Introduction:

A Question-Answering (Q&A) system receives the users' questions posed in natural language and provides them with automatic answers. It's an information retrieval system that returns the exact satisfying answers instead of a collection of documents.

In order to design such a computer system, multidisciplinary research should be conducted within the domains of Natural Language Processing (NLP), Machine Learning, Data Mining and information Retrieval.

The first Q&A systems were Green's "BASEBALL", built in 1961, and "LUNAR", 1977, which were effectively able to answer questions about the US baseball league and the geological analysis of rocks returned by the Apollo moon missions, respectively. The development of "comprehensive theories" in computational linguistics goes back to 1970s and 1980s which has led to the NLP projects such as Q&A systems.

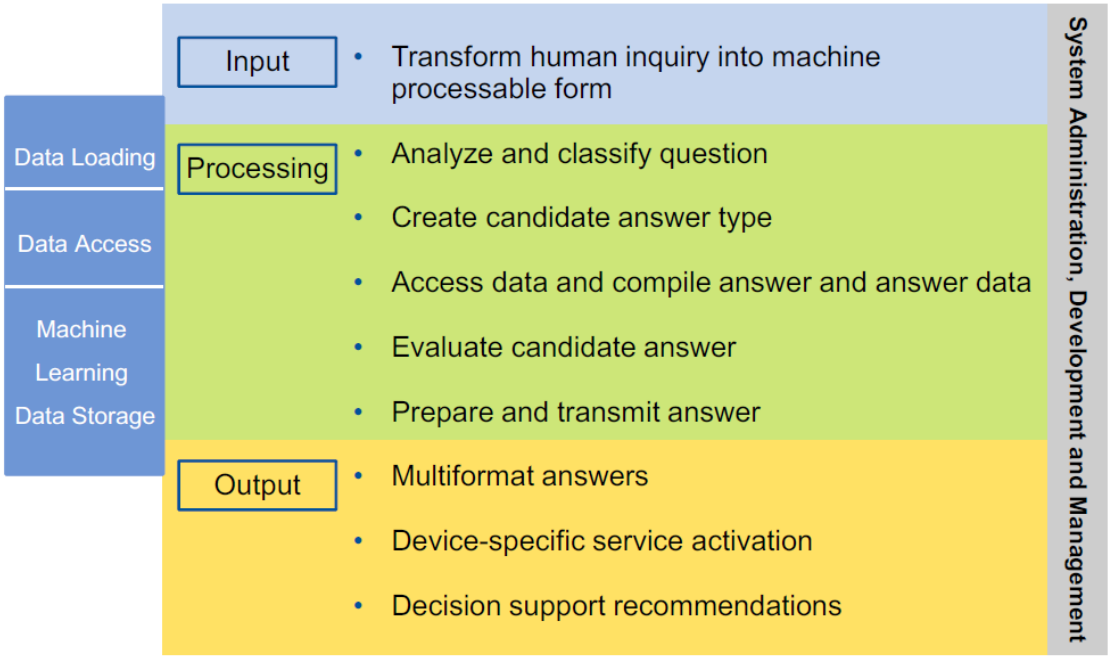
Some significant achievements in this domain include, but are not limited to, Google/Google Now, Apple's Siri, IBM's Watson, Wolfram Alpha, START and AskMSR.

Question-Answering (Q&A) system could offer the followings [1]:

1. A human natural language interface
2. Direct answers instead of a collection of documents
3. 'Derived information and inferences':
4. 'Answer triggers actions: System can be programmatically link to other applications so that answers led directly to actions.'

At SAP Research lab, we decided to contribute in the progress of this ambitious project and apply our knowledge in Machine Learning, NLP and Information Retrieval to tackle one of the needs in our company. The Fuzzy Adventure's Q&A System is designed to intelligently address some of the managers' questions in IMS department. The system receives the questions in English Natural Language and after some processing creates the related query and finally returns the proper answer.

Following figure depicts the different levels of a Q&A system which is Input, Processing and Output [1].



