Volume 7, Issue 3, March 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Ratatta: Chatbot Application Using Expert System

[1]C.P. Shabariram[*], [2]V. Srinath, [3]C.S. Indhuja, [4]M. Vidhya
[1]Assistant Professor, [2, 3, 4]Student
[1, 2, 3, 4]Department of CSE, Sri Shakthi Institute of Engineering & Technology, Coimbatore, Tamilnadu, India

DOI: 10.23956/ijarcsse/V7I3/0147

Abstract— A Chatbot is a computer program which conducts a conversation via auditory or textual methods. Chatbots are typically used in dialog systems for various practical purposes including customer service or information acquisition. Many of the chatbots today are not efficient in producing the required results quickly. This project aims to build a chatbot aka Rattata that quickly responds to a given input query. The first build of Rattata focuses on using a method of scanning for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a database. The database is an xml-based dataset from stack-exchange. The dataset is clustered using the K-Means machine learning algorithm, to make the dataset more readable, with only the required attributes. Then the input query is processed using query processing algorithms like Krovetz Stemming algorithm and the Parallel Stemmer algorithm. After processing the input query, the required data is fetched from the database and displayed to the end user. In the second build we focus on improving the processing of the keywords using Natural Language Processing.

Keywords— Machine learning, Pattern matching, Query processing, NLP

I. INTRODUCTION

A Chatbot is a computer program which conducts a conversation via auditory or textual methods. Such programs are often designed to convincingly simulate how a human would behave as a conversational partner. Chatbots are typically used in dialog systems for various practical purposes including customer service or information acquisition. Some chatbots use sophisticated natural language processing systems, but many simpler systems scan for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a database. There are two main types of chatbots, one functions based on a set of rules, and the other more advanced version uses artificial intelligence. The chatbots based on rules, tend to be limited in functionality, and are as smart as chatbots are programmed to be. On the other end, a chatbot that uses artificial intelligence, understands language, not just commands, and continuously gets smarter as it learns from conversations it has with people.

II. LITERATURE SURVEY

2.1 Keyword Search in Databases: The Power of RDBMS

To reduce the number of tuples to be searched from based on the keyword given, and hence reduce the time of response, they prune tuples that do not participate in any of the results using SQL in the primary reduction step. As a part of the join operation, they process the relational algebra expressions using SQL over the reduced relations. This in turn makes more use of the RDBMS structure as this process is middleware free. The existing work [7] is categorized into two approaches: Schema based and Schema free approach. In Schema Based Approach, there are two tasks: Candidate Network Generation and Candidate Network Evaluation. In Candidate Network Generation they generate all candidate networks (relational algebra expressions) as it does not make sense if two tuples are far away in an interconnected tuple structure. In Candidate Network Evaluation they evaluate all candidate networks using SQL.The schema-free approach support keyword search using graph-based in-memory algorithms by considering RDB as a graph. Both these approaches do not fully utilize the functionality of RDBMSs. This paper proposes a middleware free approach to support three types of keyword queries to find the three different interconnected tuple structures (all connected trees up to certain size, all sets of tuples that are reachable from a root tuple within a radius , and all the sets of multi-centre sub graphs within a radius) in the same framework on RDBMSs .They take a tuple reduction approach using SQL without additional new indexing to be built and maintained, without any precomputing required. To compute all connected trees, they propose a new approach to prune tuples that do not participate in any resulting connected trees and then do query processing over the reduced relations. To compute all the multi-centre sub-graphs, they propose a new three way reduction approach to effectively prune tuples from relations and then perform query processing over the reduced relations. The paper discusses the keyword search based on three different semantics - connected tree semantics (an m-keyword query that finds connected tuple trees), distinct root (an m-keyword query that finds a collection of tuples that contain all the keywords reachable from a root tuple (centre) within a given distance) semantics and distinct core semantics (an m-keyword query that finds multi-centre sub graphs known as communities).

Merits: Middleware free approach that makes it possible to fully utilize the functionality of RDBMSs to support keyword queries in the same framework of RDBMSs.

Demerits: Cannot process large number of SQL statements efficiently.

