma \tag{1}$$

If the length of the signature is increased, there could only be fewer strings sharing a common signature causing the inverted lists to be shorter. Thus it may decrease the time taken to merge the inverted lists. In contrast, a lower threshold on the number of common signatures shared by similar strings causes a less selective count filter to eliminate dissimilar string pairs [13]. The number of false positives after merging the lists will increase, causing more time to compute their common signatures in order to verify if they are in the answer to the query.

2.2 Length Filtering (LF)

The length of string may also be considered as one of the joining constraints as the similar strings can be represented with same length. Thus, LF concerns with the pruning of dissimilar pairs w.r.t length difference which means, if two strings are similar, then their difference in length cannot be large than γ [14]. Two strings 'j' and 'm' are similar w.r.t LF can be denoted using the equation

$$\gamma|j|\leq|m|\leq(|j|)/\gamma \tag{2}$$

LF is attained by partitioning the strings into group of strings of same length. The pruning of two groups of string is done when the length of the strings are dissimilar. LF increases the join cost and false positives which would in causes low pruning power which affects the scalability.

2.3 Prefix Filtering (PF)

PF sorts the tokens in an ordered sequence of list such as alphabetical or inverse document frequency and compares the first γ set of prefix signatures within the strings based on the fact, if two strings 'j' and 'm' are similar then the prefix order of the sequence is also similar [15]. Given the overlap threshold γ for each string 'j' the PF 'jp' is calculated using the equation

$$j_p=|j|-\gamma+1 \quad (3)$$

It is proved that if two string j and m are identical, then $jp \cap mp \neq \emptyset$. PF increases space complexity as the signatures are to be stored in the inverted index. Determination of PF ordering is one of the implementation issues for minimizing the number of comparisons. Moreover, the computational cost of PF is high.

The current trend of information technology has been shifting from software development to data analytics. Similarity join is one of the prevalent techniques that widely support data analytics in the reduction of duplicate data as the manual reduction is more complex and time consuming. Though there have been numerous string similarity join algorithms proposed in the literature, they have all been suffered from certain important issues as follows:

- Low Pruning Power
- High Computational Cost
- Increased False Positives
- High Space Complexity
- Lack of Scalability

Hence, the research on the improvisation of string similarity join has always been considered as a thrust area.

3. METHODOLOGY

This paper presents a MapReduce Hadoop Framework for effectively handling large scale data with the motivation of performing a scalable string similarity joins through an iterative Map and Reduce phases. The framework consists of three stages where each of which consists of its own Mapper Input, Mapper output followed by the reducer Input and Reducer Output phases and at the centre consists of the processing instructions of the reducer phase. A map function produces a key-value pair of input records and sends out a list of intermediate key-value pairs. A Reduce function accepts the list of values equivalent to an identical intermediate key and processes it to throw out a list of key or values. The framework has the ability to perform both character and set-based similarity functions that does not compromising the traditional filter and verification framework through a parallel distributed processing.

This work introduces four novel research contributions that overcome the limitations of the existing similarity join framework. One such contribution is PSS algorithm (Prefix Suffix Stripping algorithm that cleans the document strings by stripping the prefixes and suffixes to extract their stem that play a major role in increasing the true positives as it is directly involved with similarity joins. Moreover, the thesis also proposes three novel filtering techniques namely SSize, SPositional and Intersection-Union Filters that generates the signature for each sting to prove that two strings are similar if and only if they share common signatures. This property is utilized to generate the candidate key-value pairs. The verification step evaluates the candidate pair to generate final results. The framework of the proposed methodology is shown in Figure 1.

