1. **ABSTRACT**

In the past decade, there has been a tremendous increase in the amount of data generated every day. Due to this huge influx of raw data, it is difficult to provide a comprehensible design that is highly beneficial for data storage, analysis, and prediction. Constructing relational databases for structured data has been a methodical process with satisfactory outcomes. However, the same concepts cannot be directly applied to store unstructured, textual data.Text data requires pre-processing before it can be easily stored in a columnar fashion in a relational database. Once formulated,the RDBMS is helpful in constructing cubes with text data in atomic form. However, this approach is not very helpful in representing hierarchically structured data. In this paper, a way of implementing ontologies of hierarchies in the formation of an text cube is proposed along with its performance evaluation.

* 1. **Keywords**

OLAP, Text Cube, Ontology, Hierarchy, RDF, SPARQL, Protégé, RDFLib, TextBlob

# INTRODUCTION

In this section, a brief introduction to the concepts being implemented and how they coalesce to derive the input requirements of a data cube are outlined. Section 2.1 introduces OLAP Cube, the data warehouse concept taken into consideration for the purpose of this project. Section 2.2 introduces Ontology, the concept used to study the data and the requirements. Section 2.3 presents a scenario to understand user requirements and how to extract input data for the data warehouse into consideration.

**2.1 OLAP Cube**

Nowadays most of the data generated is in a complex format, like: text, videos, pictures, etc. Simple data structures and implementations cannot store or analyze this data easily.

A union of conformed data marts can generate many different types of data cubes. An OLAP cube is a multidimensional data structure that represents a database. It can be used for data warehousing and online analytical processing applications. If the number of dimensions is very high, it is called a hyper cube. The dimensions of the cube represent the attributes of the data. It facilitates easy access to data at various granularity levels and can help in analyzing data from various views. Some of the basic uses of an OLAP cube include generating summary statistics. Fig 2.1.1 shows how data is transformed from raw form into reports with the help of information retrieval algorithms to extract required information and store it into an text cube.

Text Data

IR Algorithms

Text Cube

Ontology- Product

Customer

Address

Time

Calendar

Reports

Fig 2.1.1 Process flow of Text cube construction

Dimensions

A text cube is a cube based on text data. It is a form of data mart. It is an efficient way of storing unstructured data such that information retrieval operations can be performed. It can, further, be used for data analysis and pattern detection. A hierarchy of text is used to specify the granularity and relationships among the terms from the text data. Suppose we have a collection of documents, D, and each document has N dimensions. Each row in database DB has (a1, a2, a3, ..., an, D) values. A document can have multiple terms from set W = {w1, w2, w3, …, wm}. A term hierarchy can have various terms from W defining their levels in the hierarchy, T = {v11,.., v21,..,v31,..….} as shown in Fig. 2.1.2.

v31={w1,w2,w3,w4,w5}

v21={w1,w2,w3} v22={w4,w5}

v1={w1} v2={w2} v3={w3} v4={w4} v5={w5}

[iPhone] [good] [Apple] [bad] [size]

Fig2.1.2 Term Hierarchy, T

To form a text cube that can be used for reporting and analytical purposes, Twitter data has been used for this research. Twitter is a social networking, microblogging website. A large amount of tweets are generated everyday where people express their sentiments, or are generating mass awareness about a news. A OLAP cube realized from the text cube is a very efficient way of analyzing twitter data in order to study human behavior and sentiments. It renders data aggregation and analysis in various contexts through measures such as slicing, dicing, and drilling.

**2.2 Ontology**

An ontologyis a representation of the concepts, relations and rules for a particular area of business information, irrespective of how that information may be stored as data. In other words, it is a specification of conceptualization. These are analysis theories with a hierarchical approach to the concepts within a domain. An ontology is machine readable. OWL is an ontology language that is used to represent the details of data present on the web. It follows W3C standards for Semantic Web.

An ontology is made up of classes and the relationships between these classes along with certain properties to identify them (Fig. 2.2.1). Once the requirements of a project are gathered, ontology can be applied in different ways. The simplest way would be to use same ontology to construct all the sources. Another method would be to use different ontology for different sources. Yet another implementation would be to combine the two forms where each source is based on a different ontology but are linked together on the basis of a common ontology.

Resource Description Framework (RDF) generates a meta-data model with a logical structure to represent the data from the web. It is made up of information units called triplets. Each triplet has 3 values - a subject, a predicate and an object. The subject, s, is referred to as the resource, with a specific property (the predicate, p) and a value (the object, o). The data on the web comes from various resources and can be identified by their URIs – Uniform Resource Identifiers. The RDFs are made up of these URIs where a subject can either be a URI or a blank node indicating an anonymous resource. Similarly, a predicate can be a URI pointing to a relationship. And, the object can also be a URI, a blank node or a Unicode string literal. The triplet form of the RDFs can be expressed as a directed graph with each subject and object being a node, while the directed edges between these nodes depict their properties. The data from a RDF structure can be queried using SPARQL (SPARQL Protocol and RDF Query Language) query language. It looks for the matching triplets to form a RDF graph.

Domain

‘has part’

relation

‘has part’

relation

Fig. 2.2.1 Ontology of a domain with its classes, subclasses and relations

Class

Subclass

A more sophisticated design of ontology can be generate using Protégé by Stanford. Al the classes and subclasses can be defined along with their properties and relations. Fig. 2.2.2 shows the class hierarchy of Apple Products where everything starts at class called ‘Thing’. It has two subclasses defined – Apple and Features, They further have multiple subclasses of their own.

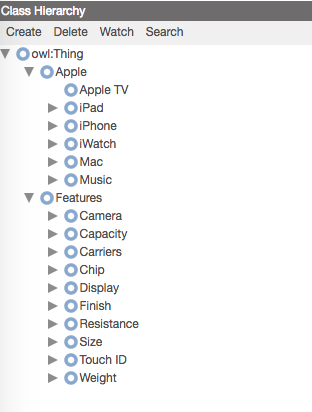


Fig.2.2.2 Two level hierarchical structure of Apple products in Protégé

**2.3 Understanding User Requirements**

Before using data related to tweets collected from Twitter for business intelligence systems, it needs to be stored and processed. User requirement plays an important role in text cube design. It influences every decision throughout implementation of the cube.

For the purpose of this paper, the user requirements outlined are:

1. Product level in the hierarchy
2. Sentiment related to a tweet

Apple products are studied and an ontology is constructed to define product levels. Protégé is used for constructing ontologies and these ontologies can be accessed and queried in python using RDFLib and SPARQL. Furthermore, the extracted data is analyzed to get derived information such as sentimental values like polarity and subjectivity of a tweet. TextBlob library in python is useful in implementing natural language processing and extracting sentiments. These steps are performed before data is stored in a text cube (Fig. 2.3). For the purpose of It is ensured that all the user requirements are met and the necessary data is ready to be loaded into the text cube.

Staging Area

Text Cube

Ontology

Protégé

RDFLib

SPARQL

TextBlob

Fig. 2.3 Implementation of Ontology to meet user requirements.

1. **PROBLEM**

**3.1 Problem Statement**

It is difficult to analyze the performance of a text cube that uses ontologies in order to store unstructured text data. In this paper, the implementation and performance evaluation of such a text cube for twitter data related to a particular domain will be performed. The main aim is be to match tweets with their correct level in the ontology hierarchy.

People tweet about a lot of topics on twitter to share their viewpoints, reviews, and information, in general. Apple trademark products form one such popular domain and people very often mention them in their tweets. This makes Apple products a good domain to work on for the implementation of this project. All the products and their features can been divided into classes and sub-classes. Ontology is an excellent way of representing this hierarchy of classes. However, these hierarchy levels aren’t of much use to the analytics team of the company until they can relate the hierarchy with other important attributes of the data. They should be able to detect the level of hierarchy in a tweet along with other related information to be able to utilize Twitter data for analysis purposes.

**3.2 Significance**

Ontologies have played a significant role in structuring and analyzing a domain’s knowledge. It can be used to study the data at various levels of granularity. Due to the tremendous increase in the data being generated every day, there is a dire need to store this unstructured data in the most efficient way. A text cube is one of the best approaches to store data in a multi- dimensional cube. A combination of text cube and ontology related to a particular domain can be very beneficial for storing, evaluating and reporting of text data. OLAP cubes can then be generated based on user requirements. For the purpose of this project an ontology of twitter data obtained for Apple products has been constructed to form the text cube. This text cube is useful to the analysts who want to study factors like trends, sales, popularity, and success of the products.

**3.3 Goals**

The goal of this paper is to construct a text cube for unstructured text data about Apple products from Twitter with its ontology being used as basis for one of the many dimensions to ensure that each tweet matches the correct level in the hierarchy. Also, the performance of the system built has be evaluated.

**3.4 Objectives**

In this paper, the objectives is to:

* Design an ontology for a particular domain from twitter.

The selected domain is Apple trademark products.

* Construct a text cube for the data related to that domain by matching it with the hierarchy levels defined in the ontology.

Thus, the text cube will have tweets, hierarchy levels, and other related attributed of the tweets.

* Evaluate the performance of the cube in terms of query performance and resource utilization.

1. **RELATED WORK**

The following research papers were studied to get an insight on the various concepts and methodologies taken into consideration for this paper.

Data cubes are used in order to manage Online Analytical Processing of large amount of data. In a very interesting study (Lin, Ding, Han, Zhu, & Zhao, 2008), the use of OLAP text cube was suggested. The goal was to create a multi- dimensional cube for storing huge amount of complex data. The main information retrieval techniques were based on term frequency vectors and inverted indexes. It gives a fair understanding of how text data can be stored to generate OLAP cubes.