Future Work: To further study SQL query optimization to process a large number of SQL statements efficiently and to extend the approach to compute top-k interconnected tuple structures that are based on scoring and ranking.

2.2 Top-K Keyword Query in Relational Databases.

Relational databases these days store text data and hence there is a greater demand for keyword search. Existing algorithms focus on retrieving results from within the same relations. Also that the existing algorithms allow casual users to search databases, as it does not require users to have knowledge about query languages or database schema. This paper proposes a non-trivial ranking method [9] that adapts the state-of-the-art IR ranking methods to run heterogeneous joined results of database tuples. Three algorithms are proposed. These are skyline sweeping algorithm, block pipeline algorithm and tree pipeline algorithm that aggressively minimize database probes and maximize computational sharing. The skyline sweeping algorithm achieves a minimal database probing by using a monotonic score upper bounding function for the new ranking formula. The block pipeline algorithm further reduces unnecessary database accesses. The tree pipeline algorithm is then proposed that share the intermediate results among CNs at a fine granularity. Extensive experiments were conducted on large scale real database. The result of the top-k keyword query is a tree, T, of tuples, such that each leaf node of T contains at least one of the query keyword, and each pair of adjacent tuples in T is connected via a foreign key to primary key relationship. Such a tree is called a joined tuple tree .This paper models the joined tuple tree as a virtual document. Each joined tuple tree belongs to the results produced by a relational algebra expression. Such a relational algebra expression is also termed as a Candidate Network (CN).The size of the joined tuple tree is the number of tuples in the tree. The relations in the CN are also called tuple sets. The entire results produced by a CN are modelled as a document collection. There are two kinds of tuple sets: those that are constrained by keyword selection conditions are called non-free tuple sets and others are called free tuple sets. Every joined tuple tree has its relevance score which indicates how relevant the joined tuple tree is to the query. All the joined tuple trees of a query will be sorted in descending order of the scores and only those with top-k highest scores will be returned in minimal database access. Their ranking method achieved high precision with high efficiency to scale to databases with tens of millions of tuples. There is a significant improvement in terms of retrieval effectiveness and efficiency of top-k keyword queries over RDBMS.

Merits: This ranking method achieved high precision with high efficiency to scale to databases with tens of millions of tuples.

Future Work: A new plan for optimization technique using polynomial time algorithm.

2.3 Blinks: Ranked Keyword Searches on Graphs

A top-k keyword search query on a graph finds top k-answers according to some ranking criteria, where each answer is a substructure of the graph containing all query keywords. These techniques have a few drawbacks like worst case performance, not taking full advantage of indexes and high memory requirements. To overcome this problem they proposed BLINKS [5], a bi-level indexing and query processing scheme for top-k keyword search in graphs. It follows a search strategy with provable performance bounds and additionally giving a bi-level index for pruning and accelerating the search. The BLINK's partitions data graphs into blocks to reduce the index space. This offers an order of magnitude performance as an improvement over existing approach. BLINKS provide a better search strategy which is based on a new policy called cost expansion that alleviates many problems of the original backward search strategy. Another feature of this paper is the idea of combining indexing with search. This index significantly reduces the runtime cost of implementing the backward search strategy. At the same time, this index makes forward search possible too, effectively making the search bidirectional. Yet another noticeable feature is partitioning-based indexing. As mentioned earlier, BLINKS partitions the data graph into multiple sub graphs, or blocks. The bi-level index stores summary information at the block level to initiate and guide search among blocks, and more detailed information for each block to accelerate search within a block. This paper focuses to find out the top-ranked answers to a query. The goodness of the answer is measured by a scoring function, which maps an answer to a numeric score; the higher the score the better the answer. This scoring function has two properties: Match-distributive semantics and Graph-distance semantics. In Match-distributive semantics, the net contribution of matches and root-match paths to the final score can be computed in a distributive manner by summing over all matches. In Graph-distance semantics, the score contribution of a root-match path is defined to be the shortest distance from the root to the match in the data graph, where edges have non-negative distances.

Merits: BLINKS improves the query performance by more than an order of magnitude.

Future Work: Index maintenance is an interesting direction.

