A Similarity-Based Method for Entity Coreference Resolution in Big Data Environment

Yushui Geng[1, a], Peng Li[2, b] and Jing Zhao[2, c]

[1]School of Information, Qilu University of Technology, Jinan 250353, China;

[2]School of Information, Qilu University of Technology, Jinan 250353, China.

[a]gys@qlu.edu.cn, [b]brunomarslee@yeah.net, [c]zj@qlu.edu.cn

Keywords: big data, entity coreference resolution, similarity, MapReduce.

Abstract. Processing and analyzing large scale data is needed in the big data environment, however, a large number of duplicate data refer to the same entity in the data set have brought great difficulties to analyze and process the acquired data. The method based on cluster analysis is one of the main methods of entity coreference resolution, but it is time-consuming and does not apply to big data environment. This paper presents a similarity-based method for entity coreference resolution by introducing weight and similarity and using Hadoop platform and MapReduce framework, which will process data into the form of key-value data pairs and can be efficiently applied to the entity coreference resolution. Experiments show that the proposed method greatly improves the speed and accuracy of entity coreference resolution, meets the demand for entity coreference resolution in big data environment.

1. Introduction

With the continuous improvement and development of industrial automation and information technology, many types of large quantities of data have been produced in the industrial field. Structured, unstructured and semi-structured data are constantly increasing at an alarming rate, which brings great difficulties to the analysis and processing of the enterprise and the better use of data. With the coming of the information era, a variety of data are continuously produced, entity coreference resolution facing new difficulties and challenges: a sharp increase in the amount of data, calculation quantity and calculation difficulty increased, calculation efficiency has also become an urgent problem; a variety of data source, there exists various kinds of structured data, semi-structured and unstructured data is dominant, while there are a lot of noise data at the same time; there is a complex relationship between the data and need more information to distinguish the relationship between different entities.

In one or more databases, the same real world entities may be described in many ways. Because of the difference of locating of object and information, the information types from different data sources are various, and the descriptions of the same entities are not the same. And the purpose of entity coreference resolution is to identify the tuples of the same real world entity from the data sets. Entity coreference resolution results can be widely used in other stages of data quality management. This problem, which is described by the same entity, is also prevalent in the information systems in various application fields.

2. Related Work

In the data sets, some data records have different descriptions of the same entity, the process of identifying the records referring to the same entity and merging together is the process of entity coreference resolution. Entity coreference resolution technology is mainly through some calculation rules and using a calculation method to identify an entity, whether it is the same one before. As entity coreference resolution is of great significance in data quality management, the research of entity coreference resolution has also been paid enough attention to. In the previous studies, entity coreference resolution algorithm[1] is mainly to detect duplicate records in so as to get the

recognition results, and the method for calculating string similarity[2] and the parallel[3] method are proposed, but the expansibility is not high. Based on the real data sets, the literature [4] evaluates the efficiency of entity resolution. In addition, there are also other research, like heuristic method [5], distance function [6], Markoff chain [7] and so on. At present, the latest entity coreference resolution method is to use machine learning[8] algorithm, and A.Cvitas[9] proposed a cooperative learning method.

However, despite the existing methods can effectively identify the entity in many applications, but there are still many deficiencies: currently, entity coreference resolution exists duplicate names and synonyms; traditional entity coreference resolution method is often based on tuple similarity comparison to obtain the results; at present, entity coreference resolution method in the used of similarity measure does not take into account the correlation between different words; the data quality assessment system is not perfect. In order to solve the problem of entity coreference resolution in big data environment, this paper presents a similarity-based method for entity coreference resolution.

3. MapReduce Framework

MapReduce is a model designed for a data-intensive parallel computation, the data is stored in the distributed file system and represented in the form of key-value pair. Using the parallel framework MapReduce on Hadoop to conduct a preliminary screening for input data sets, this could remove the less likely record pairs of the input data sets in a relatively short period of time. The parallel computation saves a lot of work time and improves the efficiency for entity coreference resolution. The idea of MapReduce programming is to break down large tasks into small tasks, which will then be performed respectively, so as to achieve the purpose of reducing processing time. The specific process as shown in Figure 1.