In addition, it is essential to comprehend how ontologies work and can be used for text cube formation. Garcia et al. came up with a data cube algorithm to build a cube based on ontology, called CUBO(2012). All the cubes at a particular level in an ontology were stored together, while each level is placed in a hash table. However, the performance of this cube has been an issue under various circumstances, such as while using unbalanced tree ontology, or using very large datasets, etc.

The following two papers helped in better understanding of how text dimensions can be formed and utilized in a text cube.

An OLAP text cube called CXT- Cube was proposed that included a textual analysis part(Oukid, Asfari, Bentayeb, Benblidia, & Boussaid, 2013). In this scenario, vector space model has been implemented and each dimension comprehends a contextual factor. Another remarkable result was the use of an aggregation operator called OLAP Rank in order to rank the documents. Yet, scalability has been an issue here too.

Another project, called EventCube (Tao et al., 2013), was funded and supported by various government agencies. It also used the concept of building a text cube from the free text data such as information from news, journals, blogs, etc. to perform data mining and machine learning algorithms. This has be helpful in pattern detection, querying, and visualization of data in multi- dimensional structures.

To get an intuition about the evaluation of a text cube, a recent paper by Suan Lee et al. (2014) that talks about new performance measuring parameters has been studied. It has been discussed that the use of Inverted Index (IV) is not ideal as it is difficult to calculate and requires large space. They suggest the use of Term Frequency Inverse Document Frequency (TF-IDF) or Language model (LM) for more accurate and efficient analysis of data.

1. **IMPLEMENTATION**

The aim of this paper is to use ontology to construct the basic structure of a text cube. The domain considered for building ontology is Apple Trademark products and their features. The main reason for selecting this domain is a lot of people talk about and share reviews about Apple products. A proper analysis of these reviews can help the company as well as the potential buyers.

**5.1 Data Source**

The data considered for this experiment and analysis is from Twitter and has been stored in MongoDB database. It has been extracted from Twitter API using Consumer API key and Consumer Secret API key. The keys can be generated in Twitter Application Management.

The extracted data is stored in database called ‘AppleProductReviews’ in MongoDB. It consists of 3 collections – Fact, Location, Time (Fig.5.1.1).

****

Fig.5.1.1 Data from Twitter is stored in MongoDB collections

Fact collection has following keys in each object: \_id, uid, lid, Text, category, tid, cid

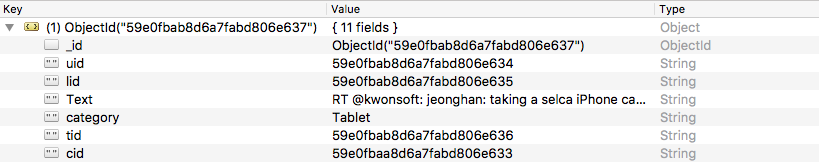
****

Fig.5.1.2 A sample object with key – value pairs from Fact table

Location collection has following keys in each object: \_id, Location

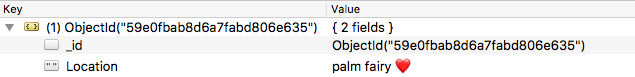


Fig.5.1.2 A sample object with key – value pairs from Location table

Time collection has following keys in each object: \_id, Time, TimeZone

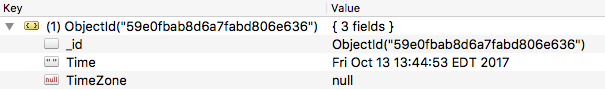


Fig.5.1.4 A sample object with key – value pairs from Time table

**5.2 Protégé – Ontology Construction**

Protégé by Stanford (“protégé,” n.d.) is an open source ontology editor that provides a graphical interface to construct ontologies. It can define classes and their properties. For this experiment, a basic ontology of Apple products has been built. With Apple being one of the main classes, the sub classes under Apple are Apple TV, iPad, iPhone, Mac, etc. These classes are at level one of the hierarchy. At level two, more sub- classes of products related to a particular category are defined such as iPhone has its various models – iPhone 7, iPhone 7 Plus, iPhone 8, etc. (Fig.5.2.1). Similarly, the two levels of features have been defined separately and are linked to the products via properties section of each class (Fig5.2.2).

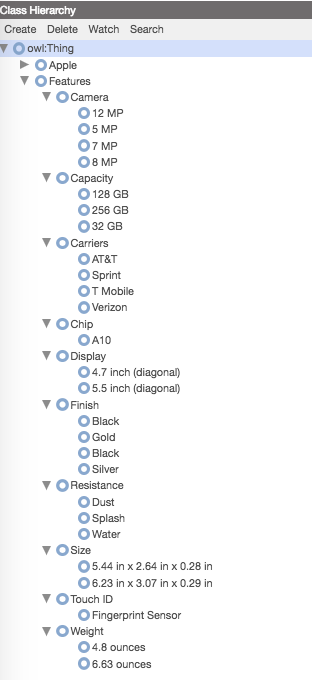
 

Fig.5.2.1 Level 1 and level 2 hierarchies of Apple class.

Fig. 5.2.2 Level 1 and level 2 hierarchies of Features class.

The constructed ontology can be saved as an OWL file. Notice that the owl structure starts with a label called ‘Thing’. Following Fig.5.2.3 shows a sample of the OWL file generated for the Apple product ontology – root-ontology.owl.

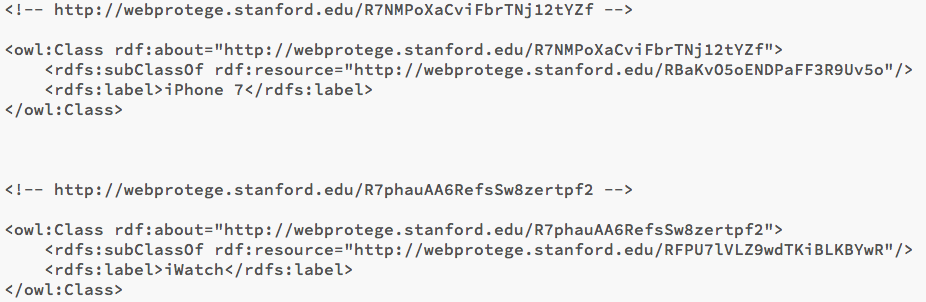


Fig. 5.2.3 Sample code from ontology OWL file

**5.3 Sentiment Analysis and Level Detection**

The following steps are to calculate the sentiment related to each tweet and to identify the level in hierarchy at which the product from the tweet falls. Sentiment analysis is helpful in checking the popularity of products among people.

Section 5.3a - Reading ontology using RDFLib library

Section 5.3b – Using SPARQL to query ontology

Section 5.3c – Extracting elements from different levels of ontology

Section 5.3d – Using PyMongo API to connect to MongoDB

Section 5.3e – Using TextBlob to calculate sentiments related to tweets

Section 5.3f – Matching and determining hierarchy level in tweets

**5.3a RDFLib**

Rdflib (“rdflib 4.2.2 — rdflib 4.2.2 documentation,” n.d.) Is a python library that can parse RDF data and represent it as graphs. An RDF graph consists of RDF triples - a subject, a predicate, and an object. The data in graph (g) with values for subject (s), predicate (p) and object (o) can be viewed by looping over it (Fig.5.3a):

*for s, p, o in g:*

*print(s,p,o)*

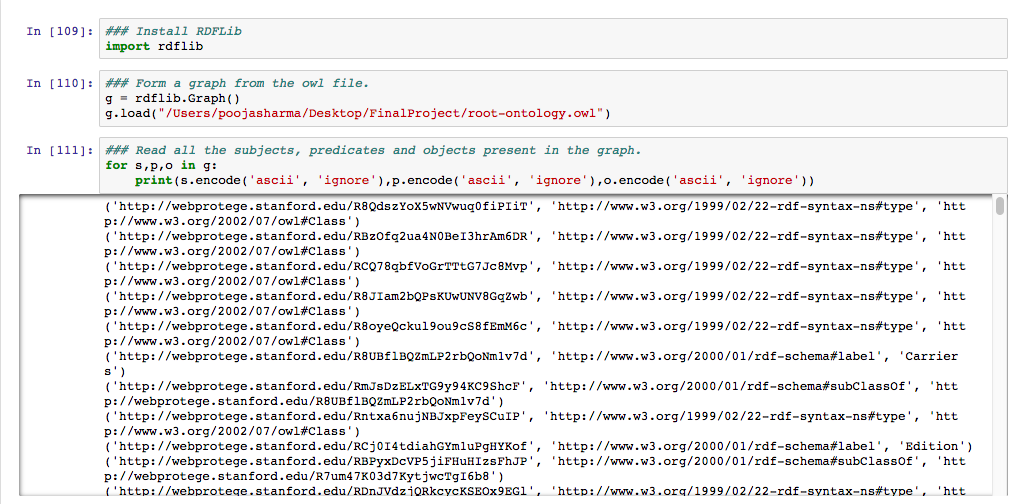
****

Fig.5.3a RDFLib is used to form graph of OWL file

**5.3b SPARQL Queries**

SPARQL queries can be evaluated against a graph. They are used to analyze the data. SPARQL queries are run on graph produced in Section 5.3a to read classes and sub-classes present in the ontology. Some of the queries run on the graph (g) are shown in Fig.5.3b.