2.4 A Framework for Evaluating Database Keyword Search Strategies

Their comparison of 9 state-of-the-art keyword search system contradicts the retrieval effectiveness supported by existing evaluation and reinforces the need for standardized evaluation. The existing database is greater than this threshold and hence the results motivate the creation of new algorithm and indexing techniques that scale to meet workloads. In this paper they evaluate the effectiveness of keyword search system for relational database [8]. The evaluation presented in this paper is used to compare a wide variety of IR scoring and proximity search techniques. To measure the effectiveness of search systems, four metrics are used. The number of top-1 relevant results is the number of

queries for which the first result is relevant. Reciprocal rank is the reciprocal of the highest ranked relevant result for a given query. Both of these measures tend to be very noisy but indicate the quality of the top-ranked results. Average precision for a query is the average of the precision values calculated after each relevant result is retrieved (and assigning a precision of 0.0 to any relevant results not retrieved). Mean average precision (MAP) averages this single value across information needs to derive a single measure of quality across different recall levels and information needs. The evaluation presented in this paper is the first to compare a wide variety of IR scoring and proximity search techniques. Their results indicate that no existing scheme is best for search effectiveness, which contradicts previous evaluations that appear in the literature. The sets of results retrieved by different systems are not highly correlated, which indicates that performance is not the sole factor that differentiates these systems.

Merits: Improves the execution time. The ranking of keyword and not requires the knowledge of database queries.

Future Work: To include additional datasets and query workloads. To re-examine our design decisions for our evaluation (e.g., binary relevance assessments) and to include additional metrics (e.g., normalized discounted cumulative gain (nDCG)) for evaluating search effectiveness.

2.5 QUNITS: Queried Units for Database Search

In this approach the database querying and ranking are separated. Now the problem is divided into two tasks. The first task is to represent the database as a collection of independent queried units (qunits), each of which represents the desired result for some query against the database. The second task is to evaluate keyword queries against a collection of qunits, which can be treated as independent documents for query purposes, which further permits the use of standard IR techniques. Queries are first processed to identify entities using standard query segmentation techniques. Qunits derivation can be done using schema and data, external evidence and using query logs [13]. The benefit of maintaining a clear separation between ranking and database content is that structured information can be considered as one source of information amongst many others. This makes our system easier to extend and enhance with additional IR methods for ranking, such as relevance feedback. Additionally, it allows us to concentrate on making the database more efficient using indices and query optimization, without having to worry about extraneous issues such as search and ranking. For the front end keyword query, the structured database is a collection of independent qunits. Hence standard information retrieval techniques can be applied to choose the appropriate qunits, rank them, and return them to the user. In case of structured database backend, each qunit is a view definition, with specific instance tuples in the view being computed on demand. As a result, there is no issue of underspecified or vague queries.

Future Work: to be able to substantially improve upon the qunit finding and utility assignment algorithms presented above; To deal with qunit evolution over time as user interests mutate during the life of a database system; To extend qunit notions to databases with substantial mixed text content and to use IR techniques to query the text content in conjunction with the database structure.

2.6 Efficient IR-Style Keyword Search over Relational Databases

Commercial RDBMSs generally provide querying capabilities for text attributes that incorporate state-of-the-art information retrieval (IR) relevance ranking strategies. But this search functionality requires that queries specify the exact column or columns against which a given list of keywords is to be matched. This requirement can be complex and inflexible from a user perspective. Good answers to a keyword query might need to be "assembled", by joining tuples from multiple relations. This observation has motivated recent research on free-form keyword search over RDBMSs. A key contribution of this paper is the incorporation of IR-style relevance ranking of tuple trees into our query processing framework [1]. This scheme fully exploits single-attribute relevance-ranking results if the RDBMS of choice has text-indexing capabilities (e.g., as is the case for Oracle 9.1). By leveraging state-of-the-art IR relevance-ranking functionality already present in modern RDBMSs, we are able to produce high quality results for free-form keyword queries. For example, a query [disk crash on a net vista] would still match the comments attribute of the first Complaints tuple above with a high relevance score, after word stemming (so that "crash" matches "crashed") and stop-word elimination (so that the absence of "a" is not weighed too highly).Modern RDBMSs already include IR-style relevance ranking functionality over individual text attributes, which they exploit to define their ranking scheme. Specifically, the score that they assign to a joining tree of tuples T for a query Q relies on:

- Single-attribute IR-style relevance scores Score (ai, Q) for each textual attribute ai ∈ T and query Q, as determined by an IR engine at the RDBMS, and
- A function Combine, which combines the single-attribute scores into a final score for T.

