I Spy Decision Logic

Retrain all of the following after each round of play.

Train an object recognition classifier on all the image feature vectors for each individual object
Use one-versus-all classifiers (i.e., if there are 17 objects, there should be 17 classifiers, one for each object, where the positive examples are all the feature vectors from that object and the negative examples are a subset of the feature vectors from all other objects (for now, just use a couple of random examples from each of the other objects). Then you classify each object using each classifier and you (typically) believe/output the classification that has the highest probability.)

For each content word (CW) and for each object,
Count the number of that object's descriptions, thus far in the game, that include the CW.
If the number is below some threshold (for now, let's just say if it's in none of the object's descriptions),
Then include all of the object's image feature vectors as negative examples for that CW/tag.
If the number is at or above some threshold (for now, let's say >= 50% of the object's descriptions),
Then include all of the object's image feature vectors as positive examples for that CW.
If the number is above the lower threshold (> 0%) and below the upper threshold (< 50%),
Then do not include the object's image feature vectors either as a positive or a negative ex.
For each CW
Train a classifier on its associated positive and negative image feature vectors.

When playing the game, we want to ask the question that is most likely to lead to the correct guess after the fewest questions. This isn't quite the same as maximizing the Information Gain, from information theory, but is related. You could look at that, as one possible means of deciding what question to ask. I would consider the following. Estimate the probability of the person thinking of a particular object given a Yes answer to a question about a given tag,
P(*object* = O_i | Q_{tag} = Y). Let's write this as P(O | *Tag*). Then by Bayes rule:
P(*object* = O_i | $Q(tag_j)$ = Yes). Let's write this as P(O | Y_t). Then by Bayes rule:
P(O | *Tag*) = P(*Tag* | O) P(O) / P(*Tag*),
P(O | ¬*Tag*) = P(¬*Tag* | O) P(O) / P(¬*Tag*),
where P(*Tag* | O) is calculated below, P(O) is the probability assigned to the object after the previous question was answered (or 1/|**O**| = 1/17 before the first question), and P(*Tag*) is independent of O, and should be:
P(*Tag*) = Σ_O P(*Tag* | O) P(O).
On each round, calculate the P(O | *Tag*) for all O, and all unasked *Tag* questions. Choose the *Tag* that maximizes the *Expectation* of the difference between the average probability of **Θ** and **Ω**, where **Θ** is the top P(O|*Tag*) / 2^i and **Ω** is the next P(O|*Tag*) / 2^i and i is the round number (i.e., on each round, try to cut the search space in half). If there is a large margin between a subset of (**Θ** ∪ **Ω**) and everything else, you could instead focus on cutting that subset in half.

Compute a first $P_d(Tag \mid O)$ for each object based on the prior descriptions.
Compute a second $P_i(Tag \mid O)$ for each object based on running the CW/tag classifiers on each image.
Compute a third $P_q(Tag \mid O)$ for each object based on the users' actual responses to this question thus far.
Compute a fourth $P_{q,\mathbf{D}}(Tag \mid O)$ based on the responses to ***all*** questions, given a description's tag usage.
Compute a final $P(Tag \mid O)$ for each object based on the three preceding probabilities. This should be at least as high as the weighted average (maybe higher, let's see the performance): $P_W >= (w_1P_1 + w_2P_2 + w_3P_3)/(w_1+w_2+w_3)$, where w_i indicates a confidence.

Note for ACL Submission: For unknown objects, no prior descriptions or responses will exist, so we can only computer P_i. -Natalie

First, classify each image/object in the current round using the one-vs-all classifiers built above. (In our case, since there should be a one-to-one mapping between our 17 (potentially sets of) object image feature vectors and our 17 object classes, we should start by accepting the object classification that has the largest margin, the greatest difference between the two most likely classes. Then remove that object class as a possible value for the remaining image feature vectors and repeat the procedure.)