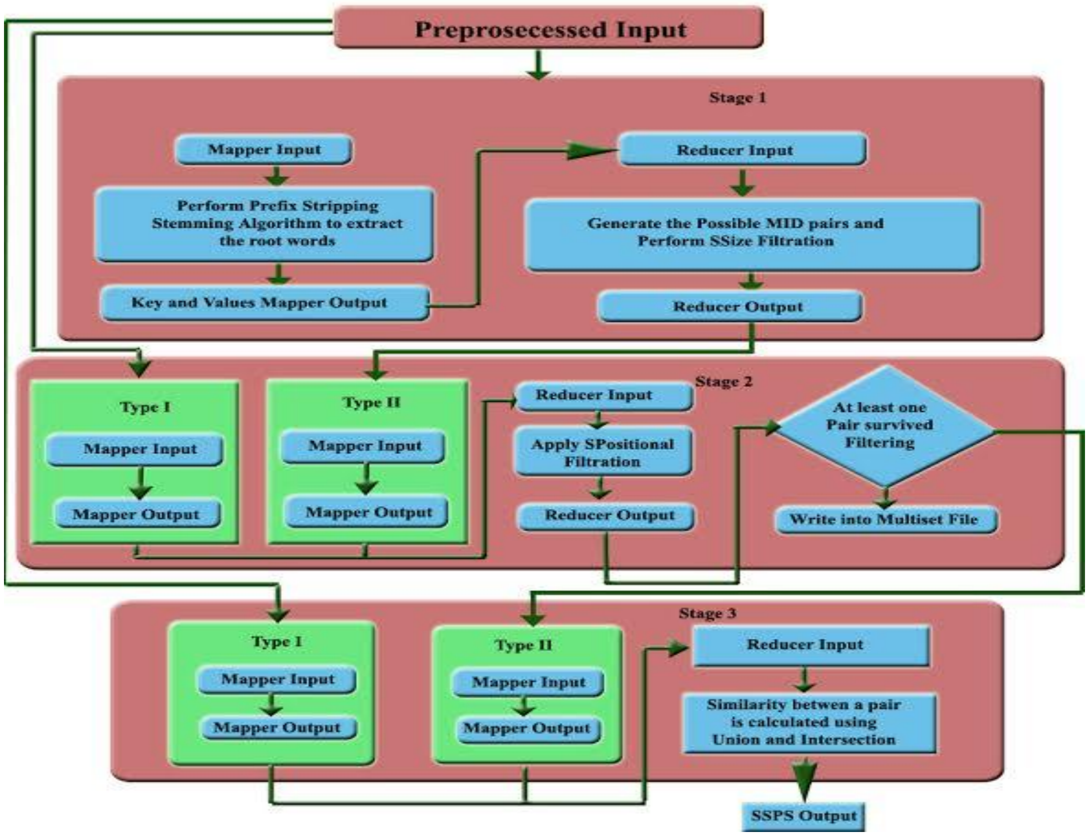
Preprosecessed Input
Stage 1
Mapper Input
Perform Prefix Stripping Stemming Algorithm to extract the root words
Key and Values Mapper Output
Reducer Input
Generate the Possible MID pairs and Perform SSize Filtration
Reducer Output
Stage 2
Type I
Mapper Input
Mapper Output
Type II
Mapper Input
Mapper Output
Reducer Input
Apply SPositional Filtration
Reducer Output
At least one Pair survived Filtering
Write into Multiset File
Stage 3
Type I
Mapper Input
Mapper Output
Type II
Mapper Input
Mapper Output
Reducer Input
Similarity betwen a pair is calculated using Union and Intersection
SSPS Output

Fig 1. SSPS Framework

The following are the four novel research contributions of this paper that comprises of a stemming algorithm and three filtering techniques that differ in the way the strings are filtered.

3.1 Stage 1: SSize Filtration MapReducer

3.1.1 Prefix-Suffix Stripping (PSS) Algorithm

The PSS algorithm separates the strings of inputted documents as individual tokens. The algorithm then derives two substrings consists of first five and last seven characters from each token. The reason for making the substrings of first five and last seven characters is that the highest number of characters with prefixes is five and suffixes are seven. The substring that consists of the first five characters is called a pgram and last seven characters is called sgram. Consequently, the algorithm compares pgram with the strings stored in the prefix-suffix (ps) table consists of the prefixes and suffixes. If the pgram exists with ps table it simply removes the pgram from the token and calls it as stem, and passes the stem to the next stage of suffix stripping or else the characters of pgram is reduced by one and the same process is repeated until it finds the right match or the end of pgram is reached. The resultant stem is then passed on to the suffix stripping stage. The same process is repeated with the sgram to identify the suffixes of the given words. The stem words that are derived by the PSS algorithm are finally compared with WordNet of glossaries which is attached to the software for extracting the root word of each token. The tokens are documented into multisets.

