AI Model for 20 Questions

GROUP - 4

Bhavarth Pandya (201301119)
Tanay Agrawal (201301121)
Anubhav Jain (201301123)
Alay Shah (20130126)

Introduction :

20 Question game, traditionally, consists of an answerer and one or more guessers. The idea is that the answerer thinks of a famous personality or a character and the guessers question the answerer. The catch here is that the answerer can only answer in yes or no. The goal is to guess the correct answer within 20 questions.

In this variant of 20 question, the algorithm takes place of the guesser. The player can guess a name and the algorithm asks questions and tries to guess the correct answer within 20 questions.

Data and attributes :

In order to explain effectively, we have considered the dataset related to cinema, specifically, of the actors. The domain of the actors is Hollywood and Bollywood. A lot of attributes that a particular actor can have are easily available and to an extent quantifiable. Age, gender, nationality, domain of acting (Hollywood/Bollywood), religion, physical attributes (whether the person is tall/short, whether the person is fat/thin), relationship/marital status (both current and past) can be used using different combinations to provide a unique name.

In addition to these attributes, there can some secondary attributes if these are unable to provide a unique name from the dataset. If the list of all the actors and their movies is available, we can find out the co-actors of a particular actor. Similarly, a popular movie (having required number of cast members) can also be a good criteria to divide the dataset. More than that, if the family status of the said person can be found, the questions such as whether the children of the said person work in a particular industry (cinema, music, television et cetera) or whether the wife of the said person belong to a famous family. As mentioned, the questions pertaining different industry can also be relevant for there can be actors who work in both television and film. Just like the industries, we can also work on the cross

connections between Hollywood and Bollywood. At the end of it, the questions like 'whether the actor has a statue at the madame tussaud museum' can also be very effective to divide and narrow the dataset.

Actor/Attributes	Age	Gender	Nationality	Religion	Co - Actor(s)	Relationship Status...
Tom Hanks	60	Male	American	Christian	Zeta Zones	Married
Amitabh Bacchan	74	Male	Indian	Hindu	Shah Rukh Khan	Married
Russell Crowe	52	Male	New Zealand	Christian	Emma Watson	Married
Emma Watson	26	Male	British	Christian	Alan Rickman	Unmarried

Model

The problem that we need to address when we play 20Q is that what question to ask so that we can zero in on the answer in an efficient manner. A series of such questions leads us to the solution. To do, on the basis of the data, we define a set of questions based on each attribute. These questions facilitate the splitting of the data into subsets. The data given in the form of objects v/s attributes is converted to a form of objects v/s questions. The entries of the table 'yes' or 'no' tells whether the object satisfies the the question or not. We also create another table of objects and their weights. These weights signify the probability of the object being chosen as the correct answer by a player. The weights are obtained by learning through many games and updating the weights each time.
We use the following notations throughout the model explanations.
Objects : v_1, v_2, ...
Questions : q_1, q_2, …
Weights of objects : w_1, w_2, …

Pre-processing :

To make the model effective and more robust we perform some pre-processing so that we can draw more inferences about the objects. We consider a co-starrings network and we calculate the eigenvector centrality for each actor/actress. Now we consider a network of movies that has an edge between two movies given that they share a common actor. We now add the eigenvector centralities of those common actors and assign it to the movies calling it the 'cumulative stardom value'. The movies with the highest such cumulative centrality are considered to be the most viewed and the actors in such movies are considered to be quite famous. Thus we assign higher initial weights to the actors/actresses

of the movies with high cumulative stardom. This gives us a kick start in the learning process.

Decision tree model :

In this project we have used an ID3 based algorithm. The algorithm uses a decision tree to divide the set of Objects into subsets using their cumulative information gain values. The ID3 algorithm picks an attribute for the object and computes its entropy. For each attribute it calculates the information gain and selects the attribute with highest information gain. Using the selected attribute it divides the set into two new sets and then repeats the above steps for the divided sets obtained. We use a similar concept here for the given dataset. Here the set of movie actors are used as objects and their various characteristics are used to calculate the information gain for each question. We define the entropy and information gain for each question as follows.