Unfortunately, existing query-processing strategies for keyword search over RDBMSs are inherently inefficient, since they attempt to capture all tuple trees with all query keywords. Therefore, this paper also discusses efficient query processing techniques for IR-style queries over RDBMSs. These techniques produce the top-k matches for a query –for moderate values of k– in a fraction of the time taken by state-of-the-art strategies to compute all query matches. Furthermore, our techniques are pipelined, in the sense that execution can efficiently resume to compute the "next-k" matches if the user so desires. The query model here has three components: IR Engine, Candidate Network Generator and the Execution Engine. Given a set of CNs together with a set of non-free tuple sets, the Execution Engine needs to

efficiently identify the top-k joining trees of tuples that can be derived. We have three algorithms for this namely, Naive algorithm, sparse algorithm, Single-Pipelined algorithm, Global-Pipelined algorithm and Hybrid algorithm. Sparse is the most efficient algorithm for queries with relatively few results, while Global Pipelined performs best for queries with a relatively large number of results. Hence, it is natural to propose a Hybrid algorithm that estimates the expected number of results for a query and chooses the best algorithm to process the query accordingly. This hybrid algorithm has the best overall performance for both AND and OR query semantics.

Merits: A query in this model is simply a list of keywords, and does not need to specify any relation or attribute names and this paper produces an efficient query processing model.

2.7 Effective Keyword Search in Relational Databases

Since the size of text data available in relational database increases exponentially, the need for ordinary users to search such information is dramatically increasing. Even though the major RDBMSs have provided full-text search capabilities, they still require users to have knowledge of the database schemas and use a structured query language to search information. This becomes difficult for normal users. Keyword search in RDBMS has recently reached heights, inspired by the huge success of IR style keyword search [3]. Due to the difference in the basic answer unit between document searches and database searches, in relational databases, there is a need to assign a single ranking score for each tuple tree, which may consist of multiple tuples with text columns, in order to rank the answers effectively. In relational databases, there are three key steps for processing a given keyword query. (1) Generate all candidate answers, each of which is a tuple tree by joining tuples from multiple tables. (2) Then compute a single score for each answer. The scores should be defined in such a way so that the most relevant answers are ranked as high as possible. (3) And finally return answers with semantics. This paper focuses on search effectiveness. They propose a novel ranking strategy that ranks answers (tuple trees) effectively and returns answers with basic semantics. This strategy can be used both at the application level and be incorporated into a RDBMS to support keyword search in relational databases. They identify four factors that affect search effectiveness. These are Tuple Tree Size Normalization, Document Length Normalization, Document Frequency Normalization and Inter-Document Weight Normalization. This strategy also uses phrase-based and concept-based models to improve search effectiveness further. And the concept-based model can also return answers with semantics.

Merits: Phrase-based search and concept-based search improve effectiveness significantly. This approach not only can be used at the application level for keyword search in relational databases, but also can be incorporated into the core of a RDBMS.

Future Work: To utilize link structures (primary key to foreign key relationships as well as some hidden join conditions), and some non-text columns (for example, user review rates on a product, box-office income of a movie, and whether an actor won an Oscar award) in the relational databases along with pure text data. To improve search effectiveness further by combining the above. To investigate the efficiency issue. To conduct experiments with more real world databases (for example, movie databases, academic paper databases, product databases and job databases) and more user queries.

III. SYSTEM ANALYSIS

3.1 Problem Statement

The problem associated with existing system is that all the chatbots will have a predefined purpose. The functionalities of each are limited to a specific area. And when it comes to processing of large amount of data, the processing time increases as the size of the data increases. This leads to an unpleasant way of receiving answers from the bot.

