Clustering of JSON document using Derived X-Path Method

Kaushik I.
kaushik5699@gmail.com
National Institute of Technology Karnataka (NITK), Mangalore, Karnataka

Abstract—In this modern era where data creation is in abundance, handling that data is the biggest question researchers have been trying to solve for the past couple of decades. The data in any form needs to be stored in such a way that the retrieval and manipulation of the data are easier and simpler. In this paper, we look at how huge datasets in JSON format are being analyzed and made into clusters for an easier understanding of the documents that we are handling. This process is done by the first pattern analyzing the documents followed by clustering those documents one by one on similarity of the attributes (arrays and sub-documents included). The clusters are then evaluated on an algorithm designed specifically for this purpose. To conclude, this paper lays out the model on how to cluster huge JSON documents with varying levels and types.

Keywords—JSON, Cluster, Schema, Attributes, Similarity, Decision Tree, X-Path

1. INTRODUCTION

The proposed work aims to discuss and focus on how JSON documents are clustered. To begin with, what are JSON and its uses. JavaScript Object Notation (JSON) is a data format/data structure that is used extensively to store information in a tree-like structure for more organization and easy access way. This format of data is usually used between server and a client browser embedded with Java scripts and HTML files to ease the communication between the same. It provides many details into the type of data by being able to hold meta-data along with the original data. This type of data makes it easier since there is no need for interpretation into another language since the metadata provides the information required on both ends. JSON is a lightweight data-interchange format, "self-describing” and easy to understand. It is language-independent, and why we used this is because JSON format is text only; it can be communicated between servers and the much simpler data format allows for easier parsing. JSON enables faster accessibility, memory optimization is shorter and simpler in nature, and does not contain complicated syntax tags.

Let’s move on to the limitations of JSON. It is not fully secure. It is limited in terms of the supported data types. There is another issue with JSON as it is quite weak with respect to security. If this JSON is used as the medium of communication with an untrusted service, then there is a possibility of malicious attack during the transmission since JSON service that is returned is wrapped in a function call.

This function call can be used by untrusted certificates that can call this object and can lead to loss of data to the intended receiver.

Clustering is a unique technique for machine learning on unsupervised data. Clustering is the process of segregating the whole group of data points into a number of groups in a strategic way. By strategic, we mean that the clustering of these data points must have some logic as to the clustered datapoints have maximum similarity among them. This means that a given point must have more similarities to the corresponding data point in the same cluster than any other point in the wholeset. This paper explores an entirely python-based approach to JSON Clustering and its feasibility on multiple factors like developer friendliness, deployment interface, efficiency, and scalability. Since this paper also considers the complete structure of the document, the inter-cluster and the intra-cluster similarity is high. The clusters are formed during a single scan of the dataset, which helps in avoiding the multiple scans of the dataset, which in turn reduces the time hence speeding up the Clustering Process.

The further section gives a brief overview of the literature survey done as part of the project. Section 3 shows the motivation behind this project—section 4 details the methodology, including the algorithms used. Section 5 summarizes the paper, also touching upon the limitations of the project and potential future work. Section 6 gives the results. Section 7 gives the final conclusion of our project and section 8 gives a list of references.

2. RELATED WORK

Some studies on JSON in the clustering environment have been carried out as yet. This paper aims to deal with the clustering application and the implantation aspects of clustering with...
So, referring to the query optimization and analysis of document-oriented databases[1], we found that the document-oriented databases end up in a very tedious state where there is a large amount of data, and this data cannot be analyzed in the fastest time possible. Since JSON follows an unordered-list type of format, we have no idea of how the attributes relate to one another and where one attribute might definitely occur. For this very reason, there aren’t many research options to cluster these databases. Hence, we tried to look at the optimized way of representing the JSON documents but were unable to find anything related to optimizing it for further analysis.

There is a process called schema profiling[2]. In schema profiling, they try to figure out the hidden rules in document-oriented databases like JSON, which is further used to ex-plain the variance that is occurring in these datasets. Schema profiling represents these data in the form of a decision tree, which in the end tries to cluster the data. We use schema profiling to build a similarity tree as the schema profiling is per the author. This process provided much better results than going step by step analysis of the JSON documents as it instantaneously profiled all the similar documents, and we were able to calculate the score of the documents. Though this process seemed to be optimistic, we still couldn’t figure out the exact similarity score for the given JSON document and hence were still researching on finding one. In a schema-based split the vertices (say v) has two children and the corresponding edges are labelled, while in a value-based split v has two or more children and the corresponding edges are labelled with a condition expressed over the domain of a. Note that, while any splitting value “val” can be adopted for value-based splits on numeric attributes and binary splits on categorical attributes trueing to deal[3] with a structure and content semantics underlying semi-structured documents, which is challenging for any task of document management and knowledge discovery conceived for such data. In this model, it is addressed that the novel problem of clustering semantically related XML documents according to their structure and content features. Due to this semi-structure of XML, it has raised a lot of issues on representation and extraction of data. A lot of resources have been contributed in order to solve this semi-structure of XML documents and has led to a decrease in the semantic issue up to a certain extent[4].

In [5], they had presented a comprehensive review of structural XML clustering. First, it is provided with a basic introduction to the problem and highlight the main challenges in this research area. Subsequently, later divide the problem into three subtasks and discuss the most common document representations, structural similarity measures, and clustering algorithms. In addition, they also presented the most popular evaluation measures, which can be used to estimate clustering quality. Finally, moving to the clustering part, where most of the XML clustering approaches developed so far employ pairwise similarity measures.