Our system also offers an alternative solution to return answers to managers' questions which gives them the option to select a question among a list of provided questions. This alternative serves as a backup in case the system would not understand the posed question.

Question Selection Process:

After having a few interviews with the IMS managers, we were able to collect 12 main questions which address some of their concerns. These questions are listed below. The unknown entity is left as blank.

1- How many messages (employee ID or employee Name) _____ closed.
2- How many messages (employee ID or employee Name) _____ touched.
3- How long a message stays in the SAP side?
4- How many escalated messages (employee ID or employee Name) _____ has worked on.
5- What is the average MPT when the messages get forwarded to our queue?
6- How experienced is (employee ID or employee Name) _____ ?
7- When (employee ID or employee Name) _____ started working on (name of a component) _____ component.
8- Give me the list of the open messages for (name of the topic)_____ topic.
9- Give me the list of the messages without a processor.

10- Give me the list of all the messages with (name of the flag) _____ flag.
11- Who is the most productive person on my team?
12- What component contributes the most to my backlog?

System Overview:

The system includes the following modules:

1) Natural Language Processing Module and its interface to the database (NLIDB):

    a. This module receives the question. The question could be posed in different ways. Our system creates the parse tree and part of speech tagging for each question. We have developed an algorithm to choose the most informative words in each question which could represent the whole question and we refer to them as 'keywords'.
    b. The selected keywords will then be sent to our context-based glossary to be checked and replaced with a common label that replaces all its synonyms. Our context-based glossary was created from the online surveys which were spread among some SAP employees/managers and is accessible through our Github page under the 'context-based data' directory.
    c. The next step will be to apply some manually added rules to our keywords so as to make the system more accurate.
    d. The selected keywords will be fed into our Natural Language Interface to Database (NLIDB) module which its job is to map the returned keywords to the tables and fields in our database. In other words, our NLIDB module creates the semantic net between the keywords and the database.
    e. Finally, the NLP-NLIDB's output is a quintuplet of
        i. List of keywords
        ii. Related table names
        iii. Related fields in the tables
        iv. Proper nouns found in the question
        v. List of the conditions

2) SQL Converter Module:

Our system addresses 12 questions in total and they could be asked in different ways in English natural language and this was a limitation that needed to be considered before we chose our SQL Converter's architecture. Various architectures which could be used for SQL conversion module were studied, such as Pattern Matching, Hybrid of Pattern Matching and Relational Rules, Syntax-Based Systems, Semantic Grammar Systems, Intermediate Representation Language and Function-based Systems. For more information and the advantages and disadvantages of each of these systems, please refer to [2].

Every Machine Learning System needs some training data in order to learn the existing patterns and apply it to the unseen cases. Since there didn't exist any training data for the managers' questions, we conducted four different surveys and asked the employees who are

familiar with the domain to rephrase the questions in four different ways. The result was a dataset consisting of 286 questions and a domain specific glossary which helps the system to relate the similar words. For more information please refer to the Survey's section in this document.

Considering the limited number of questions we needed to tackle and small dataset that we were able to collect, we chose the "Hybrid of Pattern Matching and Relational Rules" method. SQL statements being directed to our system had a dynamic nature. Therefore, we designed some SQL templates which could be completed in real-time.

Fuzzy Adventure system's SQL Conversion Module consists of two sub systems; Template Selector and Term Selector.

a. Template Selector: The number of SQL templates corresponds to the number of questions. There exist a one-to-one relationship between the type of the question and the template. In case that we need to deal with more number of questions in the future, we could design more flexible templates that could cover many-to-one relationships between many questions and one template.

   Extracted keywords from the NLP-NLIDB modules are then vectorized and used for the system's training purposes. We conducted different experiments with various supervised machine learning algorithms and parameters such as Naïve Bayes, Support Vector Machines, Linear Regressions and Logistic Regressions. Logistic Regression returned the best results.