3.2 Problem Objective

The main problem objective is that existing chatbot is not in a conversational style. Our chatbot called Rattata is been designed using combination of machine learning and natural language processing. Most popular text mining and classification methods have adopted. Also, the training process is done on multi-node hadoop cluster by considering large raw English dataset. With multi-node hadoop cluster there was a large reduction in the total running time. In this work, we present an innovative model for relevance feature discovery. It gives a conversational style chatbot experience.

3.4 Proposed System

Sample dataset is been collected and used for the retrieval of the query processing. This proposed system is designed in a way that, it is scalable to any kind of dataset as the machine learning and natural language processing remains the same to all kind of datasets. With the queried keyword, the search is been carried out on the matching clusters and results are been retrieved. The system provides a conversational style query processing. Authorized entity to determine whether or not to accept the system.

IV. SYSTEM DESIGN AND IMPLEMENTATION

4.1 Chatbot UI Design

The UI is designed to be in conversational style. It is built using Html, CSS, JS and Bootstrap. The UI takes input queries from the end user. The responses from Rattata, the bot pops up as soon as it arrives. The chat can be navigated with ease. Additional features like Reminders, Searching the web are also available.

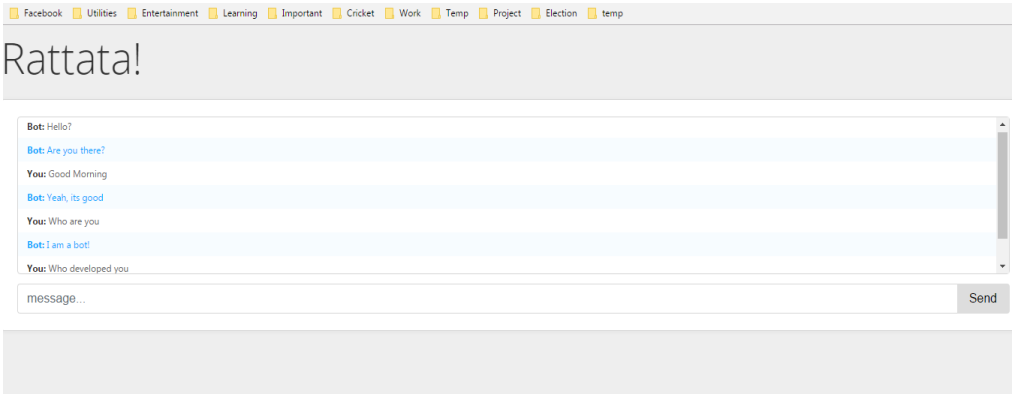

Fig. 1 The UI design of the chatbot in a conversational style

4.2 The Dataset Used for Rattata

The Dataset is a Stack Exchange data on the English Language. The Data is in the XML format. It contains the various questions and answers posted by the users on the Stack Exchange Site. The dataset includes the posts history, comments, important tags, badges received by the users.

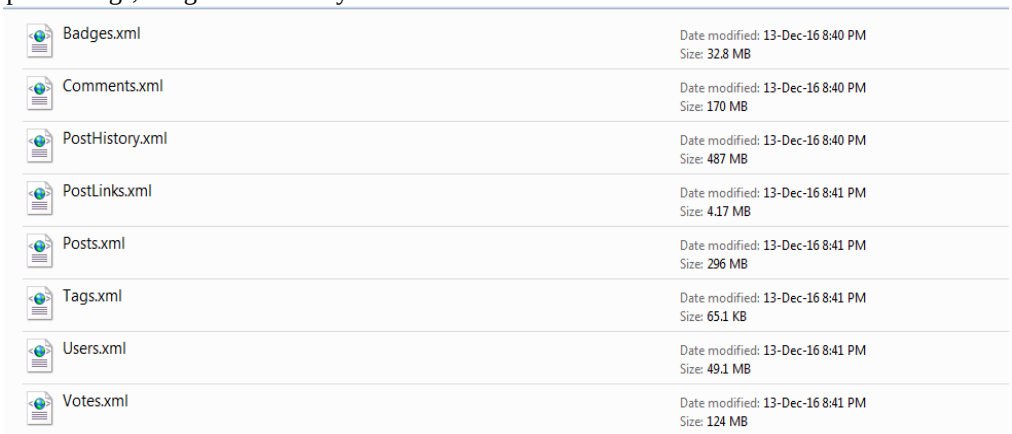

Fig. 2 Dataset listed in XML format

4.3 Xml Data Parsing

The XML data has to be parsed to get required data. Therefore the parsed data is written into a text file. This is done using the SAX Parser in Java.