Entropy : Entropy is defined as the measure of (im)purity of the data. In this context entropy shows the difference in the number of 'yes' and number of 'no' for each question asked on the data. Entropy for question is given by

$$E = - P_{yes}\log(P_{yes}) - P_{no}\log(P_{no})$$

Here $P_{yes/no}$ gives the proportion of yes/no for the question. The entropy is less when the proportion of yes and no are almost equal. Thus lower the entropy, better efficiency of the question in splitting the data set into almost equal parts.

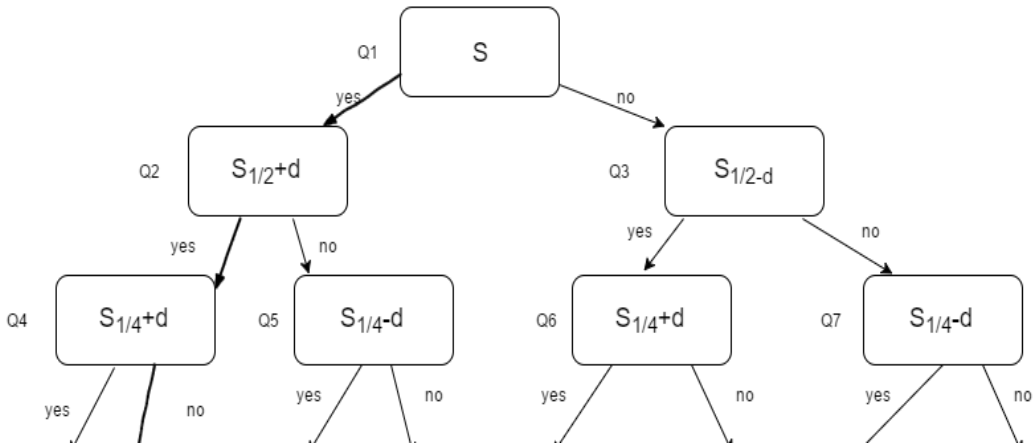
Information gain : Information gain gives us a measure of reduction in the value of entropy induced by the splitting of the current training set into subsets. This is recursive measure that says how effective a question is in splitting the data. The information gain of a question with respect to a subset S is given as,

$$IG = E(S) - \left(\frac{|Syes|}{|S|} E(S_{yes}) + \frac{|Sno|}{|S|} E(S_{no})\right)$$

Here $S_{yes/no}$ is the subset induced by a yes/no answer to the question. $E(S_{yes/no})$ is the entropy of the subset $S_{yes/no}$. When we choose a question based on this metric it forces the ID3 to create to shorter and more efficient decision trees. The second term in the RHS calculates a weighted sum of entropies of the subsets induced by answering the question and thus calculate the difference of entropies between the current set and its induced subsets.

We now use the information gain as one the metric in choosing the right question to ask. Initially as a part of pre-processing we calculate the information gain for all the questions considering the entire initial training set. We calculate the information gain at each iteration (at each node) thereafter for each question we calculate the IG of the question using the subset at that node. At each node the set is divided into two parts and the goal of ID3 based decision tree is to try and split the set into exactly half. This is because it the most efficient

way of narrowing in on the solution. Consider a counter case where we choose a question which divides the data into two subsets in ratio 1:4. In this case the probability of yes or no for a question is ½ but for one of them we will have a subset 4 times larger than the other to search. This would cost us cearly in terms of computations. Although we cover the possibility of selection of such high entropy questions under given conditions in the learning section of the model.



In the above figure we associate a question (the best one with each node). The d is added to the subsets to show the 'almost' perfect splitting of the sets.

<u>Learning</u> :

Let us consider playing the 20Q game as humans. Person A is the questioner who has to guess the character chosen by person B. Now person A asks a question 'Is the actor american? ' and B answers 'yes'. Now the first thing that A would think of is a pool of famous and highly probable american actors. A would assume (possibly with B's background information in mind) that B is more likely to select an actor/actress from this pool and will choose the next question such that it could help split this pool of candidate solutions efficiently. In the description of the model above we mentioned a table of objects and their weights. These weights are learned and evolve with every game played. They tell us what objects (celebrities) are more likely to be chosen by a player. This helps our algorithm in two ways, to effectively select a question that targets the most probable choices and divides them up and to break ties in case there are any between the choices.