3.1.2 Mapper

The basic notion of SSize Filtration is that if two strings are similar, if and only if they share at least C common signatures with the size threshold γ between two strings. The preprocessed input to the Map Phase comprises of various records, each consisting the Multiset ID, Multiset (Mi), followed by the elements of Mi. The elements of Mi are arranged based on the increasing order of the global alphabetical order of frequency. Each Mapper calculates the SSize of the strings of the multiset in the input record. For each string j, the SSize filtration is the multiplication of sum of the size of the strings ‘j’ and ‘m’ with the threshold γ divided by total number of elements in both sets. The SSize key is calculated by summing up the frequencies of the data elements of the multiset elements present in Mi as given by the equation

$$SSize = \frac{\sum_{i=1}^{n} |M_i| * \gamma}{Total\ Number\ of\ sets} \quad (4)$$

Where, SSize is the multiplication of addition of size multisets $i_1 \ldots i_n$ with threshold γ divided by the total number of sets. For instance, let the strings be “database systems” and “database concepts” and the size threshold for the sets be 0.9. SSize implies at least one word should be common in both sets as follows.

The total number of elements of the sets is 4.

$$SSize = \frac{4 \times 0.9}{2} = \frac{2.4}{2} = 1.2$$

3.1.3 Reducer

The required intersection ratio between the two sets should be one which means the strings must share at least one element within each other or otherwise the strings can be pruned.

In the reducer section of stage 1, the records that share the data elements that fit into the SSize threshold are grouped separately. Using stemming technique, SSize elements in multisets are grouped. If two multisets have a common data element, which are potential candidates of being similar are grouped. Therefore, all the possible MID pairs that share the same results are generated. To reduce this number further, SSize filtering technique is applied for effective pruning results. The SSize filtration technique is applied using the size information sent with every record. For every MID pair, {Mi , Mj} and threshold t, if the SSize filtering condition, | Mj |≥ t* | Mi |, is satisfied, it passes the filter; otherwise it is pruned. For every MID pair, {Mi , Mj} that survives size filtering, the frequency of dk , size and position information of both Mi and Mj are appended and sent as the reducer output to the second stage of the framework.

3.2 Stage 2: SPositional Filtration MapReducer

SPositional filtering is the technique that filters the pairs of sets, based on the positional information of the overlapping token between the sets. An important aspect of the Stage II-Reduce Phase is SPositional filtering. Stage II –Map phase consists of two types of Mappers.

3.2.1 Type 1 Mapper

The preprocessed input of Stage I-Map Phase, where each records consisting of the MID, Mi and its elements, are read and sent as output with {Mi, m} as the key and the elements of Mi as the value. Here, the ‘m’ is the key denotes that position of data in the multiset elements called multiset records. The proposed algorithm customizes the data and its position. A multiset record has compounded with its position {Mi, m}, where Mi is the key and m is the position. Records are intersected based on the primary key. Both types of records, for which the primary key and Mi is the same, are partitioned to the same reducer. Custom grouping ensures the records that have the same MID, Mi as the primary key reach the same instance pertains to a unique MID Mi.

3.2.2 Type II Mappers

The records obtained from the output of Stage I-Reduce Phase are read. These records relate to MID pairs are denoted as MID Pair records. The output key is the MID pair {Mi, Mj}, which comprises of the frequency of dk, and the positional information of both Mi and Mj, to assist SPositional filtering in the Reduce Phase. In the Stage I-Reduce Phase, the records which sharing the common position signature can be joined together and sent as output. The notion of Spositional Filtration is derived by the sequence of intersection of elements in the sets which can be denoted using the Jaccard Equation

$$Jac_{SP} = \frac{|j_{sp} \cap m_{sp}|}{|j| + |m| - |j \cup m|} \geq \gamma \quad (5)$$

The elements are compared with both similarity and the index position of the sets. For instance, the strings “I like chocolate” and “I love chocolate” is evaluated as follows with the similarity threshold 0.9.