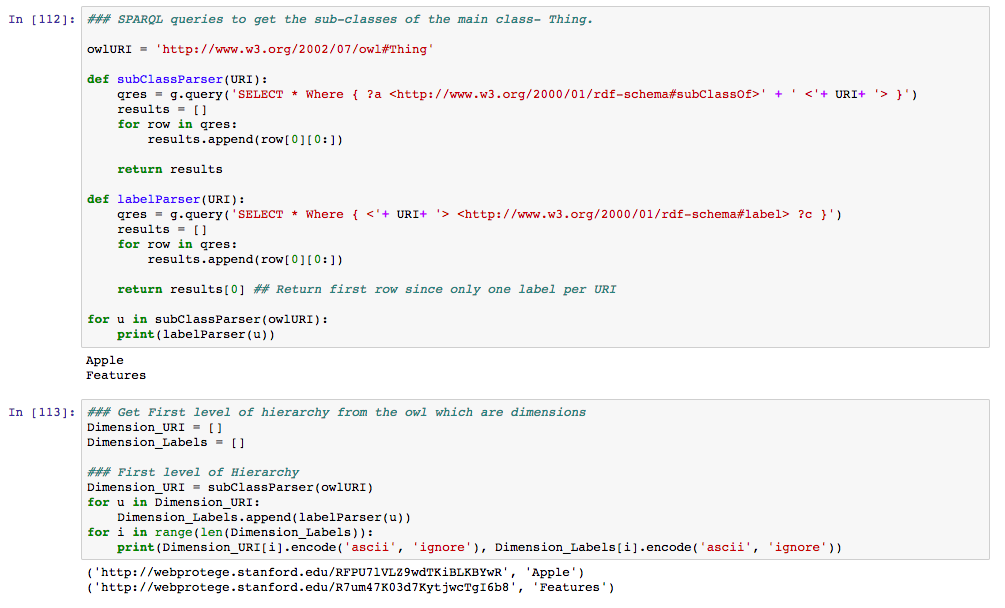


Fig. 5.3b Extraction of classes and their URIs under the main class – ‘Thing’

**Statement** - To select the URIs of all the sub-classes of the main class – Thing.

**Query** - *SELECT \**

*Where { ?a <http://www.w3.org/2000/01/rdf-schema#subClassOf> <http://www.w3.org/2002/07/owl#Thing>}*

**Output** –



The above query returns all the values for subject ( ‘?a’ – a variable) where predicate is ‘*http://www.w3.org/2000/01/rdf-schema#subClassOf’*  and object is ‘*http://www.w3.org/2002/07/owl#Thing’.*

**5.3c Extraction of Elements from Different Levels of Ontology**

Using the unique URIs, all the sub-classes of a class are extracted at each level. The retrieved values are stored in arrays.

* Apple First Level



Fig. 5.3c.1 Extraction of first level of sub classes under ‘Apple’ class

* Apple Second Level



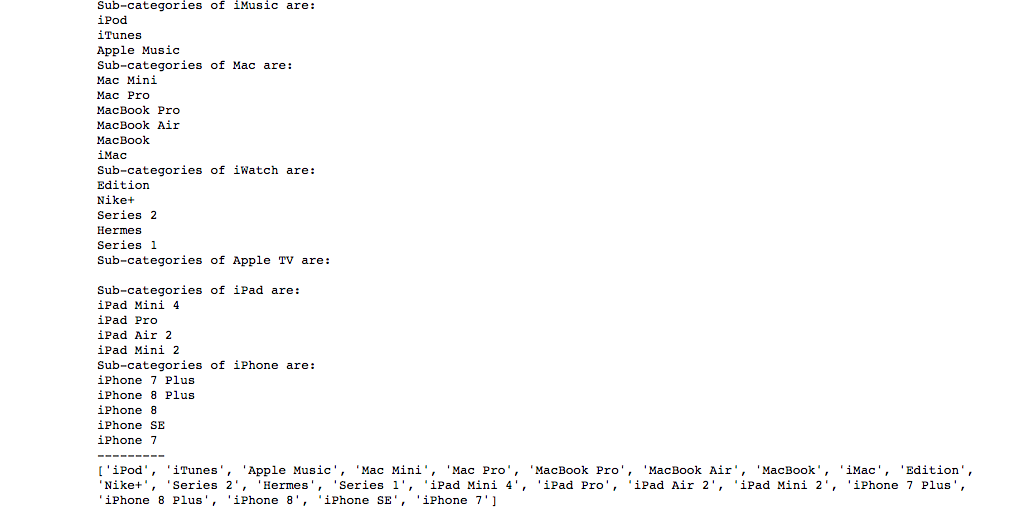


Fig.5.3c.2 Extraction of second level of sub classes under ‘Apple’ class

* Features First Level

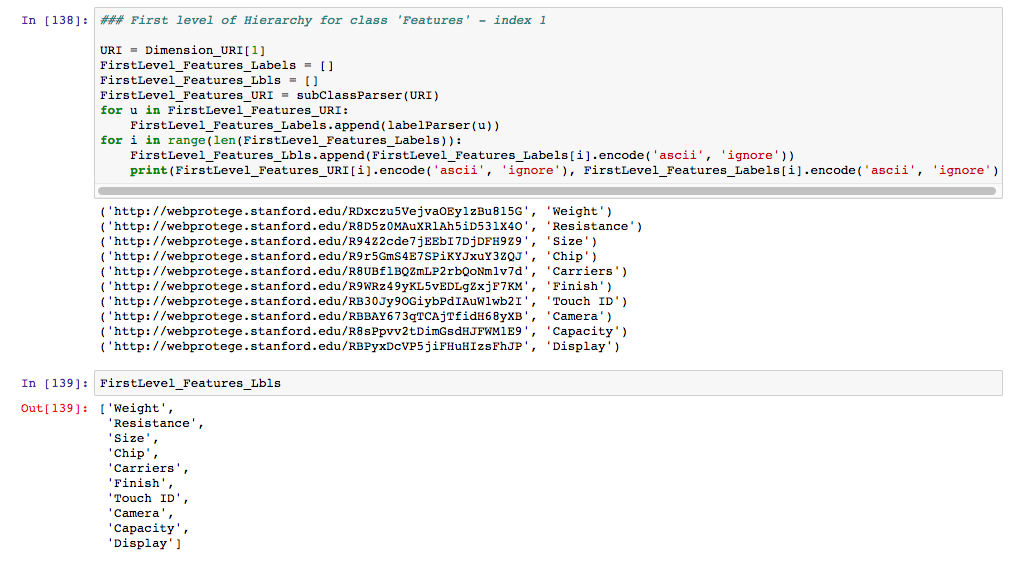


Fig.5.3c.3 Extraction of first level of sub classes under ‘Features’ class

* Features Second Level



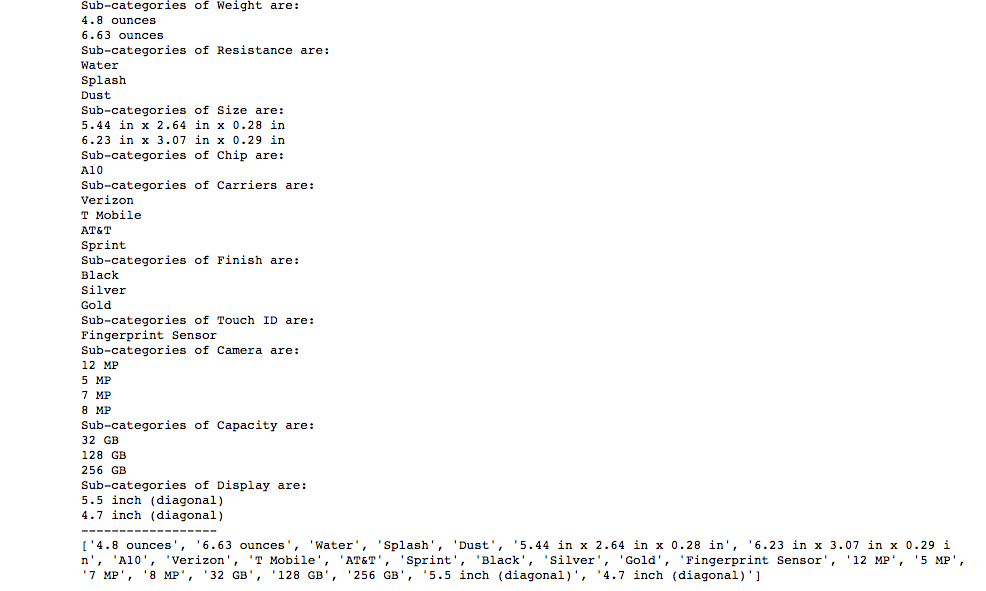


Fig.5.3c.4 Extraction of second level of sub classes under ‘Features’ class

**5.3d PyMongo API**

PyMongo distribution for python (“PyMongo 3.4.0 Documentation — PyMongo 3.4.0 documentation,” n.d.) has been used to stage the data from MongoDB database as shown in Fig.5.3d.



Fig.5.3d Connection set-up with MongoDB

**5.3e TextBlob**

TextBlob (“TextBlob: Simplified Text Processing — TextBlob 0.12.0 documentation,” n.d.) is another python library used in this project. It processes text data to perform natural language processing tasks such as word extraction, sentiment analysis, etc.

To see what people think and talk about the company – Apple and its products, sentiments related to the tweets must be calculated. It can be beneficial in analyzing questions like: if a product is popular or not, in which region the popularity is more, what makes a product less/ more popular, how can an existing product be improved.

To calculate the sentiment related to a tweet, TextBlob provides a property – sentiment, that returns a tuple Sentiment(polarity, subjectivity). Polarity ranges between (-1.0, 1.0) where -1.0 is negative sentiment and 1.0 is positive sentiment, and subjectivity ranges between (0.0, 1.0) where 0.0 is very objective and 1.0 is very subjective. The obtained values for polarity and subjectivity for each tweet are then stored in the MongoDB database.