   The dataset was divided for training and testing purposes. In training stage, system learns each question and its corresponding SQL template and in testing (execution) stage a question will be matched to the most probable template. Our system returns the right SQL template in 82%-95% of the cases.

b. Term Selector: Once the template is selected, we need to fill the values dynamically. In general the questions we tackle, either ask about a specific employee or a specific component. The relevant information is among the list of keywords being fed to the SQL module from the NLP-NLIDB module. Our current version is simply trying every permutation of every keywords returned by the NLP module into the template. Those filled templates are then sent to the database. Most of them do not return an answer because they simply don't make sense, but one of the permutations has the right answer. Since only one permutation makes sense, it is easy to select it because it's the only query that the database has an answer for. Theoretically, there may be more than one combination that returns the answer but we never faced such problem in our experiments. Solution to this problem could be considered in future improvements.

3) Graphical User Interface:

Fuzzy Adventure's GUI is a Django web application, using twitter bootstrap for the front end.

The following figure depicts an overview of our system. We explain the system with an example. Consider the question being posed as:

Q: "**Who is the most productive employee on my team?"**

- **Syntactic Parser :** This question will be syntactically parsed since our keyword_Extraction algorithm extract some keywords based on their roles in the sentence

  (ROOT (SBARQ (WHNP (WP Who)) (SQ (VBZ is) (NP (NP (DT the) (ADJP (RBS most) (JJ productive)) (NN employee)) (PP (IN on) (NP (PRP$ my) (NN team)))))))

- **Keywords extraction and Category Identification:** keyword_Extraction algorithm extract some keywords. In this case:

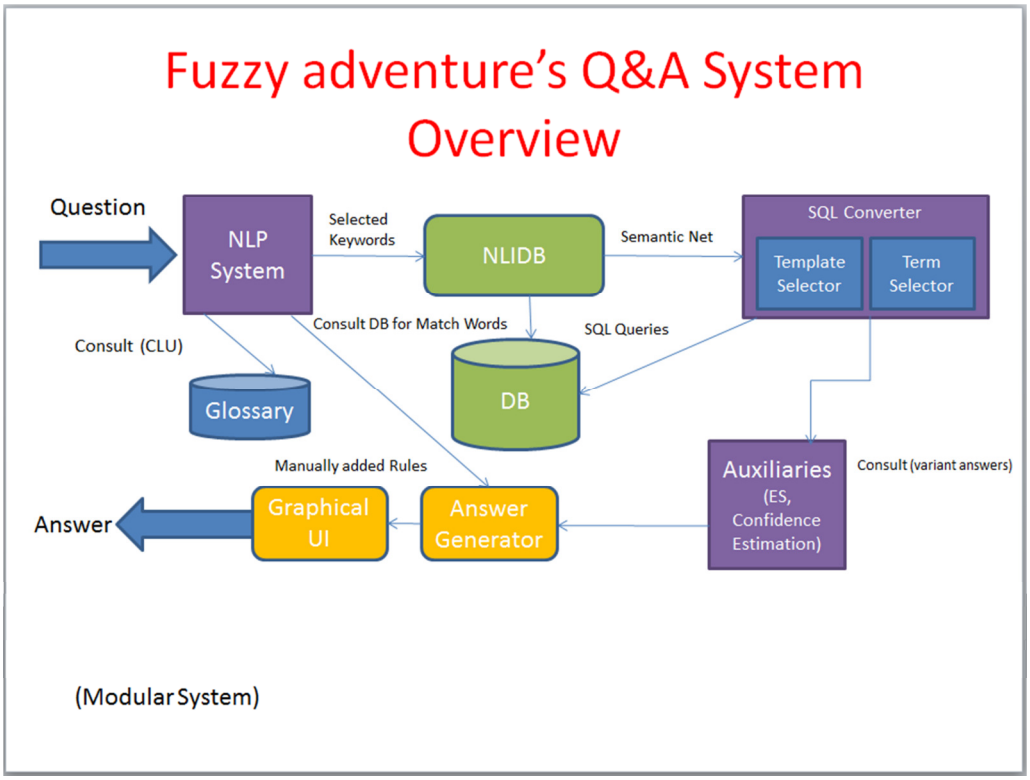
  - Key words: most productive, on team
  - Category: Looking for = Employee

- **Lexical Cohesive Units analyzer:** Checking if a group of words imply a specific meaning by consulting the glossary. For instance by consulting the glossary we figure out that "Team member" means employee.