3.1 Split and Format data source

3.1.1 Split

Split operation is based on the case of the source file and divided into a series of InputSplit in accordance with the specific rules, each InputSplit will be processed by a Mapper. Splitting the files is not to split the file to form new file fragmentation copies, but to form of a series of InputSplit. InputSplit contains the data information, such as the file block information, the starting position, the data length, the node list, etc. Therefore, all the data of the splits will be find just according to InputSplit.

The most important task of Split operation is to determine the parameter SplitSize, SplitSize is the size of the split data. Once it is determined, the source file is divided in turn according to the value. If the file is less than this value, then the file will become a separate InputSplit; if the file is larger than this value, then it would be divided in accordance with the SplitSize, left less than SplitSize become part of a separate InputSplit. The rule for determining the SplitSize value is: SplitSize= max{minSize, min{maxSize, blockSize}}. Thus, the size of SplitSize is between SplitSize and blockSize.

3.1.2 Format

Format the divided InputSplit as the <key, value> form of data, and the key is an offset, value is the content of each line. When performing the Map task, every time generation a <key, value> data will call <key, value> form of data once and then passing the value at the same time. Therefore, this part of the operation is not the first to parse all InputSplit into <key, value> form of the data and then overall call the Map function, but each data source that is parsed will be handed over to Mapper to deal with once.

3.2 Map

Mapper receives the <key, value> form of data and processes into <key, value> form. The specific process can be defined by the user.

3.3 Shuffle

From the results of Mapper direct output to it becomes to the final Reducer direct input data after a series of processing, the above process is the whole process of Shuffle and it is also the core process of MapReduce.

The whole process of Shuffle can be divided into two stages: the Mapper-side Shuffle and the Reducer-side Shuffle. The data generated by the Mapper is not directly written to disk, but first to storage in memory; when memory data reaches the set threshold, then the data is written to the local disk and simultaneously do the sort, combine, partition, etc. Sort operation is to sort the results that generated by Mapper according to the key value; combine operation is to combine the adjacent records with the same key value; partition operation involves how to evenly distribute the data to multiple Reducers, it is directly related to the load balance of Reducer. Among them, the combine operation does not necessarily to have, because it is not applicable in certain scenarios; but in order to make the output results of Mapper more compact, it will be used in most cases.

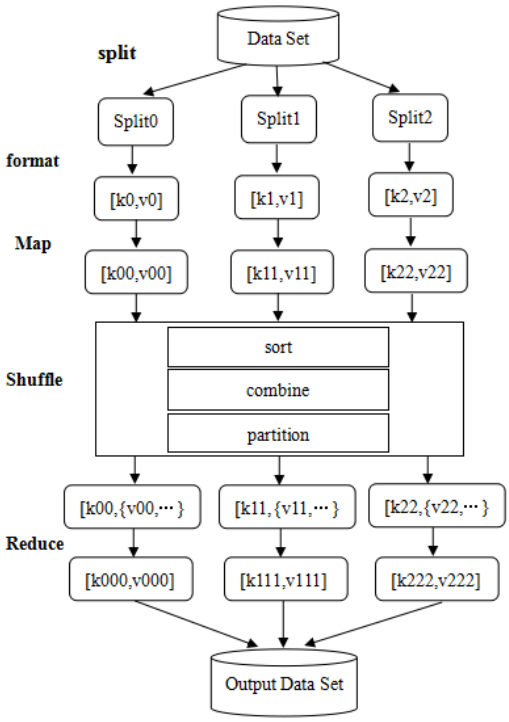

Figure. 1 MapReduce Workflow Diagram in Entity Coreference Resolution

3.3.1 Mapper-side Shuffle

Data generated by Mapper is not directly written to disk, because it will produce a lot of disk IO operations and it will directly restrict the operation of Mapper task, so Mapper data will be written to memory and it will be written to disk again by polling when reaching a certain number. This can not only reduce the disk IO, but also allows the data enable to do appropriate operation when written to the disk.