Note for ACL Submission: If, in this process, the object isn't classified above some threshold for any of the object classifiers, then assume it is a new object. -Natalie

For each potential CW/tag question that has not yet been asked
Estimate a first probability of an affirmative response with regard to each object-instance,
$P(Q_{tag} = \text{Yes} \mid Object = Object_i)$, based on the object's training descriptions thus far
$\cong P_d(Tag \mid O) \equiv P(Tag \in \text{Description} \mid Object = Object_i)$:
Note for ACL Submission: For unknown objects, just check against the feature classifier for this CW and let the first probability = 1 if the classification is positive or 0 if the classification is negative. -Natalie
For each of the tags used in descriptions thus far
For each of the 17 objects
Compute smoothed probabilities:
$P_d(Tag \mid O)$ = (count(descWTag)+1)/(n+2) // the probability is smoothed by adding 1 to the numerator and $|\mathbf{C}|$ to the denominator, where $\mathbf{C}$ is the set of possible classes, in this case $\mathbf{C}$ = {Present, Absent}, so $|\mathbf{C}|$=2, and n is the number of descriptions added thus far. On round 1, $P_d(Tag \mid O)$ is either 2/3 if the tag was used in the description, or 1/3 if it was not used. On round 3, $P_d(Tag \mid O)$ is 4/5, 3/5, 2/5, or 1/5, depending on whether it was used in all 3 prior descriptions, 2, 1, or none, respectively.

Similarly,
$P_q(Tag \mid O)$ = (count(yesForTagQuestion)+1)/(m+2) // the probability is

smoothed by adding 1 to the numerator and |**C**| to the denominator, where **C** is the set of possible classes, in this case **C** = {Yes, No}, so |**C**|=2, and *m* is the number of times the question has been asked thus far.

$P_{q,\mathbf{D}}(Tag \mid O)$ is computed using a combination of the following information:
$P_q(Tag \mid O)$ from above,
$P(\text{count(descWTag)}=k \mid n \text{ descriptions}) = (P_T)^k(1-P_T)^{n-k}\text{Combos}(n,k)$,
Average $_{Object,\ Description,\ Tag}$
$P(Q_{Tag}=\text{Yes} \mid \text{count}(Tag \in \text{Description}), \text{count(Description)}, Object=Object_i)$

Add the object's image feature vector(s) as positive examples to all of the Tag classifiers where the user responded Yes, and as negative examples where they responded No.

==========

Note for ACL Submission: Not related to this particular algorithm, but when unknown objects are used the robot will have to have some other way of guessing the item (since it won't know the object name). It could possibly just identify it by all of the CWs for which this object has received a positive response, e.g., "Is it the round, orange object?" Or, if objects were always going to be situated on the gamespace in easily-identifiable locations (to the human and to the system), it could identify it by numerical ID, e.g., "Is it object 6?" The latter technique is less natural, but also less likely to confuse human players (if the only thing differentiating two items was a negative response, such as "no" to Object = tangerine, Question = "Can it bounce?" then "Is it the round, orange object?" could describe both an unknown basketball and the tangerine). -Natalie

==========

Later, need to consider this from an earlier project description:

If the question correctly guessed the final object, then training instances would be added to the corresponding object training set(s), and the images would also be added to several feature datasets. For example, training instances would be added where the robot did not ask a question due to a high-confidence belief about the value of a particular feature, which was consistent with the final object classification. As with the feature training datasets, there would be multiple object training sets; binary datasets such as cup vs. not_cup and a multi-label dataset including, for example, cup, dog, book, person, etc. Additionally, an object training dataset will be created based strictly on the features identified, rather than the images. The feature vectors in such a dataset would look similar to this:
color=white, handle=yes, container=yes, holds_drink=yes, handle=yes, … , label=cup

If the robot guesses the object correctly, it wins. Items that it never does successfully recognize within its 20-question limit are eventually added to its datasets when the human tells the agent the correct name of the object.

After the game, especially in the case where the robot does not correctly determine what the object is, the robot would ask the person to describe the object, and would then update its speech and language understanding datasets. As part of this process, the robot will look up various descriptions of the object and use those descriptions to create language models and vocabularies in order to understand the human's description.

Similarly, the robot would ask the human what questions would have been good questions to ask to determine what the object was, or what are the most unique aspects of the object. The responses to these questions should be interpretable based on the descriptions found online and based on the person's description of the object. The questions will help the robot understand the importance of various features and help it refine the way it asks questions. Similar processes might be followed to improve other aspects of its Natural Language Understanding.