Information Extraction using Incremental Approach

T.Ramesh Chary¹, N.Naveen Kumar²
¹(M Tech, Computer Science, School of Information Technology (SIT)/Jawaharlal Nehru Technological University, Hyderabad, AP, India)
²(Assistant Professor, Computer Science, School of Information Technology (SIT)/ Jawaharlal Nehru Technological University, Hyderabad, AP, India)

ABSTRACT: Data mining is playing vital role in text extraction as now a day’s large amount of data available in scientific research, biomedical literature and web data. Data retrieval using existing approaches use sequential approach to process the data. It is suitable for one time processing whereas using this approach performance will prunes whenever the new data is added to the existing information we need to reprocess the entire data to perform extraction and it consumes large amount of time as same the initial time of processing. If at all there is any frequent modifications in the existing data, it will require large amount of time to reprocess. This scenario will be repeats same even new extraction of goal is required for the same existing data. There is a high demand in the information extraction but available method such as UIMA and GATE performs IE by file based approach will not use any relational database in the extraction process. Key challenge of data extraction for incremental data, we need to identify which part of the data is getting affected by the change of any component or goal. To achieve this large corpus data will be stored using special type of data storage and optimized queries for data retrieval. It requires more storage compare to existing approach but now a day’s storage size not a key requirement. New approach also introduces automated query generation based on available input data for efficient performance. This method will reduce ninety percent of processing time whenever there is any modification of data comparatively to existing approach.

Keywords – information extraction, data mining, incremental extraction, PTDB, PTQL

1. INTRODUCTION

Now a day’s there is high demand in information extraction whereas extracting using traditional approaches uses pipe lining process. It use files for processing that are suitable for the one time process and we need to process entire data whenever any modification occurs in the available data. Existing approaches like UIMA, NLP and GATE use sequential process and requires same time as close to initial process and has become more worse when ever data is changed frequently. This method stores only final outcome of the results. To avoid this we are proposing the new method of paradigm, where it uses special type of queries and databases. In this approach we store intermediate results which are extracted in process. This approach will increase the storage space but decrease the processing time whenever there is a change in the existing data. The modified data will be stored in appropriate place using suitable queries like insertion and deletion in place of stored data. To retrieve the required information from the corpus data a special type of query language is required to extract information in efficient way we need to generate automated queries depending upon the available input data. Therefore It will reduce the processing time by using dynamic query generation.

For this approach it mainly uses two phases.

1. Data storage
2. Data retrieval

Data storage

Processed data will be stored in the parse tree where the data is stored according to the class of entities and it also stores syntactical dependence between the entities. This database is called as Parse tree data base (PTDB)

Data retrieval

Data retrieval is done by applying queries to PTDB. These queries are optimized and specific task related which narrows the focus /concentrate the goal. These queries uses special type language called as Parse Tree query language (PTQL) where these queries are converted to general SQL and get the required result.

2. PROBLEM STATEMENT

Normal user wants to retrieve the information using simple keyword based query that routes more relevant fulfilling following requirements,
1. Information extraction from large corpus of data using simple key words based search as well as special query language.

2. Data retrieval time should be minimized and accurate i.e most relevant documents to be retrieved from the data.

3. Generation of dynamic queries based on the different types data.

4. Minimise the reprocessing time for any modification of existing data or any new type of extraction in the existing information.

5. Increase of Efficiency by utilizing semantic and syntactic analysis and word to word dependencies

3. INFORMATION EXTRACTION

In this we discuss about extraction approach and modules that are used get the required data. Data will be extracted in two phases i.e 1. Data storage 2. Data extraction

Data storage phase, in this phase the given input data is parsed split into different classes according to parts of speech and these classes are linked using link grammar. This can be achieved by using Parse tree data base. It has two components Constituent tree and linkages.

Constituent tree is syntactic tree where each node of tree is represented by parts of speech and each Leafs corresponding to words in the sentence. final leaves which are in lower level of the tree will contain actual information of the words. Linkages will represent syntactic dependency between the words.

Documents are represented as a hierarchical tree representation called parse tree of a document and collection all of these document parse trees collectively called as Parse Tree Data Base. Each document is split into sentences where these sentences further divide into different sections according to their nature and meaning. Finally each word of the sentence will be tagged according to the Parts of speech. Each division in document will be represented in form of nodes of tree. Final words which cannot be divided further will be at the end of the tree. If any word which does not belong to any parts of the speech will be represented as UNKNOWN. Finally word to word dependencies will be represented by linkages.

Example of Parse tree for the sentence ‘RAD53 positively regulates DBF4’ shown in below figure

Inverted index is used to get more accurate results from the PTQL queries in a faster way. Inverted index will have columns, Document id, Sentence id, Sentence and Sentence_format where Doc id is the unique numeric number given for each document, sentence id is the unique number in given Document id. Sentence format is how it structured.

Example for inverted index

Once PTDB is built we need to get the results according to the requirement by using Parse Tree Query Language Which is a general query language where we have option to get more specific/concentrative results by giving more specific options.

The format PTQL query is

<pattern> : <link condition> : <proximity condition> : <return type>

Tree Pattern: Describes the hierarchical structure and the horizontal order between the nodes of parse tree. Uses two types of axis-Vertical axis, Horizontal axis

Link Condition: Describes the linking requirements between nodes. x<link>y is a link term

Proximity Condition: Used to find words that are within a specified distance.
m[x1, x2, ..., xn] is a proximity term

**Return Expression:** Defines what to return.

<var>, <attr> is a return expression

Tree pattern and return expression are compulsory where as link condition and proximity condition are optional. In place of these optional parameters we can simply place ‘.’

**Parse tree query evaluation**

1. Translate the PTQL query into a filtering query.
2. Use the filtering query to retrieve relevant documents D and the corresponding sentences S from the inverted index.
3. Translate the PTQL query into an SQL query and instantiate the query with document id d -> D and sentence id s -> S.
4. Query PTDB using the SQL query generated in Step 3.
5. Return the results of the SQL query as the results of the PTQL query.

In step 2, the process of finding relevant sentences with respect to the given PTQL query requires the translation of the PTQL query into the corresponding filtering query.

We performed experiments in finding the time performance of the evaluation of PTQL queries, as well as experiments to illustrate the amount of time saved in the event of change of an extraction goal and deployment of an improved module.

**Flow diagram**

The above diagram illustrates the sequence diagram where data file is stored by selecting from the user. Dictionary is applied to identify the classes of each word. Once classified tree is created according to the classes and word to word dependencies linkages will be applied. These tree is stored in database using inverted index. Data can be searched from the Database using simple key words.

**CONCLUSION:**

Existing framework is more vulnerable if any small modification in available input data. It takes same amount of time as initial processing. Hence more expensive. By utilising the incremental approach we can decrease reprocessing time by storing intermediate data which is extracted during process. Parse tree database uses the Inverted index mechanism for storing data. PTQL queries are designed in such way to get more efficient results in faster way using inverted index mechanism.

**REFERENCES**