In fig.5.3e.1, for every tweet text obtained from MongoDB, we insert two new keys called Polarity and Subjectivity in MongoDB and update the values with calculated polarity and subjectivity, resp.

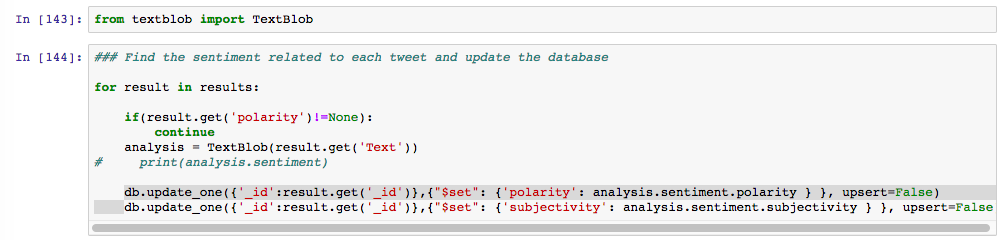


Fig.5.3e.1 Determine sentiments - Polarity and Subjectivity of the tweets and update in MongoDB

MongoDB Object after update:

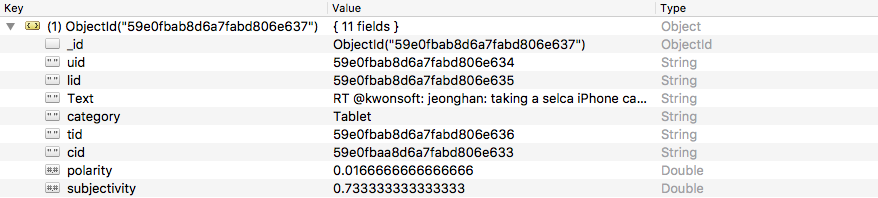


Fig.5.3e.2 Polarity and Subjectivity values for a tweet are added in MongoDB

**5.3f Hierarchy Level Determination**

Match the terms present in first and second levels of ontology with words in the tweets. If found, update MongoDB with the found term against hierarchy level.



Fig.5.3f.1 Find occurrences of First and second level words of ‘Apple’ class in tweets and update MongoDB

To match the elements in Apple class, the words from Apple first level array and Apple second level array are matched with words in tweet text (Fig.5.3f.1). If a match occurs, the MongoDB data is inserted with a new key denoting the class level and the found word is updates as value to the new key. Similarly, words in Features class arrays are matched with words in tweet text and MongoDB updated with new values, if match is found (Fig.5.3f.2).



Fig.5.3f.2 Find occurrences of First and second level words of ‘Features’ class in tweets and update MongoDB

MongoDB Object after update:

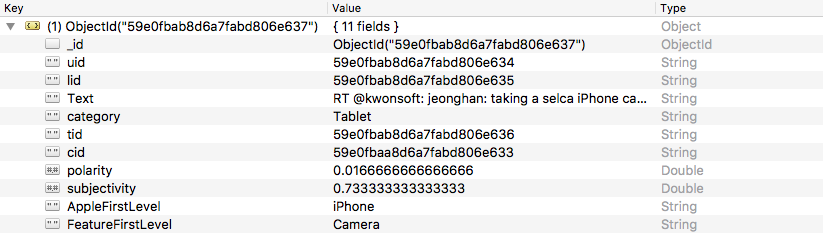
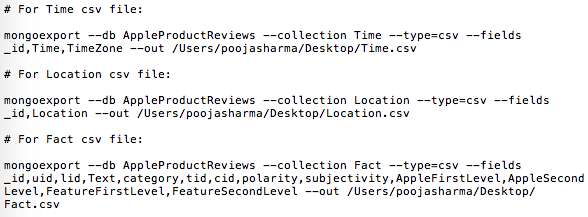


Fig.5.3f.3 Apple and Feature level values found in a tweet are added in MongoDB

**5.4 Save MongoDB collections into .csv files**

The required collections – Time, Location, and Fact – are exported and saved as .csv files. These csv files are used to form the text cube.

The following commands are executed in the console to get the csv files:



**5.5 Cleaning of Location.csv**

The location data from twitter for the collected tweets has lot of anomalies. Below are the samples of such bad data and the process followed to clean it:

**Sample 1** – Random data (special characters and emoticons) in location column with no meaningful value.







**Solution** - This unclean data has been replaced with ‘No Info’.

**Sample 2** – Location is not a geographical location.



**Solution** - Even after removing special characters, the location isn’t actually a real geographical place. However, such data after cleaning has been maintained in the file. Processing of such locations must be discussed with the client to ensure the required format.

**Sample 3** - Location includes a geographical place along with special characters.



**Solution** - The special characters are removed and only the location is retained.

**Sample 4** - Data is neither in correct format nor have location values.



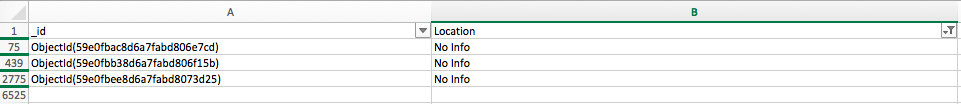
**Solution** - The field format is set to text and ‘No Info’ is inserted.

**Sample 5** – The location field is blank.



**Solution** - ‘No Info’ is inserted in blank cells.

Thus, the updated location file has ‘No Info’ value in the location column for the rows with anomalies or blank values.



**5.6 Text Cube Construction in MySQL**

Using MySQL, a star schema for the cube is designed. It consists of four dimensions and a fact table. Each dimension has a surrogate key which is used as a foreign key in the fact table. The fact table consists of facts like Polarity and Subjectivity values of the tweets.

**Product Level**

**Time**

**Location**

Facts

**Text**

Fig. 5.6 Text cube structure using Ontology

**5.6a Text Cube**

A text cube called – OntologyDataMart, is constructed. The dimension and fact tables import data from csv files – Location.csv, Time.csv and Fact.csv.

ProductLevel and Text dimensions are populated from Fact.csv as the actual text field and the updated Apple and Features Levels are a part of Fact collection in MongoDB.

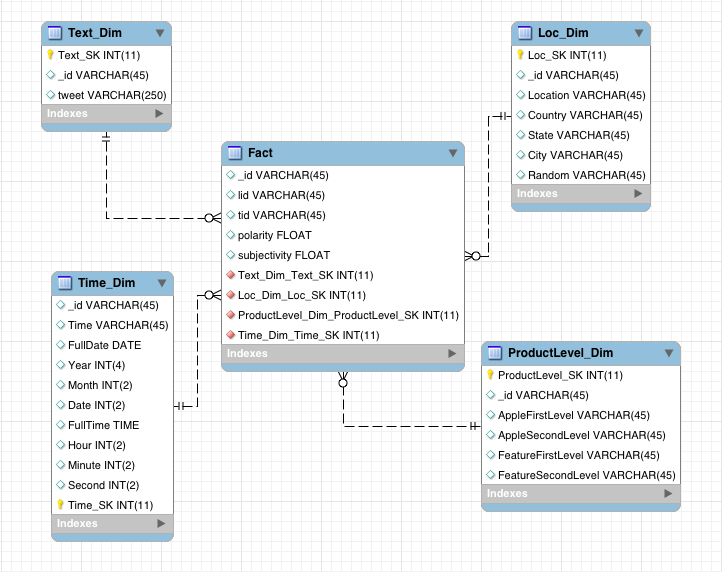


Fig.5.6a Model designed for the text cube

**5.6b Dimensions**

**Dimension 1: Time Dimension\***

Query – SELECT \* FROM OntologyDataMart.Time\_Dim;

Output –



Fig.5.6b.1 Date dimension table structure

**Dimension 2: Location Dimension**

Query – SELECT \* FROM OntologyDataMart.Location\_Dim;

Output –



Fig.5.6b.2 Location dimension table structure

**Dimension 3: Text Dimension**

Query – SELECT \* FROM OntologyDataMart.Text\_Dim;

Output –

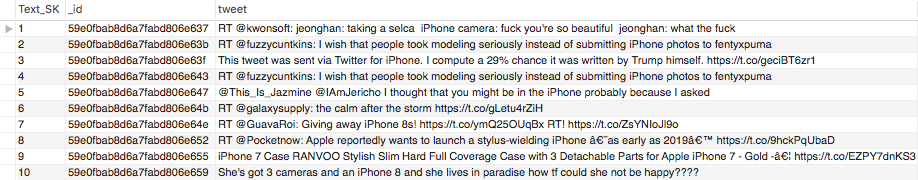


Fig.5.6b.3 Text dimension table structure

**Dimension 4: ProductLevel Dimension\***

Query – SELECT \* FROM OntologyDataMart.ProductLevel\_Dim;

Output –



Fig.5.6b.4 ProductLevel dimension table structure

Note(\*) - The Date dimension has rows where TimeZone values are null. Similarly, ProductLevel dimension should have columns with values from Ontology levels that exist in the tweet per row. However, such data can lead to a large number of null values in columns where no word from a level is found. This is because only certain attributes are present for a subset of the dimension data. Null values in a dimension can cause a lot of confusion. Therefore, a good way to deal with nulls in a dimension is to replace them with a descriptive value (Thornthwaite, 2003). For example, for rows in Date Dimension where TimeZone is not given and in ProductLevel dimension, words from a level are not found in the tweet, ‘Not Found’ has been inserted.

**5.6c Fact Table**

The fact table consists of all the foreign keys and the measurable facts related to the dimensions. It describes the data warehouse in the lowest possible grain. Therefore, the numeric facts, polarity and subjectivity, are included in this table.