1	2	3
I	Like	chocolate
1	2	3
I	Love	chocolate

The SPositional intersection of the strings results the similarity matches of indices 1 and 3. Hence, the number of SPositional intersected elements is 2. The value is then validated against the Jaccard Coefficient similarity measure as follows:

$$Jac_{sp} = \frac{2}{|3| + |3| - |4|} = \frac{2}{2} = 1 \geq 0.9$$

As the SPositional intersection of the strings is greater than the threshold, the strings can be paired to for the similarity join.

3.2.3 Reducer

Candidate pairs can be generated by the Type I-Mappers of Stage II. The records which have the same Mi as primary key are grouped in the same instance. These include the multiset record corresponding to Mi and the MID Pair records with the same Mi as their primary key. In every reduce instance, the multiset record with key, {Mi ,m}, arrives are interpreted to retrieve the results. The MID Pair records that pertain to the same {Mi , Mj} pair are grouped together and SPositional filtering is applied. Every unique pair {Mi , Mj}, that survives SPositional filtering is sent as output. If there is at least one pair that survives SPositional filtering, MID Mi and its elements are written to a file named as the Multiset File.

3.3 Stage 3: Union Intersection (UI) MapReducer

Stage III-map phase also consists of two types of Mapper.

3.3.1 Type I Mapper

It reads- the preprocessed input of Stage I-Map Phase, where every record consists of the MID, Mi and the elements of Mi. These records are sent as output with {Mi ,m} as the key and the elements of Mi as the value.

3.3.2 Type II Mapper

It reads the outputs of Stage II-Reduce Phase which are the MID Pairs. In the previous stage, the elements of every multiset, Mi, having at least one pair surviving SPositional filtering, having Mi as the first of the pair is written to the Multiset File. So, the elements of multiset Mi can be retrieved from the Multiset File in the Reduce Phase, but we cannot retrieve elements of multiset Mj from it. To solve this problem, the record is reversed and sent out from the Mapper with the {Mj , Mi} as the key and Mj as the value.

3.3. 3 Reducer

The Multiset File is loaded into the memory by every reduce node. Partitioning, Grouping, and Sorting are done in the same way as Stage II Reducer. Records that have the same Mi as the first part of the key arrives at each reduce instance. Every MID pair {Mi , Mj} gets the elements of Mi from the multiset record that arrives to the same instance, and looks up the Multiset File for the multiset elements of Mj.

4. EXPERIMENTATION

The experiments have been conducted on a Hadoop cluster with 51 virtual nodes and one additional node for handling the Hadoop master daemons. Each node has a memory allocation of 8 GB, a single 2.8 GHz CPU, 64bit Operating Systems and 40 GB of disk space. The simulations are completed using 60 GB of raw twitter data in the JSON format. These data are preprocessed to remove stop words and the root words are extracted to get the desired form. Each record containing the user's ID and a multiset of the words of the tweets which are sent by the user. Similarity Joins are performed between the multisets of various twitter users taken from different scenarios to determine their similarity. The experimentation is made with the intension of comparing the performance of the proposed SSPS with the existing SSS and SSJ-2R in terms of similarity pair reduction, running time and accuracy by setting the threshold limit as 0.7. Table 1 depicts the comparison of similarity pair reduction of the experimental methods, with four sets of tweets with varied record sets, where the reduction of pairs with the proposed SSPS is minimum than the other two methods.