- **Semantic Analyzer:** Consulting the WordNet, the linguistic analyzer looks for the entity label that also matches the fields in database. For instance, the system replaces any word within the list of {person, member of the team, team member, programmer, contributor, developer} with the word "employee"

- **NLIDB**'s job is to map the words in the extracted keywords with the tables and fields in database and creating the relationship between the attributes (semantic net).

- **SQL Converter** then receives the quintuplet from NLIDB and sends the appropriate query to the database.

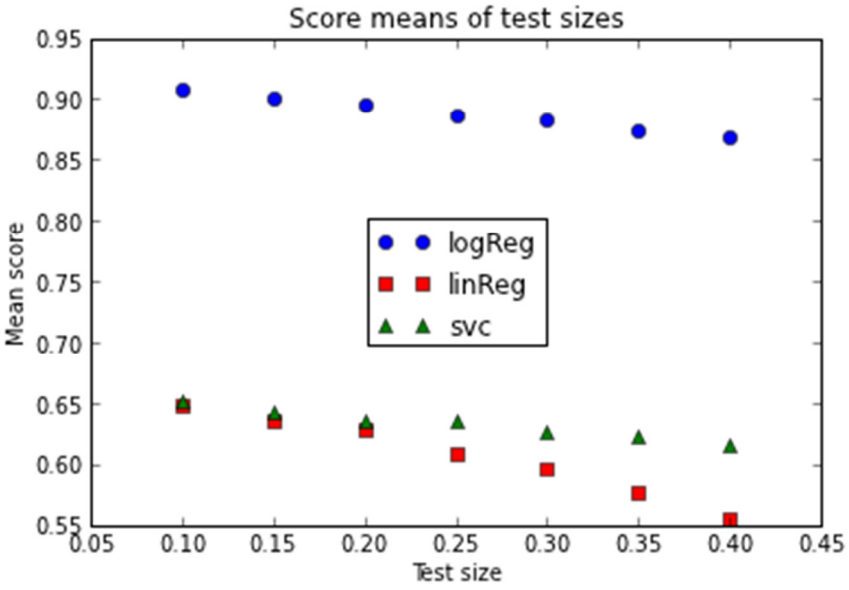


Experiments and Results:

To choose the right template, various machine learning approaches such as Logistic Regression, Linear Regression, Support Vector Machines and Naïve Bayes examined. To train and test the system, the list of selected keywords was used as a vector of features which were explained in previous sections.

Following figure shows the results we gained by applying Logistic Regression, Linear Regression and Support Vector Machines. Each of these models ran through multiple simulations:

1. The set of questions was separated between training and testing. The proportion varied from 90% for training and 10% for testing to 60% for training and 40% for testing by steps of 5%.
2. Every model ran 500 times for every test/train split.
3. Scores are averaged over the test sets of the same size.



As we can see, logistic regression has a higher precision comparing to the Linear Regression and Support Vector Machines. Considering the size of our question set we didn't explore any further with more models. We were satisfied with the result for the moment as it would provide a user experience close enough to perfection for this project.

Future Work:

Considering the limited number of questions we needed to tackle and the small dataset that we were able to collect, we chose the "Hybrid of Pattern Matching and Relational Rules" method which presents an acceptable performance. However, our next milestone would be to improve our SQL converting modules to generalize the system and broaden the number of questions.

For more information about different NLP to SQL conversion methods refer to our presentation in [2].

Constraints and Limitations:

The main constraint we are facing during the course of our system development is the data accessibility through the CSS application. The problem still persists but we are trying to solve the problem.

References:

1. Popkin, Jamie, "Google, Apple Siri and IBM Watson: The Future of Natural-Language Question Answering in Your Enterprise", June 2013.
   More info: jens.fuchs@sap.com
2. [link to NLP-SQL Converter.pdf]