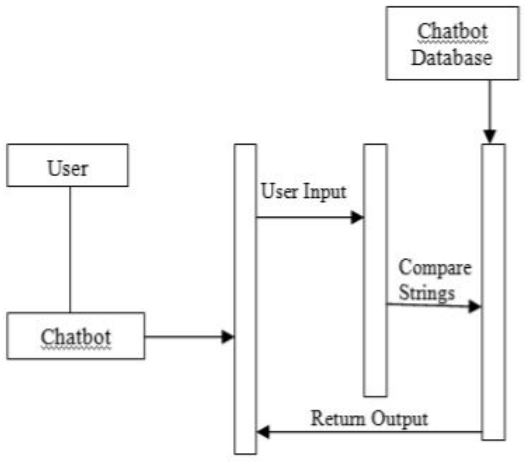
A Tool of Conversation: Chatbot

M. Dahiya

Summary

The paper is basically about implementation of a chatbot based on pattern matching . A chatbot is defined as a computer program that converses with the user. The program for the chatbot is written in java using java applets for dialogue box. The Chatbot is created in such a way to help the user, improve the communication and amuse the user. Pattern matching is a technique of artificial intelligence used in the design of this Chatbot. The input is matched with the inputs saved in the database and corresponding response is returned.The chatbot just answers to the questions asked by the user, if the question is found in the database.

Simple design of a chatbot can be represented by the following diagram.



A two dimensional string arrays are created to build a database. All the even rows contain the request or questions and all the odd rows contain the response or answers. Columns in the array are created to save different types of questions that could be asked by the user and responses that a Chatbot can answer. There is one row in the array which contains default responses which is used when the

matching question is not found in the array.

The input from the user is taken. All the punctuation marks in the user's input are removed. All the uppercase letters are converted to lowercase. A variable called response is used to hold a byte value and it is set to 0. If the response is 0, the match for the input is found in the database and it is returned as a response which is displayed in the text area. If the response is 1, then the match for the input is not found in the database. In this case, a default response is returned. Random () function is used to choose the response saved in the database.

The chatbot described in this paper is actually a very simple and not as complicated as other chatbots as it uses simple pattern matching to represent the input and output , If the input is not found in the database, a default response is generated.

A Lightweight Stemmer for Hindi

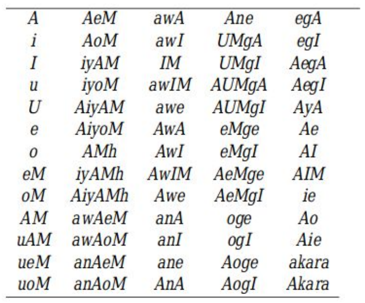
Ananthakrishnan Ramanathan, Durgesh D Rao

Summary

This paper basically discusses about a stemmer for hindi language. Stemmer is a tool which finds the stem (root) of every word input to it so that similar terms get the same root after stemming. The formal term is ‘conflation’ which is used to denote the act of mapping variants of a word to a single stem. Stemming helps in improving performance and is used in Information Retrieval Systems. The advantage of it is that it decreases the size of the index files in the IR system because many terms get mapped to a single stem.

The paper talks about various approaches to stemming. One approach is to store all possible index terms and their stems in a table and then stem terms through looking up the table. But such data is usually not available. Also, it restricts stemmer to the data given in the table which makes the stemmer domain dependent, which should not be the case. Some other approaches use statistical measures to conflate terms. The last approach, which is also described in this paper, is to perform stemming by removing prefixes and suffixes. More specifically, the stemmer strips suffixes from a list of suffixes on a longest match basis and is non-iterative.

Hindi is a highly inflectional language and is, relatively, a free word order language. The relations between various components of a sentence is shown by using postpositions. Nouns are inflected to show case information and verbs are inflected to show gender, number and person information. The paper then discusses various suffixes that must be removed by showing examples of words having those suffixes and the various exceptions to it. Inflections corresponding to nouns, adjectives and verbs are described. The final suffix list they reached upon was:



A	*AeM*	*awA*	*Ane*	*egA*
i	*AoM*	*awI*	*UMgA*	*egI*
I	*iyAM*	*IM*	*UMgI*	*AegA*
u	*iyoM*	*awIM*	*AUMgA*	*AegI*
U	*AiyAM*	*awe*	*AUMgI*	*AyA*
e	*AiyoM*	*AwA*	*eMge*	*Ae*
o	*AMh*	*AwI*	*eMgI*	*AI*
eM	*iyAMh*	*AwIM*	*AeMge*	*AIM*
oM	*AiyAMh*	*Awe*	*AeMgI*	*ie*
AM	*awAeM*	*anA*	*oge*	*Ao*
uAM	*awAoM*	*anI*	*ogI*	*Aie*
ueM	*anAeM*	*ane*	*Aoge*	*akara*
uoM	*anAoM*	*AnA*	*AogI*	*Akara*

Figure 3: Suffix List

The paper then goes on to evaluate this stemmer and a stemmer was evaluated correct if words that are morphological variants do conflate to a single stem and the words that conflate to a single stem are actually morphological variants.

It describes two possible errors: Overstemming and Understemming. Overstemming is when words that are not morphological variants are also conflated. Understemming is when words that are morphological variants do not get conflated to a single stem. They calculated the error percentages regarding Overstemming and Understemming and they were found to be 13.84% and 4.68% respectively.

