***I.J. Mathematical Sciences and Computing,* 2015, 4, 11-20**
Published Online November 2015 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2015.04.02

***Available online at* http://www.mecs-press.net/ijmsc**

A Natural Language Query Builder Interface for Structured Databases Using Dependency Parsing

Rohini Kokare[a*], Kirti Wanjale[b]

[a]*ME Computer Student, VIIT, Department of Computer Engineering, Pune-411048, India*
[b]*Associate Professor, VIIT, Department of Computer Engineering, Pune-411048, India*

Abstract

A natural language query builder interface retrieves the required data in structured form from database when query is entered in natural language. The user need not necessarily have sufficient technical knowledge of structured query language statements so nontechnical users can also use this proposed model. In natural language parsing, getting highly accurate syntactic analysis is a crucial step. Parsing of natural languages is the process of mapping an input string or a natural language sentence to its syntactic representation. Constituency parsing approach takes more time for parsing. So, natural language query builder interface is developed in which the parsing of natural language sentence is done by using dependency parsing approach. Dependency parsing technique is widespread in natural language domain because of its state-of-art accuracy and efficiency and also it performs best. In this paper, the buffering scheme is also proposed for natural language statements which will not load the whole sentence if it was done previously. Also there was a need of generalized access to all tables from database which is handled in this system.

Index Terms: Natural Language Query Builder Interface (NLQBI), Natural Language Processing (NLP), Dependency parsing, Structured Query Language (SQL), POS (Part Of Speech) tagging.

1. Introduction

Since the end of 1960's there have been a large number of researches done regarding the theories and implementations of NLQBI's. Asking question to databases in natural language is very convenient and easy method of accessing data especially for casual users who do not understand complex database query languages. As the usage of databases has spread widely, the concept of user interface presented new challenges to the designers. The main goal of this system is to provide communication between user and computer without recalling any sort of database query syntax.

* Corresponding author. Tel.: +91 8888003378;
E-mail address: rohinikokare@gmail.com

Constituency parsing and dependency parsing are the two parsing techniques broadly used in natural language processing. In constituency parsing, the parse tree breaks a sentence into sub-phrases. Non-terminals in the tree are types of phrases, the terminals are the words in the sentence, and the edges in constituency parsing are unlabeled. For example consider a simple sentence "John sees Bill", a constituency parse would be as in fig 1.

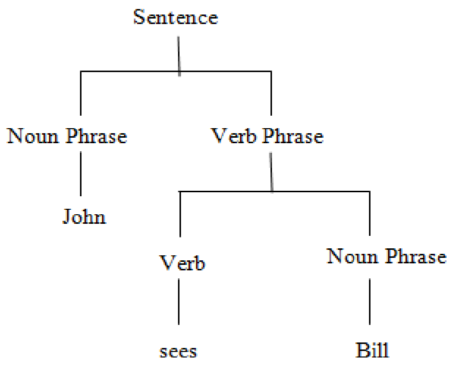

Fig. 1. Constituency parsing for sentence "John sees Bill"

Whereas, a dependency parse connects words according to their relationships. Each vertex in the tree represents a word, child nodes are words that are dependent on the parent, and edges are labeled by the relationship. A dependency parse of same sentence "John sees Bill", would be as shown in figure 2.

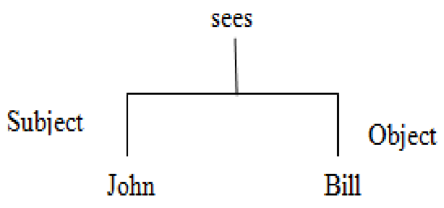

Fig. 2. Dependency parsing for sentence "John sees Bill"

But constituency parsing takes a lot of time to parse the words of the sentence. So there was a need to develop a NLQBI which will extract POS and process the query in less time. The parser of the proposed natural language query builder interface is developed using dependency parsing method. Dependency parsing is a technique where a sentence is given in natural language as an input and produces output in the form of dependency tree. Dependency parsing has proved useful in many applications like question-answering, machine translation, information extraction and natural language generation. Dependency structures contain much of predicate argument information. The basic notation of dependency is based on the idea of lexical items linked by binary asymmetric relations called as dependencies which focus on relations between words.