Table 1. Comparison of Candidate Pair Reduction

Number of Records	Algorithm	Reduced Pair
7281	SSPS	4328
	SSS	5241
	SSJ-2R	5793
11306	SSPS	8036
	SSS	9355
	SSJ-2R	9920
14336	SSPS	9631
	SSS	12173
	SSJ-2R	13189
16244	SSPS	10785
	SSS	12963
	SSJ-2R	14824

Figure 2 denotes the graphical representation of similarity pair reduction comparison of SSPS with the SSS and SSJ-2R, where the x-axis of the graph represents the experimental methods with the number of records in the twitter dataset and y-axis represents the reduced pair reduction in number units.

Table 2 shows the running time (in milliseconds) and the performance improvement analysis (effective reduction in %) of SSPS with the experimental methods of SSS and SSJ-2R, tested over the twitter datasets with the threshold value of 0.7.

The visual representation of the run time analysis of SSPS, SSS and SSJ-2R is presented in Figure 3, proves the time taken to process the records using SSPS is considerably minimum than the SSS and SSJ-2R for all four datasets. The x-axis of the graph represents the record size of the datasets and the y-axis represents the running time in milliseconds.

Figure 4 denotes the effective reduction percentage of string similarity pairs in percentage which also emphasizes that the reduction of similarity pairs with SSPS has achieved the highest reduction accuracy than SSS and SSJ-2R for all four experimental datasets.

Thus, the proposed method has proven that it could produce highest reduction accuracy with minimum run time than the existing frameworks of SSS and SSJ-2R. The x-axis of Figure 4 denotes the MapReducer methods with number of records and y-axis denotes the reduction accuracy in percentage. When the reduction accuracy is high, the pruning power is also high. Hence, the framework is suitable to all real time string similarity joins applications.

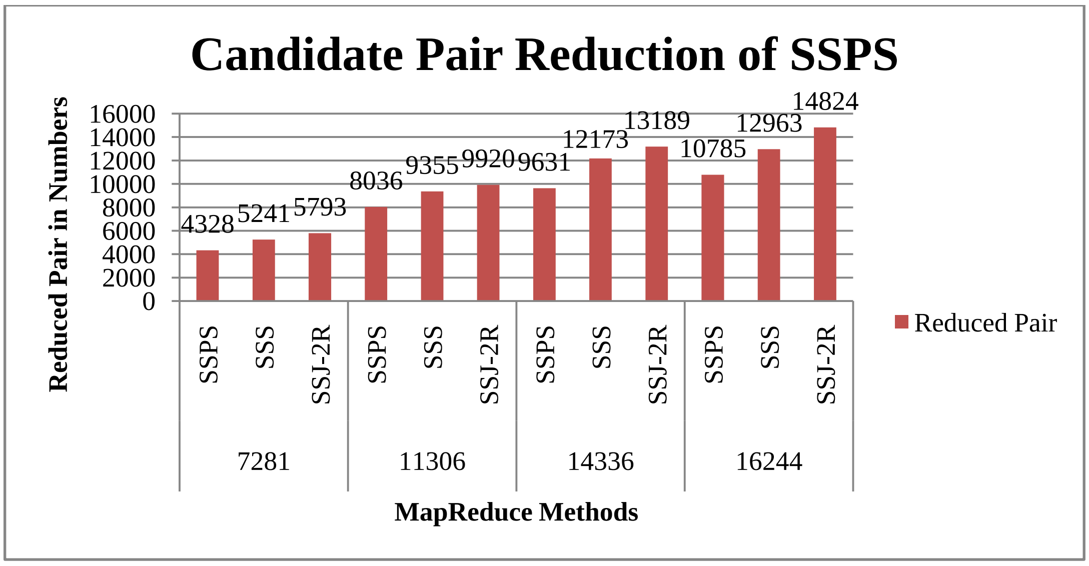

Fig 2. Candidate Pair Reduction Comparison

Table 2: Performance Analysis of SSPS

Number of Records	Algorithm	Running Time (ms)	Performance Improvement (%)
7281	SSPS	321	91%
	SSS	558	86%
	SSJ-2R	1763	78%
11306	SSPS	655	92%
	SSS	897	85%
	SSJ-2R	2905	79%
14336	SSPS	1211	86%
	SSS	1620	74%
	SSJ-2R	4504	79%
16244	SSPS	1710	85%
	SSS	1855	83%
	SSJ-2R	6480	76%