**Grain** – The sentiments associated with tweets about products per hierarchy level per day at a specific location.

Query – SELECT \* FROM OntologyDataMart.Fact;

Output –



Fig.5.6c.1 Fact table structure

-- Table structure for table `Fact`

DROP TABLE IF EXISTS `Fact`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `Fact` (

`Time\_SK` int(11) DEFAULT NULL,

`Location\_SK` int(11) DEFAULT NULL,

`Text\_SK` int(11) DEFAULT NULL,

`ProductLevel\_SK` int(11) DEFAULT NULL,

`\_id` varchar(45) DEFAULT NULL,

`uid` varchar(45) DEFAULT NULL,

`lid` varchar(45) DEFAULT NULL,

`category` varchar(45) DEFAULT NULL,

`tid` varchar(45) DEFAULT NULL,

`cid` varchar(45) DEFAULT NULL,

`polarity` float DEFAULT NULL,

`subjectivity` float DEFAULT NULL,

KEY `Time\_FK\_idx` (`Time\_SK`),

KEY `Location\_FK\_idx` (`Location\_SK`),

KEY `Text\_FK\_idx` (`Text\_SK`),

KEY `ProductLevel\_FK\_idx` (`ProductLevel\_SK`),

CONSTRAINT ` fk\_Fact\_Time\_Dim1` FOREIGN KEY (`Time\_Dim\_Time\_SK`) REFERENCES `Time\_Dim` (`Time\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT ` fk\_Fact\_Loc\_Dim1 ` FOREIGN KEY (`Loc\_Dim\_Loc\_SK`) REFERENCES `Location\_Dim` (`Location\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT ` fk\_Fact\_ProductLevel\_Dim1` FOREIGN KEY (`ProdLevel\_Dim\_ProdLevel\_SK`) REFERENCES `ProductLevel\_Dim` (`ProductLevel\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT ` fk\_Fact\_Text\_Dim1` FOREIGN KEY (`Text\_Dim\_Text\_SK`) REFERENCES `Text\_Dim` (`Text\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

The foreign keys included in the fact table and their referenced tables and columns are shown in Fig5.6c.2:

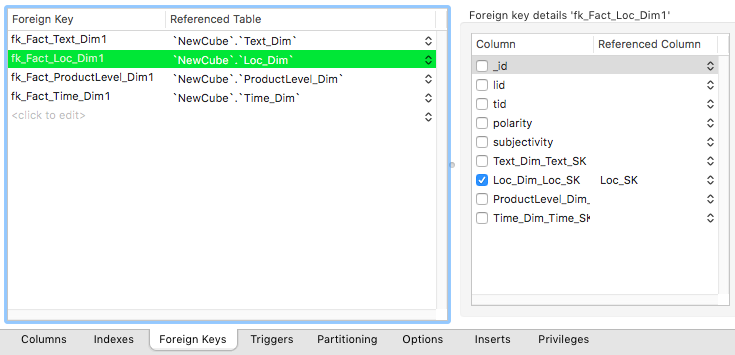


Fig.5.6c.2 Foreign keys included in the Fact table

Note – The foreign keys are prefixed with ‘fk\_’ and the column names associated with them in Fact table and Referenced table are suffixed with ‘\_SK’.

**5.7 Verification of Sentiment Values**

The sentiment values associated with the tweets have been generated using TextBlob library. To verify the values, a simple python program has been written that calculates the sentiments (Fig. 5.7). Given a tweet, it can be classified in categories. The categories have been defined as positive and negative.

Naïve Bayes classifier has been used for sentiment analysis of the text. In this classifier, a text document is represented as a bag of words, that is, an unordered set of words with their position ignored, maintaining only the frequency in the document. Naïve Bayes is a probabilistic classifier. For a document d, out of all classes c, the classifier returns the class which has the maximum posterior probability given the document (“Naive Bayes and Sentiment Classification,” n.d.). Naïve Bayes classifier is based on two assumptions. The first is, in bag of words we assume position of word doesn’t matter. So, a word in 1st, 15th or last position has the same effect. The second assumption is, the probabilities are independent given the class c and hence can be naively multiplied.

Tweets

Naïve Bayes Classifier

Positive

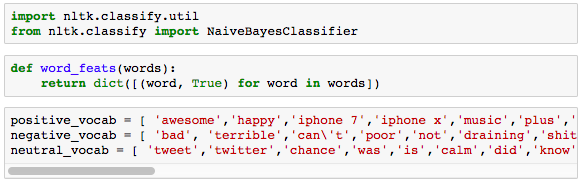
Negative

Prediction

Fig.5.7 Sentiment Analysis using Naïve Bayes

The following steps have been implemented to classify the tweets:

Step 1 – Three classes have been defined – positive, negative and neutral.

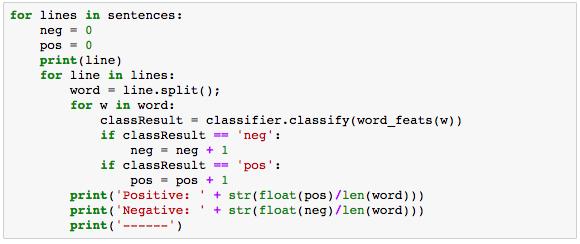


Step 2 – Each word is converted into a feature using bag of words model. Training set is the sum of the three feature sets.

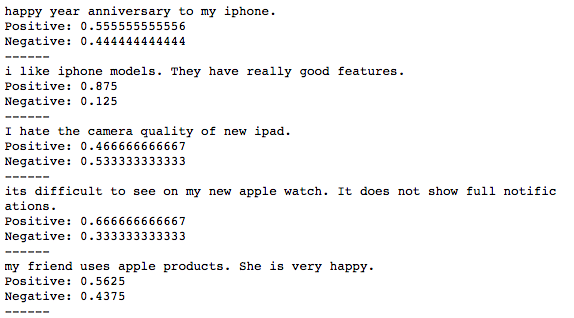


Step 3 – Naïve Bayes Classifier is used to train the classifier and make predictions. A set of 100 tweets are given as input to the classifier. The positive and negative sentiment values for all the tweets are generated.

****

****

Output:



Step 4 – Compare the output of Naïve Bayes classifier with the results obtained from TextBlob Sentiment Analysis polarity values. The following table shows the number of positive and negative records calculated by TextBlob and Naïve Bayes, respectively.

|  |  |  |
| --- | --- | --- |
| Method | Text Blob | Naïve Bayes |
| # Positive Records | 68/100 | 72/100 |
| # Negative Records | 32/100 | 28/100 |

Table.5.7 Number of positive and negative tweets generated by TextBlob and Naïve Bayes

It is, therefore, observed that both the algorithms give very similar sentiment values for the tweets with a marginal fluctuation of 12%. Since TextBlob uses more advanced and sophisticated algorithm, the results from TextBlob have been considered for further analysis of the data.

1. **PERFORMANCE EVALUATION**

Raw tweet dataset of Apple products has been collected from Twitter for this paper. After implementing all the steps described in Sections 5.1 to 5.5, it was converted in a format suitable to form the Text Cube (Section 5.6). As this data has not been used before for implementing any other model, the performance evaluation of the Text Cube is done on the basis of dataset size to study its query performance and hardware utilization. No performance comparison is shown with respect to any existing models.

* 1. **Query Performance**

To evaluate the query execution time of the constructed Text Cube, two approaches have been followed. In the first approach, queries are executed in MongoDB as well as MySQL to observe the difference in execution time. In the second approach, queries on two different sized datasets are executed and execution time difference is recorded.

**Approach 1:**

The main difference between No SQL and Relational database lies in the underlying model. The Text Cube formed in Section 5.6 has a structured star schema model whereas MongoDB No SQL database is schema less.

The following queries are executed on MongoDB as well as MySQL. In case of MongoDB, the database used for query execution is ‘AppleProductReviews’ and the same 5 tables that exist in ‘OntologyDataMart’ used for TextCube construction are generated – Fact, Time, ProdLevel, Location and Text.

**Query Type 1** – Simple Select Queries and their Execution time in MongoDB and MySQL

|  |  |  |  |
| --- | --- | --- | --- |
| MongoDB Query | MongoDB Exec Time | MySQL Query | MySQL Exec Time |
| db.getCollection('Fact').find({}) | 0.004 sec | SELECT \* FROM OntologyDataMart.Fact LIMIT 0, 100000 | 0.006 sec |
| db.getCollection('Time').find({}) | 0.003 sec | SELECT \* FROM OntologyDataMart.Date\_Dim LIMIT 0, 100000 | 0.002 sec |
| db.getCollection('Location').find({}) | 0.005 sec | SELECT \* FROM OntologyDataMart.Location\_Dim LIMIT 0, 100000 | 0.005 sec |
| db.getCollection('Text').find({}) | 0.002 sec | SELECT \* FROM OntologyDataMart.Text\_Dim LIMIT 0, 100000 | 0.003 sec |
| db.getCollection('ProdLevel').find({}) | 0.003 sec | SELECT \* FROM OntologyDataMart.ProductLevel\_Dim LIMIT 0, 100000 | 0.003 sec |

Fig. 6.1a Execution time of select queries in MongoDB and MySQL

**Query Type 2** – Queries with Where clause

|  |  |  |  |
| --- | --- | --- | --- |
| MongoDB Query | MongoDB Exec Time | MySQL Query | MySQL Exec Time |
| db.Fact.find({polarity:{$gt: 0.75}}) | 0.005 sec | Select \* from OntologyDataMart.Fact where polarity > 0.75; | 0.0061 sec |

Fig. 6.1b Execution time of queries with conditions in MongoDB and MySQL

It is observed that MongoDB executes simple queries (Fig. 6.1a) and queries with conditions (Fig. 6.1b) relatively faster than MySQL even though the difference in execution time is not very significant. However, the main aim of text cube is to get insights of data from various viewpoints. This required multi-dimensional table joining. One of the drawbacks of No SQL databases is no joins can be performed on the tables. They are designed to store de-normalized data. There should not be any relationship between the collections and, if required, data must be repeated.