Each Mapper task has an output cache and a write threshold in memory and the default is 80MB and 80%. When the write cache data proportion reaches this threshold, the Mapper will continue to write data to the rest of the cache, but it will start a new process in the background to sort the cached data in front of 80%, then written to the local disk, the operation is called spill, and the write disk file is called spill file or overflow writing file. If the remaining 20% cache has been filled and the previous spill operation is not yet completed, the Map task will be blocked until the spill operation is completed and then continue to write data to the cache. When in the spill operation, if the combine function is defined, do the combine operation after the sort operation and then write to the disk.

Sort process is twice quick sort by key based on the data source, collecting the data that contains the same key together after the sort. Thus, it has a very large significance whether for the

back of the combine operation or sort merge operation. The combine operation is to merge the data with the same key into one row, which must be done after the sort operation is complete.

There will be more than one spill files after a Map task is complete. It is clear that the data for each spill itself is ordered, but they are not the whole global order. Here merge sort will be used to merge all the spill files into one file, and partition method based on interval will be provided in the process of merging. The method will partition by size of the combined file, to ensure that the data in the next partition is bigger than the previous partition on the key values, and each partition will be assigned to a Reducer. So, in addition to finally get a large data files, but also get a index file, in which stores the location offset of each partition data.

Merge sort is a multiple merging process, and the default number of the simultaneously open files is 10. But not the more simultaneously open files, the more faster the merging speed will be. After the completion of the merger process, Mapper-side task will come to an end, at this time Mapper will delete the temporary spill files, and notify the TaskTrack task has been completed. Then, Reducer can get the corresponding data from the Mapper-side through the HTTP protocol. Generally speaking, a MapReduce task will have multiple Mappers and distributed in different nodes; they tend not to be completed at the same time, but as long as there is a Mapper task to complete, Reducer-side will begin to copy data.

3.3.2 Reducer-side Shuffle

From the completion of the merge task in Mapper-side, to the completion of copying data from each node and complete the copy task in Reducer-side, those are completed by the MRApplicationMaster scheduling. After Reduce has removed all the data, the output data in Mapper-side would not be deleted immediately, because the Reduce task may fail, and the data will also be utilized in the time of speculation.

Reducer-side will use multiple threads to copy data from the Mapper-side through the HTTP protocol, the number of threads can be set and the default value is 5. The copied data will not be written directly to disk, but will be stored in the JVM heap memory (JVMheap); when the maximum of heap memory is determined, the size Reducer occupied will be determined through a threshold, and the threshold can be set, but the default is 70%.

Mapper-side output data may be compressed, Reducer-side receives this data that will be automatically extracted when written to a memory, so as to be convenient for the later merge sort operation; and set up a combiner, which will also be called in merge sort operation. Sometimes, when the data is received, the file from merge sort in memory is not much, this time sort merge would deal with the data from these files and memory together, and directly passing them to Reducer to process. So, from a macro point of view, direct input data from Reducer is actually sort merge output stream. In actual processing, sort merge makes each sorted key values call Reduce function once so that to achieve data transmission.

3.4 Reduce

Reducer accepts data stream in the form of <key, {value, list}>, makes the data in form of <key, value> and output them, then the output data will be written directly to the HDFS. The specific process can be defined by the user.

3.5 OutputFormat

OutputFormat describes the output form of the data, and will generate a corresponding class object. Call the appropriate write() method to write data to the HDFS, and the user can also modify these methods to achieve the desired output format. When the Task is performed, MapReduce framework will pass the <key, value> that generated by Reduce to write method automatically, and the write method achieves the writing to the file.

4. The Process of Entity Coreference Resolution

Step 1: Taking the object described by data as the entity, and preprocessing the data in the data sets. K fields of each data are selected as the key and the entire data record as the value to constitute form of <key, value>.

Step 2: Computing the Cartesian product of each data sets, that is, each data and any other one will be made pairs respectively, and to constitute the form of data pairs. For example, there are four data like A,B,C,D, and then the six data pairs will be composed of AB, AC, AD, BC,BD, CD.