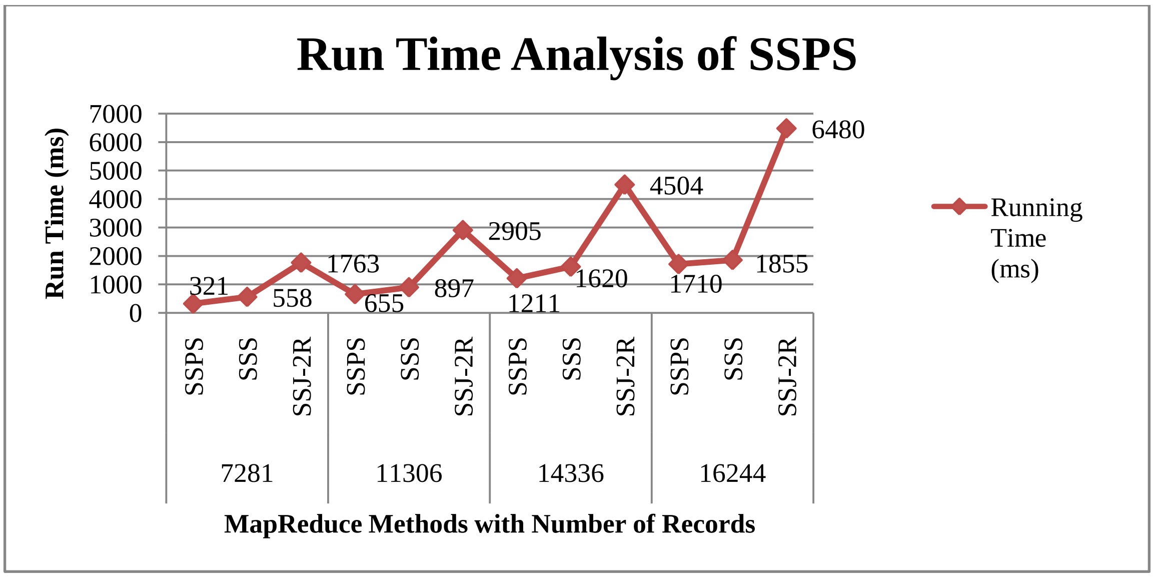

Fig 3. Run Time Analysis of SSPS

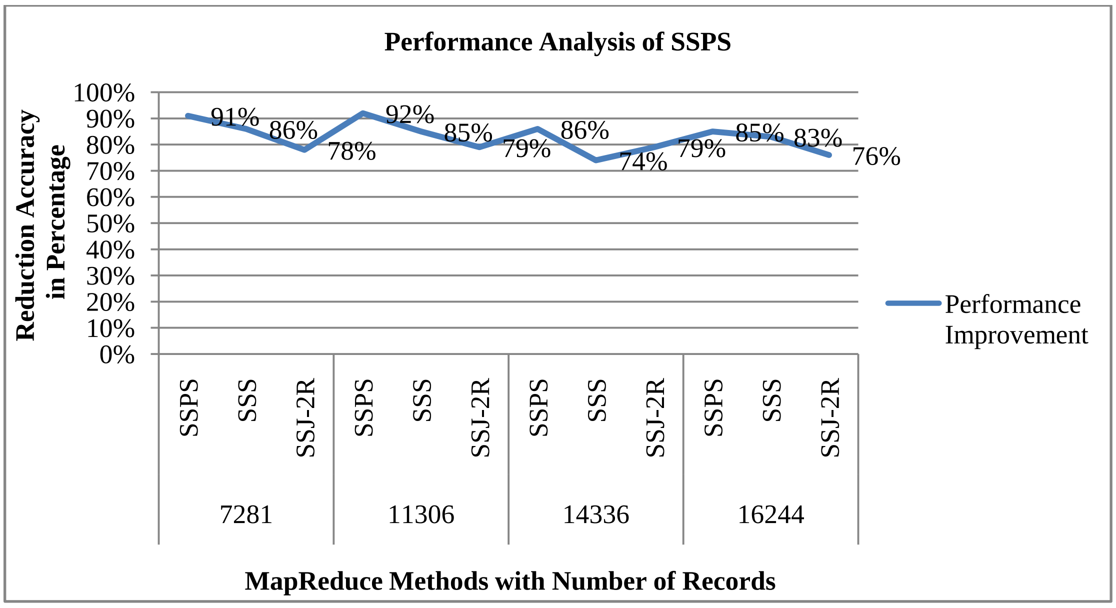

Fig 4. Performance Analysis of SSPS

5. CONCLUSION

The string similarity join filtering methods that have been proposed in this paper are more prevalent than the traditional frameworks as it reduces the percentage of false positives through the successful reduction of similar pairs of strings. The SSPS framework is also found to be effective with its high pruning power with increased scalability as its performance is stable even with increased number of records. Moreover, the time taken to process the similarity reduction is also proven as minimum when compared to other string similarity reduction frameworks such as SSS and SSJ-2R. Hence, the framework is highly suggested to the scenarios where the similarity of strings is to be performed. In future, this research work may also to be extended to perform the clustering of similar opinions or ideas pertained to strings over the large-scale data to analyze the underlying facts. Moreover, the proposed work can further be extended to test on more datasets to generalize the findings.

6. REFERENCES

[1] Fetterly D, Manasse M, Najork M (2003) On the evolution of clusters of near-duplicate web pages. J Web Eng 2(4):228–246

[2] Henzinger M (2006) 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. ACM, New York, pp 284–291

[3] Sarawagi S, Bhamidipaty A (2002) Interactive deduplication using active learning. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, pp 269–278

[4] Xiao C,WangW, Lin X, Yu JX,Wang G (2011) Efficient similarity joins for near-duplicate detection. ACM Trans Database Syst (TODS) 36(3):15

[5] Baraglia R, De Francisci Morales G, LuccheseC(2010) Document similarity self-joinwith mapreduce. In: 2010 IEEE 10th International Conference on Data Mining (ICDM), IEEE, pp 731–736