Dependency parsing presents a number of advantages when compared to syntactic parsing or phrase structured parsing. Dependency parsing has three main advantages. The very first, dependency links that are formed between two words of the sentence are close to semantic relationships needed for the next step of interpretation. Second, the dependency tree contains one node which represents one word, instead of mid-level nodes that represents words as in constituent trees, making the task of parsing more straightforward. Third, dependency parsing lends itself to word-at-a-time operation, i.e., parsing can be done by accepting and attaching words as entered by user. That means it does not wait for complete sentence to be loaded for parsing. It improves the overall system.

The challenge was a buffering scheme that has to be applied to the natural language statements. The statements which are already given as input to the system will not parse it again. It will save all statements and corresponding SQL statement which was previously generated. It will directly fetch the SQL statement if the same statement is given as input by the user.

The objectives were set after doing the survey of existing natural language query builder systems and finding the gaps between them. This is discussed in section II B. The objectives are to extract the dependencies, noun phrases, and generate the parse tree. The logical query is the generated which is further translated to SQL syntax. The SQL query is then given to underlying database to retrieve the results. The system is also benchmarked against the constituency parsing technique. The buffering scheme is implemented so that the overall performance of the system is significantly improved.

A constituency parsing is also implemented to compare results with dependency parsing and their results are discussed in section 5.

Rest of the paper is organized as follows: Section 2 presents the related work which includes survey and existing NLQBI's. In section 3, detailed description of the proposed system is mentioned. Section 4 describes the implementation details and results. And finally section 6 concludes the paper.

2. Related Work

2.1. Literature Survey

Mo Shen et al. [2] proposed a dependency grammar, in which predicate-argument structures are encoded to build modelled syntax. New reranking approach was proposed for dependency parsing that can utilize complex sub tree representations. The limitation of this system is that it prohibits incorporation of information from large-scale structured data and it was not used for developing a query builder. Emily Pitler et al.[5] proposed a graph-based algorithm for non-projective parsing with higher order features. But it has limitations that it does not give faster variants for third-order graph-based projective parsing and also was not meant for developing a query builder. Fei Li et al.[6] proposed a system called NaLIR. In this, for each ambiguity, system generates multiple likely interpretations for the user to choose from, which resolves ambiguities interactively with the user. The limitations were that there was lack of reusability of queries and hence was less precise. This system does not use dependency parsing approach. Zhenghua Li et al.[7] proposed a joint models for dynamic programming based decoding algorithms which can incorporate rich POS tagging and syntactic features . Limitations of this approach are firstly, average perceptron to learn the feature weights of the joint models are used, which equally treats the POS and syntactic features. Training procedures should be better in this case. Secondly, it is not implemented for query builder. Chen D et a[8] proposed a novel dependency parser using neural network classifier for use in greedy dependency parsers using sparse indicator features in both accuracy and speed. A limitation of this system is that it only relies on small number of dense features and was not used for query builder. Alessandra Giordani et al. [13] proposed a system that translate natural language query to SQL query by reranking with an SVM-ranker based on tree kernels. Here also this system does not use dependency parsing. Miguel Llopis et al. [14] proposed a natural language query builder interface using ontology based approach. The main drawback of using this kind of resolution is its low precision. Abhijeet Gupta et al.[15] proposed the syntactic parser that uses the Computational Paninian Grammar (CPG) framework. Limitations of this system is that aggregation operators and Join are not supported also the dependency relations are syntactico-semantic in nature. Michael Minock et.al[17] proposed a natural language query builder interface where queries are modelled in a higher-order Codd's tuple calculus and use synchronous grammars which is extended with lambda functions to represent semantic grammars.

2.2. Existing Natural Language Query Builder Interfaces

Akshay G. Satav et.al[3] proposed a system in which Word Pair Mining Technique is used for spelling correction and syntactic parsing is done. Verena Rieser et.al[4] proposed a framework for adaptive natural language generation was developed where the problem is formulated as stochastic incremental planning under uncertainty, which can be approached using reinforcement learning methods. This system is applied for spoken dialogue system. Pranali P. Chaudhari[9] developed an interface where dictionary updating with synonyms of the words is the main feature along with SQL translation. Syntactic parsing technique is used. But only implementation of aggregate functions and combination of two or more operations has been executed. Ashish Kumar et.al[10] developed a Natural Language Interface to Database systems (NLIDB) through which user can interact with the database in a more convenient and flexible way. The technique used was syntactic parsing technique. The drawback of this system is that long questions were not allowed and proper error messages were not displayed in case of query failure. Gauri Rao et.al[16] proposed a system which provides search interface and reduces the part of user for recalling the tedious syntax of databases, this system allows user to get the database information in his/her language. Semantic grammar technique is used for developing this system. The drawback of this system was that user has to fire queries in WH type question; it also could not automate the related words for table and column names.