**Query Type 3** – Queries with Joins

A query with joins such as -

*select \* from OntologyDataMart.Fact o*

*join OntologyDataMart.ProductLevel\_Dim p*

*on o.ProductLevel\_SK = p.ProductLevel\_SK*

*where o.polarity > 0.85 and p.AppleSecondLevel ='iPhone 8' LIMIT 0, 100000*

cannot be executed in MongoDB. If the size of dimensional table is very large, merging it with fact table will result in a huge fact table. This can severely slow down query execution.

It is, therefore, concluded that a structured schema can be best implemented and queried using MySQL model instead of MongoDB (No SQL) even though query execution performance of MongoDB is marginally better than MySQL.

**Approach 2:**

As size of data warehouses increases with time, two Text cubes with varying data sizes have been created. Text cube ‘NewCube’ has 9620 records and text cube ‘OntologyDataMart’ has almost double records with count of 19240. The evaluation is done in terms of change in query execution time per table(Fig.6.1e). For this purpose, execution time of query for selecting all the rows of designed dimensions and fact table have been recorded (Fig.6.1c, Fig.6.1d).



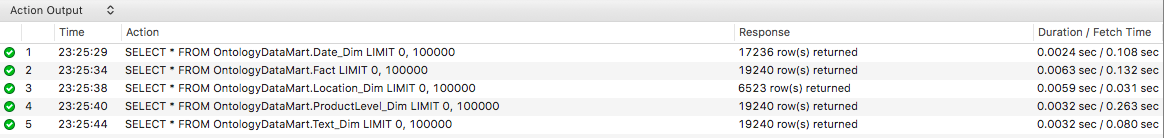
Fig. 6.1c Execution time of select queries of NewCube

Fig. 6.1d Execution time of select queries of OntologyDataMart



Fig. 6.1e Graph of increase in query execution time w.r.t data size

To further evaluate the query performance of Text Cube, execution time of complex queries, Query1 and Query 2 (Fig.6.1f, Fig.6.1g), including joins conditions have been evaluated.

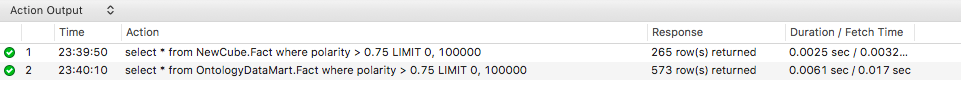


Fig. 6.1f Execution time of Query1 on NewCube and OntologyDataMart

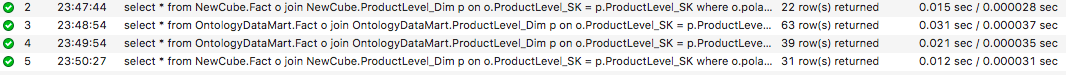


Fig. 6.1g Execution time of Query2 on NewCube and OntologyDataMart

**Query 1** – Where Clause

Apply few conditions on a table.

Select \* from NewCube.Fact where polarity > 0.75;

Select \* from OntologyDataMart.Fact where polarity > 0.75;

**Query 2 –** Table Joins

To get information about ‘iPhone 8’ that have polarity more than 0.85.

select \* from NewCube.Fact o

join NewCube.ProductLevel\_Dim p

on o.ProductLevel\_SK = p.ProductLevel\_SK

where o.polarity > 0.85 and p.AppleSecondLevel ='iPhone 8';

select \* from OntologyDataMart.Fact o

join OntologyDataMart.ProductLevel\_Dim p

on o.ProductLevel\_SK = p.ProductLevel\_SK

where o.polarity > 0.85 and p.AppleSecondLevel ='iPhone 8';

To get information about ‘iPhone 7’ that have polarity more than 0.65.

select \* from NewCube.Fact o

join NewCube.ProductLevel\_Dim p

on o.ProductLevel\_SK = p.ProductLevel\_SK

where o.polarity > 0.65 and p.AppleSecondLevel ='iPhone 7';

select \* from OntologyDataMart.Fact o

join OntologyDataMart.ProductLevel\_Dim p

on o.ProductLevel\_SK = p.ProductLevel\_SK

where o.polarity > 0.65 and p.AppleSecondLevel ='iPhone 7';

It has been observed that for simple select queries, the maximum increase in execution time is less that 6 milliseconds. Query 1 shows an increase of 9 milliseconds and Query 2 shows an increase of 16 milliseconds.

* 1. **Resource Utilization**

The resource utilization of Text Cube – OntologyDataMart has been monitored using MySQL Workbench Performance tools (“MySQL :: MySQL Workbench Manual :: 7 Performance Tools,” n.d.).

A simple SQL query can also be very resource intensive as scanning the fact table alone might take up a lot of resources. Therefore, it is important to ensure efficient utilization of CPU and disk. The current resource utilization of Text Cube has been shown below through dashboard and reports.

**Performance Dashboard**

Performance dashboard shows the graphs and statistics of a server (Fig.6.2a). It includes three statuses:

**Network Status** includes incoming Network Traffic, Outgoing Network Traffic, and Client Connections.

**MySQL Status** provides information about the table Open Cache efficiency, SQL Statements Executed, and counts (per second) for all the DDL and DML statements.

**InnoDB Status** provides buffer pool and disk activity information.

****

Fig. 6.2a A snapshot of the Performance Dashboard while constructing OntologyDataMart

**Performance Reports**

* **InnoDB Buffer Stats**

InnoDB consists of buffer pool for cache and indexes. It is a key element in improving MySQL performance. Larger pool size can store more data in the memory. This reduces disk reads.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table | Allocated | Data | Pages | Pages Hashed | Pages Old | Rows Cached |
| fact | 4898816 | 4266981 | 299 | 299 | 299 | 13407 |
| text\_dim | 3391488 | 3094630 | 207 | 207 | 207 | 19446 |
| productlevel\_dim | 2473984 | 1652726 | 151 | 151 | 151 | 19390 |
| date\_dim | 1884160 | 1452341 | 115 | 115 | 115 | 17350 |
| location\_dim | 491520 | 419757 | 30 | 30 | 30 | 6552 |

Table. 6.2.1 InnoDB Buffer Stats

* **Schema Object Overview (High Overhead)**

Shows count by object type for each schema.

Note: On instances with a large number of objects, this can take some time to execute.

|  |  |  |
| --- | --- | --- |
| Schema | Object Type | Count |
| OntologyDataMart | BASE TABLE | 6 |
| OntologyDataMart | INDEX (BTREE) | 12 |

Table. 6.2.2 Schema Object Overview

* **Schema Table Statistics**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table | Rows Fetched | Fetch Time | I/O Read Reqs (#) | I/O Read (#) | I/O Read Time | I/O Misc. Reqs (#) | I/O Misc. Time |
| fact | 676171 | 1084872.14 | 384 | 8996300 | 151853.45 | 45 | 5741.25 |
| productlevel\_dim | 270367 | 665877.4 | 178 | 2524354 | 75140.76 | 45 | 2285.41 |
| text\_dim | 78334 | 564827.4 | 255 | 5817390 | 29312.21 | 45 | 1697.82 |
| date\_dim | 246620 | 451311.66 | 190 | 3720116 | 35208.48 | 45 | 4280.18 |
| location\_dim | 97862 | 94094.59 | 57 | 541590 | 20950.55 | 45 | 2145.02 |

Table. 6.2.3 Schema Table Statistics

* **Statements in Highest 5 Percent by Runtime**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Query | Maximum Time | Avg Time | Rows Sent (#) | Avg. Rows Sent (#) | Rows Scanned (#) | Avg. Rows Scanned (#) |
| SELECT SUM ( `Fact` . `polarity` ) AS `sum\_polarity\_ok` | 0 | 709458.4 | 454744.11 | 354729.19 | 24400 | 12200 |
| SELECT \* FROM `OntologyDataMart` . `Text\_Dim` LIMIT ? | 337338.34 | 337338.3 | 337338.34 | 19240 | 19240 | 19240 |
| SELECT `Location\_Dim` . `Location` AS `Location` | 533362.06 | 341883.2 | 266681.03 | 11842 | 5921 | 75210 |
| SELECT SQL\_NO\_CACHE \* FROM `Fact` | 220539.77 | 220539.8 | 19240 | 19240 | 19240 | 19240 |
| ALTER TABLE `OntologyDataMart` . `Date\_Dim` ADD COLUMN `Date\_SK` INTEGER NOT NULL AUTO\_INCREMENT COMMENT ? FIRST | 0 | 379366.8 | 244000.61 | 189683.41 | 0 | 0 |
| UPDATE `OntologyDataMart` . `Date\_Dim` SET `\_id` = SUBSTRING ( `\_id` | 329296.81 | 299932.9 | 164648.4 | 0 | 0 | 17236 |
| ALTER TABLE `OntologyDataMart1` . `Location\_Dim` ADD COLUMN `Location\_SK` INTEGER NOT NULL AUTO\_INCREMENT COMMENT ? FIRST | 0 | 163816.8 | 163816.75 | 163816.75 | 0 | 0 |

Table. 6.2.4 Statements in Highest 5 Percent Runtime

1. **APPLICATION OF THE TEXT CUBE**

Section 7 discusses how a text cube can be utilized for analysis of data and reporting by an analytics team. Sub-section 7.1 shows how Tableau can be used for generating graphs and dashboards from the text cube constructed in Section 5. Sections 7.1a shows how to set up connection between SQL server and Tableau. In Section 7.1b, 7.1c, 7.1d and 7.1e various dashboards generated in Tableau are described.