[6] Elsayed T, Lin J, Oard DW (2008) Pairwise document similarity in large collections with mapreduce. In: Proceedings of the 46th annual meeting of the association for computational linguistics on human language technologies: short papers. association for, computational linguistics, pp 265–268

[7] Hoad TC, Zobel J (2003) Methods for identifying versioned and plagiarized documents. J Am Soc Inf Sci Technol 54(3):203–215

[8] Winkler WE (1999) The state of record linkage and current research problems. In: Statistical Research Division, US Census Bureau, Citeseer

[9] Hadjieleftheriou M, Chandel A, Koudas N, Srivastava D (2008) Fast indexes and algorithms for set similarity selection queries. In: IEEE 24th International Conference on Data Engineering, 2008. ICDE 2008. IEEE, New York pp 267–276

[10] Henzinger M (2006) 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. ACM, New York, pp 284–291

[11] Jiang, Y., Li, G., Feng, J. and Li, W.S., 2014. String similarity joins: An experimental evaluation. Proceedings of the VLDB Endowment, 7(8), pp.625-636.

[12] Deng, D., Li, G., Hao, S., Wang, J. and Feng, J., 2014, March. Massjoin: A mapreduce-based method for scalable string similarity joins. In 2014 IEEE 30th International Conference on Data Engineering (pp. 340-351). IEEE.

[13] Li, C., Wang, B. and Yang, X., 2007, September. VGRAM: Improving performance of approximate queries on string collections using variable-length grams. In Proceedings of the 33rd international conference on Very large data bases (pp. 303-314). VLDB Endowment.

[14] Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S. and Srivastava, D., 2001, September. Approximate string joins in a database (almost) for free. In VLDB (Vol. 1, pp. 491-500).

[15] Wang, J., Li, G. and Feng, J., 2012, May. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (pp. 85-96). ACM.