3. Proposed System

The proposed system architecture is as shown in fig 3.

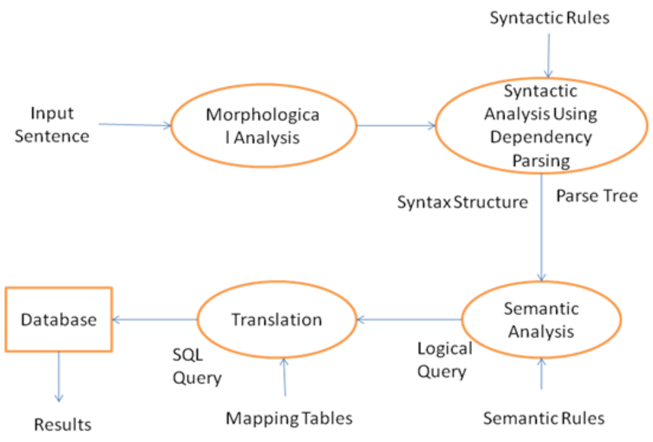

Fig. 3. System Architecture

The user will enter the query in natural language sentence. In morphological analysis, the entered sentence is then checked for the stop words and punctuation marks which are removed. After that the tokens are separated. The next step is syntactic analysis in which the sentence is then converted into a tree structured form using dependency parsing. The nouns, adjectives etc are related to each other in the form of binary asymmetrical links. These all links form a dependency structure. The dependency parse tree is then generated. This selected parse tree is then checked for the meanings in the semantic analysis phase. It is then converted to logical query. The query is mapped with the database tables and attributes with the extracted meaningful tokens in the first phase. After that, the query is translated to SQL syntax by replacing the tokens with table names or attribute names which is then sent to database. Database then fetches the exact results to the user.

3.1. Dependency Parsing

The basic notation of dependency is based on the idea of lexical items linked by binary asymmetric relations called dependencies which focus on relations between words. A dependency relation holds between a head (also called governor) and a dependent (also called modifier) of each pair of word in a sentence. The words of the sentence are connected by typed dependency relations. The arcs (links) indicate certain grammatical relation between words. Each word depends on exactly one parent. The tree starts with a root node. Dependency also resolves ambiguity. A dependency analysis of simple sentence as an example is given in fig4.

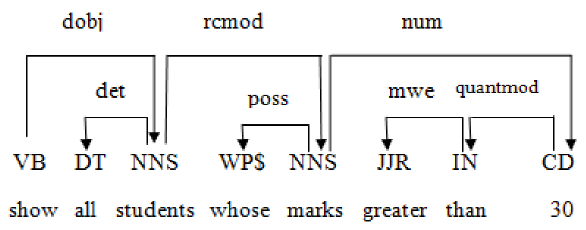
dobj
rcmod
num
det
poss
mwe
quantmod
VB DT NNS WP$ NNS JJR IN CD
show all students whose marks greater than 30

Fig. 4. Dependency Structure

As an illustration of this structure representation, consider the fig 4, the nodes are the word tokens of the sentence (annotated with parts-of speech) and the arcs are labelled with grammatical functions.

The noun extraction which is mapped to database concepts like table names and column names is done by using dependency parsing. The dependency tree is generated from the sentence. The grammatical structure and typed dependencies are used to map the governing and/or dependent tokens to the columns of the table in database. The grammatical structure of a sentence is generated from the Penn Treebank language pack. Penn Treebank is a parsed corpus of words consisting of over 4.5 million American English words. This mapping is done in a generalized manner which means all tables and respective columns of tables can be accessed from database. This extraction is not limited to any specific table in database. The tagged words are used to get the noun phrase from the input sentence. Accordingly the tables and its columns are extracted. After noun extraction the natural language sentence is converted to SQL syntax statement.