The text cube formed using ontology for Apple trademark products can be very beneficial in data analysis and prediction. It can be used by the analysts for generating reports and dashboards that can be valuable in studying factors such as trends, sales, popularity, and success/ failure of products. These reports when shared with related departments and planning and innovation teams, can be helpful in making future predictions, improve existing products and services, and analyze how the company is doing overall.

One of the popular reporting tools used in the industry is Tableau. It has been used to show how the user requirements specified in Section 2.3 have been met by the text cube obtained in Section 5. The dashboards in Tableau can be responsive and can filter data at different level for detailed analysis.

* 1. **Tableau Report**

Tableau is a business intelligence tool used for data analysis and visual representations. It makes understanding data easier for someone who is not a data analyst or scientist (“Tableau Help,” n.d.). For the purpose of this project, tableau is used to form graphs and dashboards using the text cube.

**7.1a Connection to the Data Mart**

For the purpose of generating reports, Tableau is connected to the MySQL server, OntologyDataMart, Live. If required, the data from the database can be extracted and then used for reports. All the dimensions and fact table from the connected database are pulled into Tableau. The dimensions are then joined to the fact table.

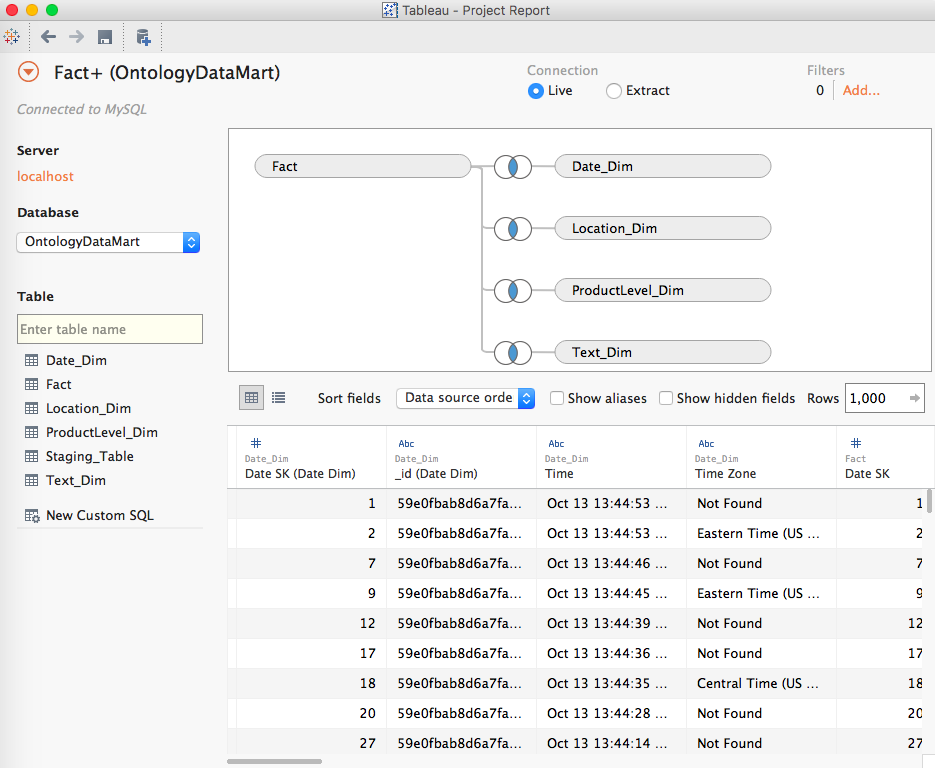


Fig. 7.1a Tableau connected to MySQL Live

**7.1b Time- Based Analysis**

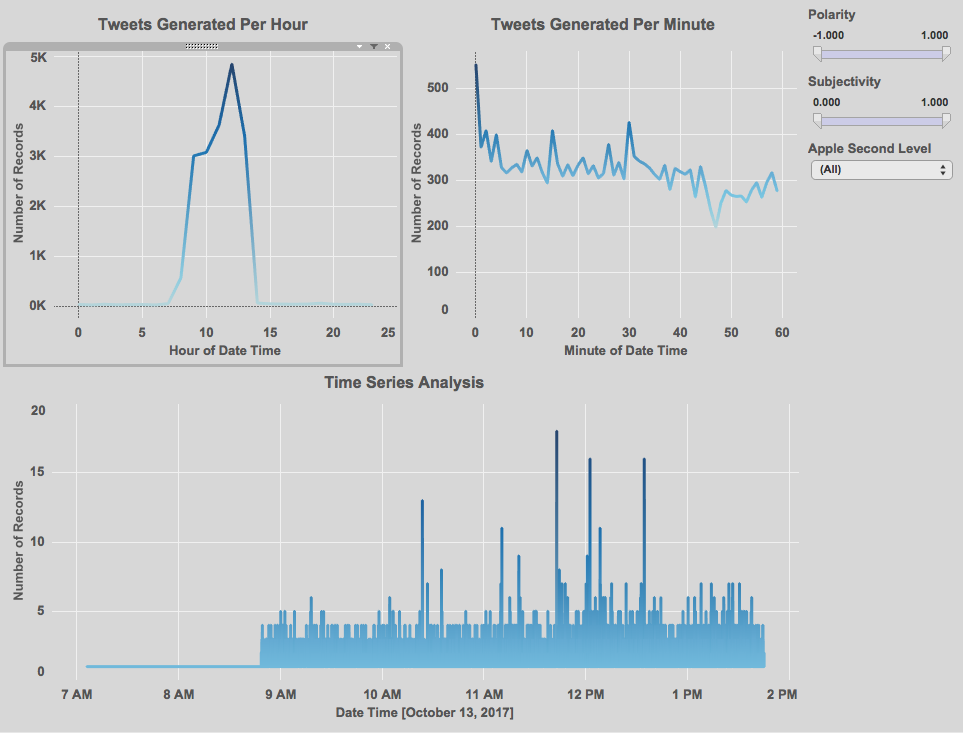


Fig. 7.1b Time based analysis

The dashboard consists of three graphs.

* **First graph** shows the number of tweets (related to Apple products) generated per hour.
* **Second graph** shows the number of tweets (related to Apple products) generated per minute to give a more detailed view.
* **Third graph** shows the overall time series for the number of tweets (related to Apple products).

**7.1c Location- Based Analysis**

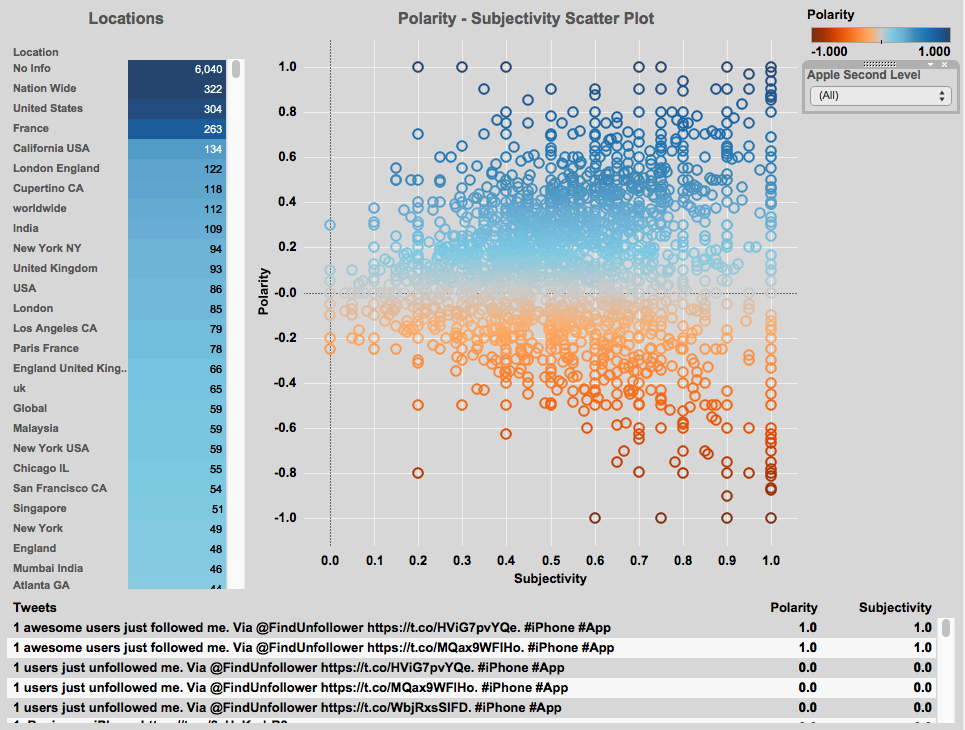


Fig. 7.1c Location based analysis

The dashboard consists of two filters, a scatter plot and a table.

* The **Location filter** allows selection of a particular location and changes the scatter plot and tweet table accordingly.
* **Apple Second Level filter** allows selection of Apple product from second level.
* The **Polarity – Subjectivity scatter plot** shows the sentiments related to tweets in a particular locations.
* The **Tweets table** shows the text, polarity value and subjectivity value of all the tweets based on the selected location.

For a location selected from Location filter and a product selected from Apple Second Level filter, the Polarity-Subjectivity scatter plot shows how many how many significant tweets with positive and negative sentiments originate in that location, and the Tweets table shows the text for those selected tweets.

It can be useful in analyzing the popularity of products in various parts of the world, or to do a comparative study of which product is more popular where, or see what positive/ negative the tweets say that makes a product likable/ not likable among people of a region.