It is essential to point out that the reviewed papers did not present the use cases and benefits of Clustering applications in alternative areas, especially with respect to JSON documents. This paper has classified the exploring and sharing of the Clustering framework. On the other hand, the existing implementations of the Clustering framework and their pros and cons are not included in the published papers thoroughly.

3. MOTIVATION

As we can see, the computerized field has a lot of options for storing and sharing data. So, we can see XML being a data-interchange format. But along with XML, even JSON is a data-interchange format. Although JSON has seen its part in improvements in the field of data analytics, there has always been a bias when it comes to the classification of data. XML has always been easier to classify, given its much familiar format and API services available to it. Given XML has a large amount of data, this data can be classified within few minutes because of the highly optimized algorithms already in place. When it comes to JSON, there has not been an algorithm that has been developed yet to overcome that bias towards XML. As one of the best data formats for accessing, storing and comprehending data. JSON is even better with parsing XML docs are difficult to parse. JSON is even faster and easier than XML. According to researchers view, there area lot of researches going on in the field of XML, including XML clustering. Even though with all these advantages over XML, we cannot find an algorithm that can classify a JSON document in the most accurate and optimized way.

4. METHODOLOGY

Accessing all the attributes: To access the sub-level at-tributes, we maintain a queue of attributes for a document. Whenever a new attribute is encountered, its children are added to the queue. If an attribute has an array of objects as its value, then we include the attributes of the object in the queue. When the end of a level is reached, ‘-1’ is added to the queue, indicating the same. Once we encounter two consecutive ‘-1’, the end of the document is reached. This is how multi-level accessing is achieved.

Formation of the attribute tree: In the attributes tree, when-ever we encounter an attribute, we search for the node with the same name in the children of the current node it resides. If there is a node with the same name, we move to that node, and this becomes the current node. A new node is formed as the children of the current node; if there is no similar node in the children of the current node, it exists.

Formation of the cluster tree: Whenever a new cluster is formed, we create a node of that cluster and add it to the cluster tree. If the new cluster already shares common attributes with previously existing clusters, then the new cluster is added as a child of the existing cluster. If the new cluster does not share any common attributes, then it is added as the child of the cluster tree root. This is an independent cluster. Fig 1 shows how the clusters are formed. Below is the process of calculating similarity score.

1) while 0:2 threshold 0:9 do
2) Reset attribute Tree
3) Reset cluster Tree
4) Reset Optimal Tree
5) while q n do
6) fcurrentNode traverses all of the attributes in the attributeTree
7) for k in currentNode:children do
8) if node already exists in attribute tree then
9) currentNode = currentNode:children
10) matchCount = matchCount + 1
11) else
12) if matchCount(threshold clusAvgLen) then

© 2021, www.IJARIIT.com All Rights Reserved
The formula to calculate the similarity score is similar to the f1-score. During experimentation, we found that considering the inverse of X can lead to biased results, which shows high similarity scores for thresholds 0.2 and 0.9.

\[
2 \times \frac{\prod_{i=1}^{n} (X_i + 1) \ast (1/Y_i)}{\sum_{i=1}^{n} (X_i + 1) \ast (1/Y_i)}
\]  

**5. LIMITATIONS**

The limitation of our algorithm is that it doesn’t work for duplicate keys on the same level since the keys that are taken from the level are given a unique key. Hence if there are any duplicate values, it creates a dictionary with duplicate values, and it leads to a key error in the dictionary. The other limitation that is similar to the above is the unique values in the array must exist since the same functions are used to solve the array problem. The format of JSON should be in the standard JSON format of the original dataset. No other data structures other than arrays, objects and fields are allowed in the dataset.

This work has not been done by any of the research communities that are working on the clustering of the documents. So, this is an addition to the research space. The algorithm was compared with the existing Weka algorithm for various standards. According to our evaluation measure using the F1-score, the Weka was also tested with multiple data sets and was compared with our algorithm. The results proved that our algorithm fared better than the d-tree model due to the high time complexity and decision making in the d-tree Model. Since the model is very sophisticated at the fundamental level, it creates a very complex tree which is really not what our algorithm does and hence a better result output. The following images compare the results of the d-tree and our algorithm on the same data set.

![Cluster Formation](image1.png)

![Efficiency Result](image2.png)
6. CONCLUSION

On running the algorithm and completing the clustering of the documents, we got to know that the evaluation methods that need to be used are to be different from that of the previous algorithms. So as mentioned in the previous slides, we built an evaluation measure for the clustering section. This method and also the existing methods were used to evaluate the clustering of the documents. In a previous paper where they used d-trees to develop the clusters, the score was primarily used to evaluate, but there were also many other methods that were used along with it. So, with reference to [2], the algorithm that they have used is able to retrieve records and data only up to the first level, that is, the parent nodes only. Hence their algorithm uses a d-tree type approach which is much less complex to do since the number of accesses is very less. In our case, we have tried to access all the existing levels and sublevels to maintain a better clustering method and more detail.

Comparison between documents. This allows the user to have a better understanding of the type of attributes present in the data and also helps them understand the variance in the dataset. The algorithm that we have built works with different types of data structures too. Since JSON has the feature to contain arrays, we have added the ability to access the arrays, attributes in the arrays and also the dictionaries in the arrays (records). This allows a wider ability when it comes to the clustering of documents. Also, this brings in new variances between documents that contain different arrays-set and array attributes.

7. REFERENCES