3.2. Buffering

The buffering scheme is applied to the natural language statements. The input natural language sentence and its corresponding SQL converted statement is saved in a buffer. In this scheme, it will check if the buffer is empty. If so it will scan the statement and do the whole process of parsing and save the statement and input sentence in buffer. If the buffer has some statements, it will check if the input sentence is present in the buffer. If it is present in the buffer, the corresponding matching SQL statement is directly fetched form the buffer itself. It will not parse and load the sentence if it is present in the buffer. If the input sentence is the new sentence then only it will do the process of parsing. Buffering will directly fetch the SQL statement, if the same statement is given as input by the user. Thus getting the statements from buffer does not take extra time irrespective of the size of statement and hence it gives the best performance. Otherwise parsing the sentence again and converting it into SQL will take significant time period for fetching the results and also the parsing time depends on the size of the input sentence. Hence this buffering scheme will reduce time for frequently asked queries.

4. Implementation

This proposed system is implemented in java technology. MySQL is used as underlying database. We used Eclipse Java EE IDE Indigo Service Release tool for implementation in windows OS. Database connection is done by using JDBC library. Algorithm for the proposed system and results for dependencies are discussed as follows:

4.1. Algorithm

Algorithm 1: NLQBI using Dependency Parsing

```
If the buffer is empty do the following
{
1.  Tokenization: Split the input sentence into various tokens.
2. Stanford Parser is used to perform the parsing and generate parse tree.
3. The grammatical relations are extracted by using Stanford Typed Dependencies.
4. The nouns and prepositions are extracted from grammatical relations.
5. Semantic analysis is performed using the type of grammatical relations.
6. The tokens and the nouns are mapped to the database entities like tables and columns.
7. The input natural language query is mapped initially to an intermediate query by mapping rules for
the         various tables, columns and literals.
9. It is then converted into SQL query language.
10. The SQL Query is sent to database and executed there.
11. Finally the results are fetched to the user.
}
Else check if input sentence is already there in buffer
            If yes, take the corresponding SQL statement and directly go to step 10.
Else start from step 1.
```

4.2. Results

This system is tested for various natural language queries which are provided by technical as well as non-technical people. The results are evaluated. The output of the query is checked with the actual records at database side and verified for correctness of results. Dependency extraction and POS tagging are the important steps in the parsing. For example consider the sentence: “show all students whose marks greater than 30”. The dependency representation and POS tagged words for this sentence can be given as shown in table 1.

The types of queries supported by this system are conditional queries (WHERE clause, greater than, less than, equal to), range queries (BETWEEN, AND, OR clause), aggregation queries (SUM, COUNT, MIN, MAX, AVG), ORDER BY. The exact meaning of all the dependencies and its usage is well described by Marie-Catherine et.al [18].

Table 1. Mapping of POS and Typed Dependencies

Typed Dependencies	Tagged Part of Speech
det(students-3, all-2)	NN show
dobj(show-1, students-3)	DT all
poss(marks-5, whose-4)	NNS students
rcmod(students-3, marks-5)	WP$ whose
mwe(than-7, greater-6)	NNS marks
quantmod(30-8, than-7)	JJR greater
num(marks-5, 30-8)	IN than
	CD 30

The tables and attributes are mapped to the database by using tokens as shown in table 2. Tokenization of input natural language sentence is done and these tokens are then checked if those are column names or table names. Accordingly, the column names and table names are extracted. This mapping produces generalized access to all the tables in database. Also, if the tokens are NNS or NNP are also examined and mapped as shown in table 2 i.e. if the token is “students”, it is mapped to “student” table of database. So user can access any table from the database. After mapping the SQL query is constructed. The constructed SQL query for above example is: “select * from student where marks>30”. This is then sent to database for results.