Step 3: For entity e_i and e_j, the more similar the content of the k fields information are, the more close to the same entity the two entities will be. The k fields are given corresponding weight w, and the w of each field is different. For the entity e_i, the more the decisive factor of the field is, the greater the value of its w_i will be. Then, according to the importance of the field, the weight of each part can be set. The similarity of each entity pair can be calculated according to the values of w_i. Computing the similarity value and then screening each data pairs. Set the similarity threshold is s, the entities which reach the specified threshold will enter step 4, and all the other data waiting to enter the step 5.

Step 4: All the data pairs that reach the threshold will be unified, that is to merge the same entity pairs into one data. This process is to merge the same kind data and form a data set with unified entities.

Step 5: A new data set is formed by aggregating the data from the entity pairs from the step 3 and step 4.

After the above five steps, entity data set is gradually screened from the vast amounts of data.

The pseudo code is as follows:

Input: $E=(e_1,e_2,e_3...,e_n)$

Output: The data set after coreference resolution.

Process:

```
1.  Map(key key, value value)
2.      for each item d in DataSet
3.          d=0, k=4
4.          if isEntity(value, d.getInfo() )==1
5.              Emit(d.getkey(), d.getvalue());
6.          d++;
7.      end for
8.  Computing Cartessian product of each iteam;
9.  for each key-value data pairs
10.     m=0, i=0, j=0;
11.     w1=0.49,w2=0.63,w3=0.16,w4=0.12;
12.     Similarity(ei, ej)
13.     m++;
14.     s=0.72;
15.     char E[], F[];
16.     if (Sim(ei, ej)>s)
17.         Merge(E[]=(ei, ej));
18.     else
19.         end;
20.  end for
21.  Reduce(key key, value value)
22.  for each key-value data pairst
23.      while(E[]!=null)
24.      Aggregate(E[]);
25.  end for
```

5. Experiment and Analysis

In this paper, experiment environment is based on Hadoop operating platform, and the cluster contains one namenode and twelve datanodes, each node has 512MB memory and 1GB hard disk. Experiment builds Hadoop cluster under the Linux environment, configuring the main node and the

slave nodes, and starting the Hadoop cluster successfully. Experiment data from a manufacturing enterprise with 17,153 items and a electronic commerce website with 9,416 items. In order to evaluate the accuracy of the algorithm, comparing the results of this paper with entity coreference resolution method based on rules, using Precision, Recall and F1-measure as the measurement standard. The five kinds of data sets of commodity, material, staff, book and movie information are used as entities to carry out the experiment. Taking the material data set as an example, the four fields of Name, ID, Price and Color are selected as the key and the entire data record as the value to constitute the form of <key, value>. The experiment chooses the method based on rules as the comparison object to test the effect of two methods respectively at different node numbers in the parallel cluster environment. The results are shown in Figure 2 and Figure 3.