**7.1d Ontology Levels and Sentiments**

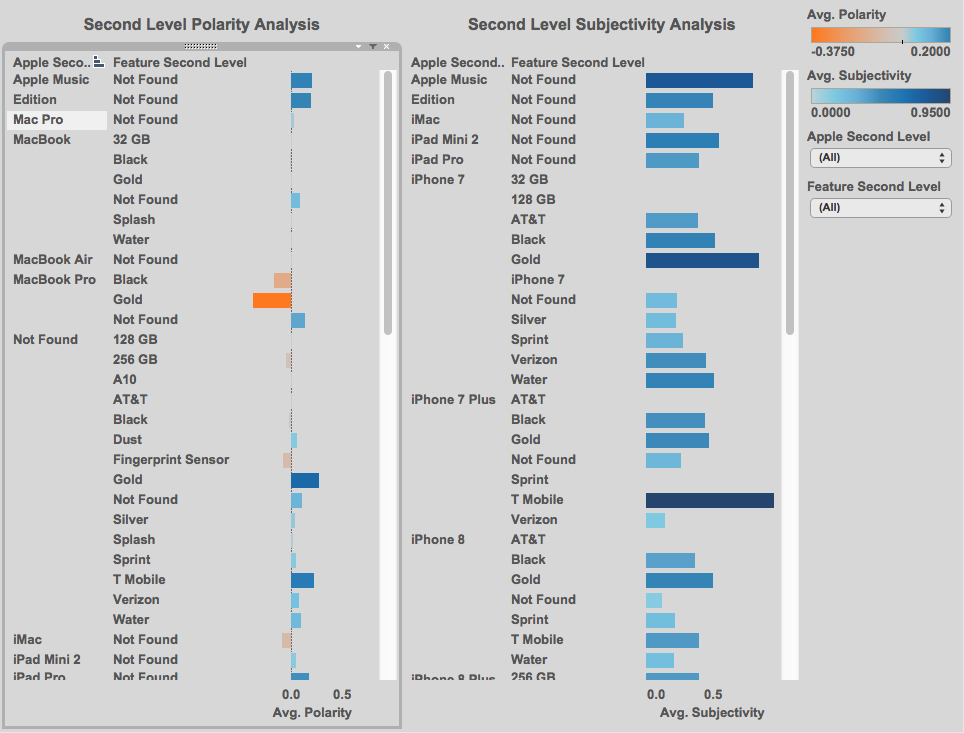


Fig. 7.1d Ontology levels and sentiments

The dashboard depicts two bar graphs showing **Polarity and Subjectivity** according to the labels in second levels of Apple and Features classes. For any selection in a graph, the values automatically change in the other graph.

**7.1e Polarity Map**

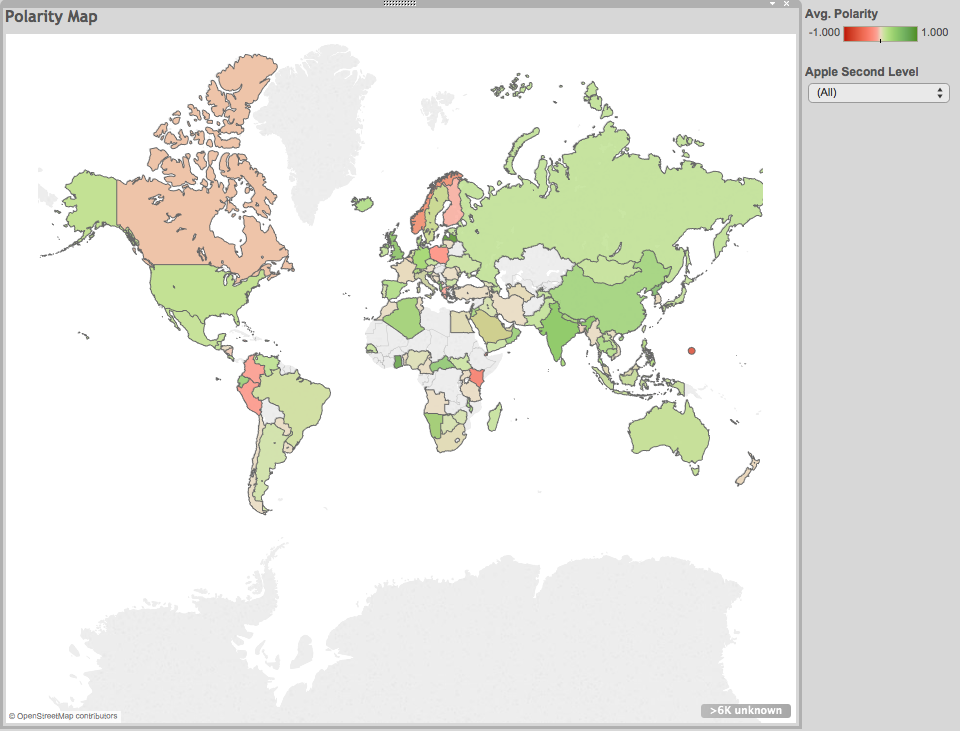
****

Fig. 7.1e Polarity map

The dashboard shows **Polarity distribution** over the world map. It helps identify people from which countries in the world have good/bad reviews about the products.

1. **CHALLENGES**

There are various challenges faced while carrying out the implementation of ontology based text cube.

* It is difficult to construct an ontology which includes all domain factual and problem solving details that can be integrated with a text cube as one of its dimensions.
* Representing a hierarchy as a dimension can lead to redundancy.
* Another major factor that can be bothersome to consider in this study is the integration of various tools in order for the required data to exist easily in different forms.
* It can be strenuous to find the parameters that can most likely improve the performance of the cube operations.

1. **LIMITATIONS**

The following constraints apply to the text cube implementation system.

* The amount of data that can be processed at one time, on an averagely configured system can be limited or time consuming.
* The compatibility among software and tools in order for the data to move from one stage to the next can be difficult to achieve.

**10. CONCLUSION**

In this paper, a text cube of Twitter data has been formed using ontology. The ontology provided structure to the raw data and was used to extract hierarchy level of each tweet. Also, the sentiments related to the various products were calculated. Thus, a text cube was formed using existing and extracted information and it performs well in query processing and hardware utilization. This cube is beneficial for analysts to study their products. A proper analysis using the text cube can help in improving the existing products and services by a company and can also lead to new ideas.

**10.1 FUTURE WORK**

* In this project, a small dataset was used to design the work flow of the process. For further study, a larger dataset can be used.
* The ontology levels defined represented two levels of hierarchy. A more detailed ontology can be implemented.
* Tools like MS SQL Server can be used to build the cube as well as to perform analysis. It can handle large datasets and is better equipped to design complex models. Its three components – SSIS, SSAS and SSRS can help build the data mart, perform analysis and generate reports.
* For large datasets, the text cube can be stored on cloud.

**REFERENCES**

Garcia-Alvarado, C., & Ordonez, C. (2012). Query Processing on Cubes Mapped from Ontologies to Dimension Hierarchies. In *Proceedings of the Fifteenth International Workshop on Data Warehousing and OLAP* (pp. 57–64). New York, NY, USA: ACM. <https://doi.org/10.1145/2390045.2390055>

Lee, S., Kim, N., & Kim, J. (2014). A Multi-dimensional Analysis and Data Cube for Unstructured Text and Social Media. In *2014 IEEE Fourth International Conference on Big Data and Cloud Computing* (pp. 761–764). <https://doi.org/10.1109/BDCloud.2014.117>

Lin, C. X., Ding, B., Han, J., Zhu, F., & Zhao, B. (2008). Text Cube: Computing IR Measures for Multidimensional Text Database Analysis. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 905–910). <https://doi.org/10.1109/ICDM.2008.135>

MySQL :: MySQL Workbench Manual :: 7 Performance Tools. (n.d.). Retrieved April 9, 2018, from <https://dev.mysql.com/doc/workbench/en/wb-performance.html>

Naive Bayes and Sentiment Classification. (n.d.). Retrieved from <https://web.stanford.edu/~jurafsky/slp3/6.pdf>

Oukid, L., Asfari, O., Bentayeb, F., Benblidia, N., & Boussaid, O. (2013). CXT-cube: Contextual Text Cube Model and Aggregation Operator for Text OLAP. In *Proceedings of the Sixteenth International Workshop on Data Warehousing and OLAP* (pp. 27–32). New York, NY, USA: ACM. <https://doi.org/10.1145/2513190.2513201>

protégé. (n.d.). Retrieved May 16, 2017, from <http://protege.stanford.edu/>

PyMongo 3.4.0 Documentation — PyMongo 3.4.0 documentation. (n.d.). Retrieved May 16, 2017, from <https://api.mongodb.com/python/current/>

rdflib 4.2.2 — rdflib 4.2.2 documentation. (n.d.). Retrieved May 16, 2017, from <https://rdflib.readthedocs.io/en/stable/>

Tableau Help. (n.d.). Retrieved April 9, 2018, from <https://www.tableau.com/support/help>

Tao, F., Lei, K. H., Han, J., Zhai, C., Cheng, X., Danilevsky, M., … Zhao, B. (2013). EventCube: Multi-dimensional Search and Mining of Structured and Text Data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1494–1497). New York, NY, USA: ACM. <https://doi.org/10.1145/2487575.2487718>

TextBlob: Simplified Text Processing — TextBlob 0.12.0 documentation. (n.d.). Retrieved May 16, 2017, from <https://textblob.readthedocs.io/en/dev/>

Thornthwaite, W. (2003, February 6). Design Tip #43: Dealing With Nulls In The Dimensional Model. Retrieved April 9, 2018, from <https://www.kimballgroup.com/2003/02/design-tip-43-dealing-with-nulls-in-the-dimensional-model/>