Table 2. Mapping of Nouns and Attributes with Database

Tokens	Database	Type
Students	Student	Table
Marks	Marks	Column

5. Experimental Evaluations

In order to evaluate the effectiveness of our system, we have applied the following experiments:

1. Accuracy in query interpretation
2. Comparison in constituency and dependency parsing
3. Compatibility

5.1. Accuracy in Query Interpretation

The experiments we present here are based on the corpus of series of questions gathered from the technical as well as non-technical people. This corpus consists of natural language statements. To determine logical query equivalence, queries were marked as correct if their parse to logical form was equivalent to the manually constructed correct result. Accordingly, the system accuracy was calculated. System accuracy A can be calculated as shown in equation 1 below.

$$A = \frac{\text{Number of queries understood and processed by the system}}{\text{Total number of queries introduced by the user}} * 100 \tag{1}$$

This system produced an accuracy rate of 91.66%.

5.2. Comparison Between Constituency and Dependency Parsing

The time evaluation was done and compared by using two techniques dependency parsing technique and constituency parsing technique. The parameters evaluated are, time for POS tagging, time for SQL translation and total time with query results. This is shown in table 3 which is measured in seconds. Form this table we can see that time required for dependency parsing is much less as compared to constituency parsing technique. The Typed dependencies extraction of dependency parsing is more efficient and hence increases the overall system performance.

Table 3. Time for POS tagging, SQL Translation and Total time of system

#	Time for POS Tagging	Time for SQL Translation	Total Time with Query Results
Dependency Parsing	0.5947	4.8522	5.4803
Constituency Parsing	15.9467	16.23	17.3393

Graphically this evaluation is as shown in figure 5. Hence, dependency parsing is much faster technique than constituency parsing.

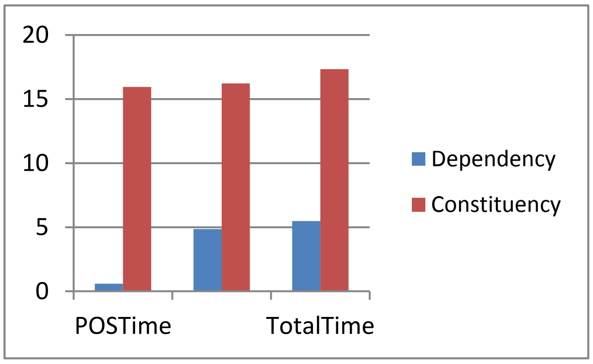

Fig. 5. Dependency and Constituency Parsing Comparison

We saw that dependency approach takes less time for generating natural language query results. But our proposed buffering scheme still improves the overall system performance. The time required for retrieving query results from database using buffering scheme is around 2.5658e-5 seconds which is much less than any other approach. Hence, dependency parsing technique along with buffering scheme makes system more efficient in terms of performance.

5.3. Compatibility

We tested our approach with different operating environments i.e. windows OS and Unix OS. In our tested environment, our system doesn't affect the working applications. When this system is executed and query is entered, during query processing, database connection is established. This connection and processing of system does not affect other running processes, this shows compatibility of the system.

6. Conclusions

In this paper a natural language query builder interface is proposed which will be used by all types of users (technical, non-technical). So NLQBI with dependency parsing will become more efficient, accurate and user friendly. Also there was need of different parsing techniques like dependency parsing which improves the overall system performance as compared to constituency parsing. Dependency parsing also resolves ambiguity. This proposed system adds an innovative feature like buffering and generalized access to all database tables. The buffering scheme applied to the natural language statements will significantly reduce the parsing time for previous statements that are given as input. Due to this scheme, the system will not parse the statements again and improve the performance. At the end we can conclude that the proposed system is a medium of communication between user and database and fetches the results of users query in structured form.

The future scope of this approach is to support complex queries, nested queries and complex join operations which will make this system much stronger.

Acknowledgement

Inspiration and guidance are invaluable in every aspect of life, especially in the fields of academics, which I have received from my respected guide Prof. Mrs. K. H. Wanjale. I would like to thank her for her endless contributions of time, effort, valuable guidance and encouragement she has given me.

I would also like to thank external guide Mr. Vishal Macchewad for giving me sponsorship project in Tata Consultancy Services Ltd., the guidance and timely support rendered during the course of the work.