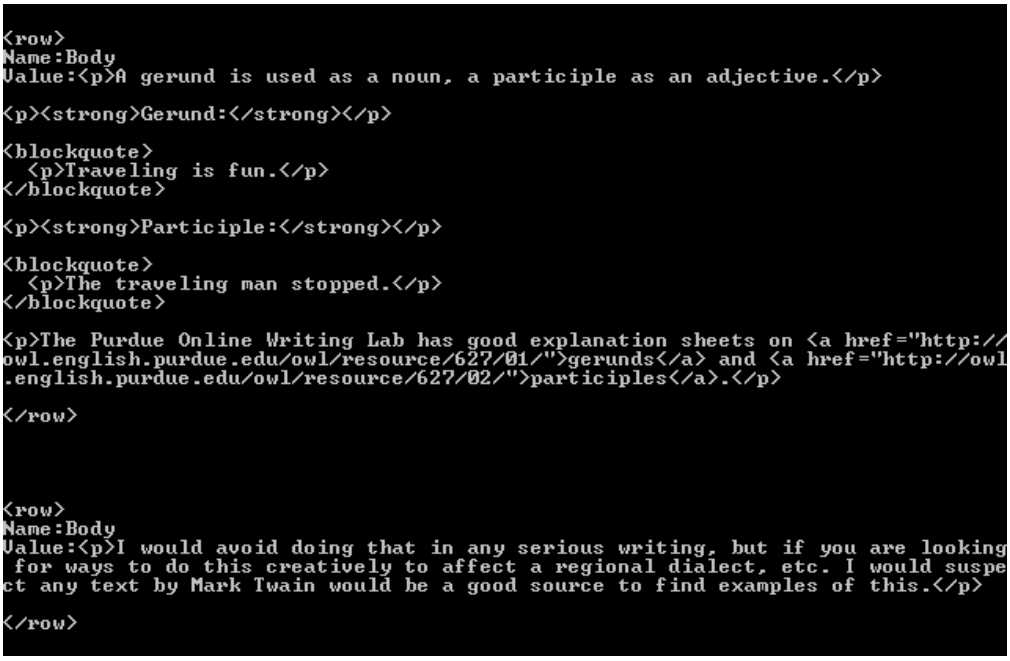

Fig. 3 Parsed XML data

4.4 Data In CSV Format

The text file is converted into CSV (Comma Separated Values) format for further processing using R language. After this stage the data are made into different clusters using K-Means algorithm based on frequent item set.

	Body	X2	X3
1	When should I use can? When should I use could? Wh...	NA	NA
2	Doesn't "quint" mean "five"? What does that have to d...	NA	NA
3	Which is the correct use of these two words, and in ...	NA	NA
4	It's the fifth element after earth, air, fire, and water, s...	NA	NA
5	I think most folk happily use either "while" or "whilst"....	NA	NA
6	I may not be coming in tomorrow... I might not be co...	NA	NA
7	I feel uncomfortable saying sentences like the follo...	NA	NA
8	quint means fifth. The "quintessence" is the fifth ess...	NA	NA
9	It seems they are interchangeable, but whilst is prim...	NA	NA
10	http://en.wikipedia.org/wiki/Shall_and_will Shall and...	NA	NA
11	Essence in this context is a synonym for "element", a...	NA	NA
12	It is rarely appropriate to use contractions in a formal...	NA	NA
13	Are there any concrete rules that say which words (n...	NA	NA

Fig. 4 Data in CSV Format

V. RESULT AND DISCUSSION

The data is clustered in order to reduce the processing time. Clustering is a technique in which a large amount of data is grouped into categories based on the similarity. This is done with the help of K-means Clustering algorithm. K-means is used to cluster the data based on the Euclidian distance. The data which is clustered is represented in the form of word cloud and plotting graph.