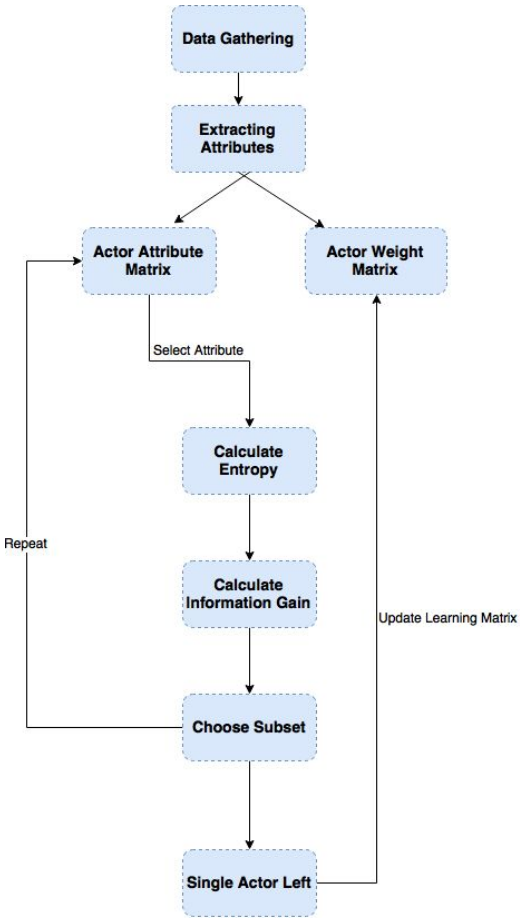
The weights are now used to filter out the the 10-20% most probable choices of the users at every node and the question having highest information gain with respect to that set is chosen. This effectively divides the pool of probable choices into half and immunes the method from outliers in terms of rarely probable choices. An important thing to note is that this condition makes the algorithm very strict and thus takes a long time to learn the proper weights of each object.

We also use the weights of the objects in pulling a trade off between low entropy questions and high entropy ones. As explained above, there can be a possibility that a question having high entropy (dividing the training set into unequal parts say 1:4) has a high probability of inducing a smaller than half sized subset. This could lead to more efficient convergence of

the algorithm. To include this factor, while choosing a question we select more than one candidates, one with the lowest entropy and others with entropies in a range say 0.75 - 0.8 . We calculate the mean weight of the objects in the subsets created due to the low entropy question and that of the objects in the smaller subset created by the high entropy function.

Implementation approach :

We now look at a simple flowchart that explains an implementation approach for the above algorithm.



The implementation starts with data gathering. IMDB provides API for the movie/actor database. Moreover we also obtain the data of the co-starring network of imdb. The next step is to extract the attributes from the data. This involves taking inferences from the available attributes. For example giving the actors initial weights in terms of how many popular films have they starred in and finding the most common co-star.

We next start the iterations of the decision tree. For each iteration first we decide the question to be asked buy calculating the entropy and thus the information gain. The choice of the question forces the algorithm to recursively calculate the entropies of the entire instance of the tree. Based in the answer to the chosen question we choose the subset i.e., we eliminate the part of the data that is not relevant.

When we reach the final solution we update the weight matrix.

Corner cases :

1. **Don't Know Option** - To make the Bot more varied and in accordance with the real life scenario an option of "Don't know" can be put for each question. This option will carry some weight as well to the given attribute and will help in decision making. Also keeping this option is beneficial as to avoid the incorrect answers provided by the user.

2. **Detecting Unauthentic Users -** At many times the users feed incorrect data to the bot. This can affect the decision making of the Bot as this will lead to misleading value addition to the learning matrix. Such users has to be detected and tackled. Such users can be of two type, those who answer always a same option or the users

feeds random answers without reading the question. For detecting same option users at a position a random question's negation will be asked to the user. So if answer "Yes" to both the answers, the Bot can detect the unauthentic user. To tackle the problem of random answers a same question can be asked twice in a slightly different way. So there is a very less probability that the user answer same answer for both the questions, Thus random answers will be discarded using this method.

References :

1. *Decision Tree Discovery*, Ross Quinlan, October 1999.
2. https://www.cs.princeton.edu/courses/archive/spring07/cos424/papers/mitchell-dectrees.pdf
3. https://en.wikipedia.org/wiki/ID3_algorithm
4. http://vlado.fmf.uni-lj.si/vlado/papers/IMDBvis.pdf