References

[1] Rohini Kokare, Kirti Wanjale, “A Survey of Natural Language Query Builder Interface for Structured Databases using Dependency Parsing”, International Journal of Computer Application, Pages 9-14, 18 December 2014.
[2] Mo Shen, Daisuke Kawahara, and Sadao Kurohashi, “Dependency Parse Reranking with Rich Subtree Features”, IEEE Transactions on Audio, Speech, and Language Processing, vol.22, no.7, July 2014.
[3] Akshay G. Satav et al., “A Proposed Natural Language Query Processing System”, International Journal of Science and Applied Information Technology, Volume 3, No.2, 2014.
[4] Verena Rieser, Oliver Lemon, and Simon Keizer, “Natural Language Generation as Incremental Planning Under Uncertainty: Adaptive Information Presentation for Statistical Dialogue Systems”, IEEE/ACM transactions on audio, speech and language processing, Vol. 22, No.5, May 2014.
[5] Emily Pitler, “A Crossing-Sensitive Third-Order Factorization for Dependency Parsing”, Transactions of the Association of Computational Linguistics, Volume 2, Issue 1, 2014.
[6] Fei Li, H. V. Jagadish, “Constructing an Interactive Natural Language Interface for Relational Databases”, Proceedings of the VLDB Endowment, Vol. 8, No. 1, pages 73-84, 2014.
[7] Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, and Wenliang Chen, “Joint Optimization for Chinese POS Tagging and Dependency Parsing”, IEEE Transactions on Audio, Speech, and Language Processing, vol.22, no.1, Jan 2014.
[8] Chen, D, Manning C, “A fast and accurate dependency parser using neural networks”, In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 740–750, Doha, Qatar, Association for Computational Linguistics, 2014.
[9] Pranali P. Chaudhari, “Natural Language Statement to SQL Query Translator”, International Journal of Computer Applications (0975 – 8887),Vol 82, No5, November 2013.
[10] Ashish Kumar, Kunwar Singh, “Natural Language Interface to Databases: Development Techniques”, Elixir International Journal, 2013.
[11] Martins, M. Almeida, and N. A. Smith, “Turning on the turbo: Fast third-order non-projective turbo parsers”, In Proceedings of ACL (Short Papers), pages 617–622, 2013.
[12] Bohnet and J. Kuhn, “The best of both worlds – a graph-based completion model for transition-based parsers”, In Proceedings of EACL, pages 77–87, 2012.
[13] Alessandra Giordani and Alessandro Moschitti, “Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked”, Proceedings of COLING 2012: Posters, pages 401–410, 2012.
[14] Miguel Llopis, Antonio Ferrández, "How to make a natural language interface to query databases accessible to everyone: An example", Computer Standards & Interfaces, Elsevier, October 2012.

[15] Abhijeet Gupta, Arjun Akula, Deepak Malladi, Puneeth Kukkadapu, Vinay Ainavolu, Rajeev Sangal, "A Novel Approach Towards Building a Portable NLIDB System Using the Computational Paninian Grammar Framework", 2011.
[16] Gauri Rao et al., "Natural Language Query Processing using Semantic Grammar", International Journal on Computer Science and Engineering, Vol. 02, No. 02, Pages 219-223,2010.
[17] Michael Minock,Peter Olofsson, Alexander Näslund, "Towards Building Robust Natural Language Interfaces to Databases", Natural Language and Information Systems Lecture Notes in Computer Science, Springer, Volume 5039, 2008, pp 187-198, 2008.
[18] Marie Catherine de Marneffe. Christpher D Manning, "Standford Typed Dependencies Manual", 2008.
[19] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch, Natural Language Interfaces to Databases – An Introduction, Journal of Natural Language Engineering 1 Part 1 (1995), 29–81.

Authors' Profiles

Rohini B. Kokare is a post graduate student, pursuing masters degree for computer engineering in Vishwakarma Institute of Information Technology (VIIT), affiliated to Savitribai Phule Pune University, Pune. Her research area is natural language processing.

Mrs. Kirti Wanjale is currently working as an Associate Professor in the Department of Computer Science in Vishwakarma Institute of Information Technology (VIIT), affiliated to Savitribai Phule Pune University, Pune. She has 14 years of teaching experience. She is currently pursuing PhD. She has presented papers in many national and international conferences and published articles in many international journals. Her research area is image processing.

How to cite this paper: Rohini Kokare, Kirti Wanjale,"A Natural Language Query Builder Interface for Structured Databases Using Dependency Parsing", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.1, No.4, pp.11-20, 2015.DOI: 10.5815/ijmsc.2015.04.02