The paper then finally talks about the possibility of extending this stemmer for other Indian languages too.

A New Model for Question-Answer based Dialogue System for Indian Railways in Hindi Language

Lovely Sharma, Vijay Dhir and Kamaljeet Kaur

Summary

This paper basically describes a Question-Answering System in a restricted domain with limited data. The restricted domain is Jalandhar District. This dialogue system will help the user to get answers to questions specific to the domain like any information regarding fare, ticket availability, different trains going from one station to another.

The paper talks about the early dialogue systems like BASEBALL and ELIZA which were chatbots and conversed with human. It also talks about a Mathematical Question Answering System called ‘Wolfram Alpha’ which is used by numerous mathematicians all over the world to get help regarding mathematical problems.

The paper then talks about three models of Dialogue Systems: Frame based Model, Finite model and Agent based model. In the finite model, the user follows predetermined stages while in Frame based model, the system asks queries to the user if the question asked by the user is incomplete to generate a response.

The user asks a question. Then two cases can arise. First if the question is sufficient in its own and the system can generate a response based on the question. The second case is when the problem arises. The user asks an incomplete question like “What is the fare for so and so train?” In this case, the question is incomplete because the fare also depends on the boarding station and dropping station and also on the age of the passenger. In such situations, the system itself goes into a dialogue mode with the user and system asks queries to the user to fill the necessary gaps to generate a response. Like in this case, the system will ask: “Train Name/Code?”, “From which station?”, “Upto which station?”, “Name and Age of Passenger?” in format of a question and in Hindi. The user will then answer the questions and accordingly, finally, the system will generate the response based on those answers and format it and then print it response to the user.

The paper also states about the possible future work that can be done such as support of speech input and output using speech recognition systems.

Rule-Based Dialogue Management Systems

Nick Webb

Summary

The paper talks about various dialogue management strategies , a rule based approach and a particular chatbot - The YPA. The paper says that there is a need to develop Dialogue Management systems which can respond to complex user queries. Complex Dialogue Management components have been developed which allow users to have a more intelligent style of interaction with their information systems.

This paper presents a method by modelling dialogue actions as a series of rules. The paper focuses on simple interrogative and command dialogues based on slot-filler representations, and attempt to write sets of rules which describe the dialogue process necessary to ensure successful filling and manipulation of these slots.

Many Question-Answering systems give answers to correctly phrased natural language questions, but are unable to help the user when required information does not exist exactly in the format required, or the question itself is not understood. At this stage it becomes necessary to conduct a dialogue with the user, in order to further specify or clarify the information they require. So independent of the system, there are a number of dialogue motivators / reasons for continuation of the dialogue.

Some of these as described in the paper are:-

1. Queries generated by missing required information: Such as missing destination information in a flight booking system

2. Relaxation: dropping those constraints preventing system response

3. Augmentation: The opposite of relaxation, if the system needs more information to steer it to an accurate response

4. Confirmation: Paraphrasing user input for confirmation

5. Disambiguation of user inputs: When a user utterance could cover a number of system entries

6. Detection of user confusion/error correction: Re-routing the current dialogue to deal with user confusion

The criteria for good for a general dialogue system but may not be valid for specific applications.

The paper now talks about slot filling dialogues , where the goal of the interface is to fill some slots from natural language like input and pass these to the backend/ information source like yellow pages etc.

In other words, interrogative or command dialogues, where the dialogue can be seen as repeated attempts to operate over the domain. Each operation may take several steps, updating or revising slots to overcome incompatibility, for example. Once input is in a correct form, the slots are executed on the domain, which generally leads to termination and reporting of outcome. This may lead to a repeated attempt (in case of success) or revision and retrying (in the case of failure).

Ideally, all information retrieval systems should be able to meet a set of criteria, which could include recognising attribute being set from value, explaining mismatches of settings with domain and each other, offering ranked alternative actions in case of failure, having a degree of quality consciousness from a users perspective, avoiding pointless repetitions, guiding the user towards system goal, and spotting missing or contradictory parameters

The paper then talks about a model where the user and all internal modules are seen as alike by the DM system, and as such have to deal with arbitrary input and state in response to any of these modules. This suggests a model based on the product of machines or a production rule system. The specification of a product of machines representation could be too complex a description for our requirements, so the paper have chosen to use production rules to represent the

Conversational level of our dialogue, so long as the range of possibilities is fairly limited.

To implement such a model requires to have a single set of (task dependent) slots surrounded by some modules which produces events, where events are seen as event source, event type and some operands.

Current dialogue state can be calculated at any time from the dialogue history, which is a complete list of past events and states. Input from the user would be parsed, and dependent on the content of the dialogue history, added to existing slots or put into new ones. Operations over those slots would then depend on slot contents and previous operations.

The YPA

The YPA is a directory enquiry system which allows a user to access advertiser information in classified directories.

It converts semi-structured data in the Yellow Pages machine readable classified directories into a set of indices appropriate to the domain and task, and converts natural language queries into filled slot and filler structures appropriate for queries in the domain The generation of answers requires a domain dependent query construction step, connecting the indices and the slot and fillers. The Ypa illustrates an unusual but useful intermediate point between information retrieval and logical knowledge representation.