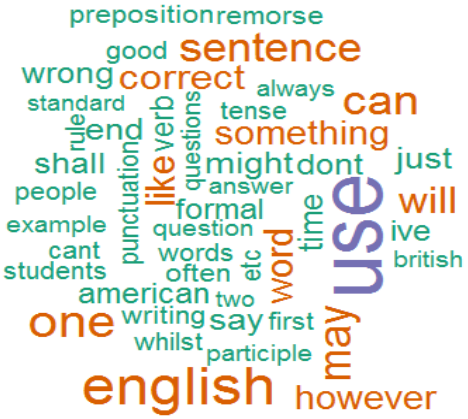

Fig. 5 Word Cloud

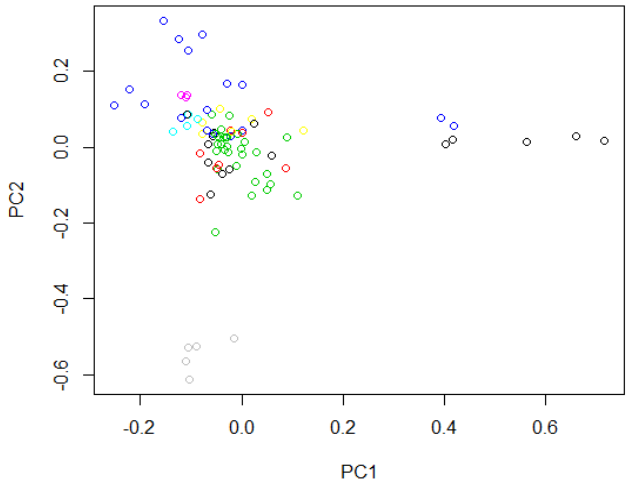

Fig. 6 Graph in R for given parsed data

VI. CONCLUSION AND FUTURE WORK

The Chat Bot -Rattata has been done with a conversational style. Expected clusters are obtained using K-means algorithm. The result indicates effective way of clustering the data which reduces the processing time. It provides good user experience. It gives high speed up and efficiency. It has been constructed using Machine Learning and Pattern Matching Concepts. In our future work, the query is processed based on the user input using NLTK.

ACKNOWLEDGMENT

We acknowledge all authors whose work gave knowledge to us. Thanks a lot.

REFERENCES

[1] Sanket S.Pawar, Abhijeet Manepatil, Aniket Kadam, Prajakta Jagtap "Efficient IR-Style Keyword Search over Relational Databases".
[2] Varun Kacholia , Shashank Pandit , Soumen Chakrabarti, S. Sudarshan,
[3] Rushi Desai, Hrishikesh Karambelkar, "Bidirectional Expansion for Keyword Search on Graph Database"
[4] Fang Liu, Clement Yu, Weiyi Meng, Abdur Chowdhury, "Effective Keyword Search in Relational Databases"
[5] Bolin Ding, Jeffrey Xu Yu, Shan Wang, Lu Qin, Xiao Zhang, Xuemin Lin3, "Finding Top k-Min Cost Connected Trees in Database"
[6] Hao Hey, Haixun Wangz, Jun Yangy Philip S. Yuz , "BLINKS ranked keyword searches on graphs"
[7] Bhavana Bharat Dalvi, Meghana Kshirsagar, S. Sudarshan, "Keyword Search on External Memory Data Graphs"
[8] Lu Qin, Jeffrey Xu Yu, Lijun Chang, "Keyword Search in database: The Power of RDBMS"
[9] Joel Coffman , Alfred C. Weaver, "A framework for Evaluating Database Keyword Search Strategies"
[10] Yi Luo, Wei Wang, Xuemin Lin, Xiaofang Zhou, Jianmin Wang, Keqiu Li, "Top-K Keyword Query in relational database"
[11] Joel Coffman, "Toward Practical Relational Keyword Search Systems"
[12] Joel Coffman, Alfred C. Weaver, "An Empirical Performance Evaluation of relational keyword search systems"
[13] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, "DBXplorer- A System for Keyword-Based Search over Relational Databases"
[14] Arnab Nandi, H.V. Jagadish, "Qunits- Quried units for database"