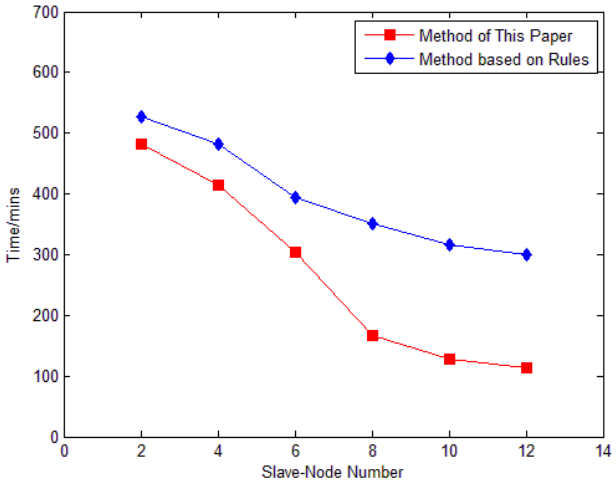

Fig. 2 Speed Results

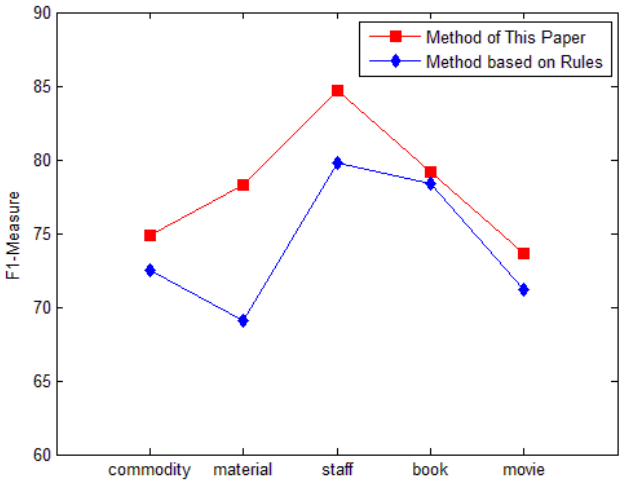

Fig. 3 Accuracy Results

From the results, it can be found that dealing with entity data in the parallel environment greatly improves the operation efficiency, and it is concluded that Hadoop platform operating environment can be well used for entity coreference resolution. The experiment has obvious effect in the following two aspects: Firstly, under the conditions of the same data set, the handling time of the similarity-based method for entity coreference resolution is relatively short, and the processing speed is relatively fast. With the increase of slave nodes, parallel processing efficiency is also accelerating, and then gradually tends to be stable. According to the red line in Figure 2, the advantages of the method based on similarity is more obvious. Secondly, under the conditions of the same entity object, the accuracy of the similarity-based method for entity coreference resolution is relatively high, and it has a high degree of recognition to the entity. The red line in Figure 3 shows that the effect of the similarity-based method is better than another.

According to the above analysis of the results, it is apparent to draw a conclusion that the similarity-based method has greatly improved the efficiency of entity coreference resolution while increasing the speed, so that it has sufficient advantage to be used to entity coreference resolution.

6. Summary

This paper proposes a similarity-based method for entity coreference resolution in big data environment, whose realization relies on introducing weight, similarity and using Hadoop platform and MapReduce framework. Experiments show that the proposed method greatly improves the speed and accuracy of entity coreference resolution. However, because of the limitations of the data sources and other constraints, there are still a lot of work need further in-depth research. In future research, more attention will be paid to other algorithms and computational models so as to better applied to entity coreference resolution.

Acknowledgments

[1] Science and Technology Development Plan Project of Shandong Province (No.2014GGX101052)
[2] Independent Innovation and Achievements Transformation Special Project of Shandong Province (No.2014ZZCX03408)
[3] Shandong Provincial Natural Science Foundation, China (No.ZR2014FQ021)
[4] Key Research And Development Plan Project Of Shandong Province (No.2015GGX106003)
[5] A Project of Shandong Province Higher Educational Science and Technology Program (No.J15LN03)

References

[1] Elmagarmid A K, Ipeirotis P G, Verykios V S.Duplicate Record detection: A survery.IEEE Trans on Knowledge and Data Engineering,2007,19(1:1-16)

[2] Hassanzadeh O, Miller R J.Creating probabilistic databases from duplicated data[J]. The VLDB Journal, 2009,18(5):1141-1166

[3] Vernica R, Carey M J, Li C. Efficient parallel set-similarity joins using MapReduce[C]. Proceedings of the 2010 International Conference on Management of Data, Indianapolis,Indiana, USA: ACM,2010:495-506

[4] Pcke H, Thor A, Rahm E. Evaluation of entity resolution approaches on real-world match problems[J].Proceedings of the Vldb Endowment,2010, 3(1-2):484-493

[5] Arasu A, Chaudhuri S, Kaushik R. Learning String Transformations From Examples[J]. Proceedings of the Vldb Endowment, 2009, 2(1):514-525

[6] Chen Z, Kalashnikov DV, Mehrotra S. Adaptive graphical approach to entity resolution[C].Acm/ieee Joint Conference on Digital Libraries, 2007:204-213

[7] Singla P, Domingos P.Entity Resolution with Markov Logic[C].Proc of IEEE ICDM'06.piscataway, NJ:IEEE, 2006:572-582

[8] Ghoting A,Krishnamurthy R, Pednault E, et al.SystemML: Declarative machine learning on MapReduce.IEEE International Conference on Data Engineering, 2011, 6791(4):231-242

[9] Cvitas A.Relation extraction from text documents[J].MIPRO 2011, Opatija, Croatia, 2011:23-27