Problem 1: 15 points

Suppose a program for natural language processing is given the following text:

Sam handed a plate with a sandwich to George. He ate it quickly.

A. What is the result of morphological analysis applied to this text?

**Answer:** “handed” is identified as the past tense of “hand”; “ate” is identified as the past tense of “eat” (by dictionary lookup); and “quickly” is identified as the adverbial form of “quick”.

B. Show the correct parse tree for the first sentence. (The exact details of the grammar are up to you, but do something plausible.) Do not give a trace of any parsing algorithm.

**Answer:**

```
S ---> NP ---> Name ---> Sam
    |    
    |--> VP ---> Verb ---> handed
        |      
        |--> NP ---> Det ---> a
        |         |      
        |--> Noun ---> plate
        |                   |      
        |--> PP ---> Prep ---> with
        |                        |      
        |--> NP ---> Det ---> a
        |                          |      
        |                          |--> Noun ---> sandwich
        |                        
        |--> PP ---> Prep ---> to
                        |    
                        |--> NP ---> Name ---> George
```

C. There are two anaphoric references in the second sentence, both of which are ambiguous. Explain how both ambiguities can be resolved.

**Answer:** “He” is ambiguous as between Sam and George. Resolving it to George requires the world knowledge that (a) After X has handed O to Y, Y has possession of O and X does not; and (b) in order for Z to eat O, it is necessary for Z to have possession of O.

The word “it” is ambiguous as between “plate” and “sandwich”. This can be resolved using selectonal restrictions: The object of “ate” must be edible, sandwiches are marked as edible whereas plates are not.

A number of students also pointed out the undeniable fact that the recency heuristics carries out both of these resolutions correctly.
Problem 2: 20 points

Let \( U \) be a domain consisting of people and countries. Let \( \mathcal{L} \) be the first-order language with the following predicates:

- \( s(X,Y) \) — Person \( X \) spies on person \( Y \).
- \( c(X,Q) \) — Person \( X \) is a citizen of country \( Q \).
- \( b,f,g \) — Belgravia, Freedonia, Gothia.

Assume that "X is a spy" can be expressed "There is someone whom X spies on."

Express the following sentences in \( \mathcal{L} \).

1. Some Belgravian is a spy.
   \[ \text{Answer: } \exists X,Y \ c(X,b) \land s(X,Y). \]

2. All Goths are spies.
   \[ \text{Answer: } \forall X \ c(X,g) \Rightarrow \exists Y \ s(X,Y). \]

3. Every Goth has some Freedonian they’re spying on.
   \[ \text{Answer: } \forall X \ c(X,g) \Rightarrow \exists Y \ s(X,Y) \land c(Y,f). \]

4. Every Belgravian spy is being spied on by some Freedonian. \[ \text{Answer: } \forall X \ [c(X,b) \land \exists Z \ s(X,Z)] \Rightarrow \exists Y \ s(Y,X) \land c(Y,f). \]

Problem 3: 15 points

Note: in this problem and in problem 5, the numbers have been chosen so that the arithmetic is very simple. If you make an arithmetic mistake, there will be a deduction of 0.5 points out of 10. If you write down the correct arithmetic expression and do not attempt to evaluate it as an exact fraction, there will be a deduction of 1 point out of 10. You do not have to reduce fractions to lowest terms.

Consider the Bayesian network shown below.

```
   A
  ^   \
 /    \
B     C
```

Assume that all the random variables are Boolean, and that the following probabilities are recorded:

- \( \text{Prob}(A=T) = \frac{3}{4} \)
- \( \text{Prob}(A=F) = \frac{1}{4} \)
- \( \text{Prob}(B=T) = \frac{1}{2} \)
- \( \text{Prob}(B=F) = \frac{1}{2} \)
- \( \text{Prob}(C=T|A=T,B=T) = 1 \)
- \( \text{Prob}(C=F|A=T,B=T) = 0 \)
\[
\begin{align*}
Prob(C=T|A=T,B=F) &= 1/2 \\
Prob(C=F|A=T,B=F) &= 1/2 \\
Prob(C=T|A=F,B=T) &= 1/2 \\
Prob(C=F|A=F,B=T) &= 1/2 \\
Prob(C=T|A=F,B=F) &= 0 \\
Prob(C=F|A=F,B=F) &= 1
\end{align*}
\]

Compute
A. Prob(C=T)
Answer:
\[
\text{Prob}(C=T) = \text{Prob}(C=T,A=T,B=T) + \text{Prob}(C=T,A=T,B=F) + \text{Prob}(C=T,A=F,B=T) + \text{Prob}(C=T,A=F,B=F)
\]
\[
= \text{Prob}(C=T|A=T,B=T) \text{Prob}(A=T,B=T) + \text{Prob}(C=T|A=T,B=F) \text{Prob}(A=T,B=F) + \text{Prob}(C=T|A=F,B=T) \text{Prob}(A=F,B=T) + \text{Prob}(C=T|A=F,B=F) \text{Prob}(A=F,B=F)
\]
\[
= 1 \cdot (3/4) \cdot (1/2) + (1/2) \cdot (3/4) \cdot (1/2) + (1/2) \cdot (1/4) \cdot (1/2) + 0 = 10/16 = 5/8.
\]
B. Prob(C=T|A=T)
Answer:
\[
\text{Prob}(C=T|A=T) = \text{Prob}(C=T,A=T)/\text{Prob}(A=T).
\]
Now \[
\text{Prob}(C=T,A=T) = \text{Prob}(C=T,A=T,B=T) + \text{Prob}(C=T,A=T,B=F) = \text{as above}
\]
\[
1 \cdot (3/4) \cdot (1/2) + (1/2) \cdot (3/4) \cdot (1/2) = 9/16.
\]
Therefore \[
\]
C. Prob(A=T|C=T)
Answer:
\[
\]

**Problem 4: 10 points**

Fill in the squares of this table True or False. (You may do this on the exam sheet.)

<table>
<thead>
<tr>
<th>K-nearest neighbors</th>
<th>Perceptron</th>
<th>Feed-forward back-propagation network</th>
<th>Naive Bayes</th>
<th>ID3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

(The ID3 algorithm actually can make use of both training and test items with null values, but the method for doing so is complicated and I did not describe it in class.)

K-nearest neighbors is actually incompatible with minimum description length learning, since the classifier is just the entire table of training data. However, since I specifically (incorrectly) stated in class that all classification algorithms and indeed all learning algorithms are compatible with MDL, I did not deduct for answering “T” there.)
Problem 5: 15 points

Consider the following data set. Q, R, S are the predictive attributes; C is the classification attribute.

<table>
<thead>
<tr>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>C</th>
<th>No. of instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>8</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>15</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>2</td>
</tr>
</tbody>
</table>

A. What is the best rule computed by the 1R algorithm for predicting C based only on Q? What is the accuracy of that rule over the data set?

**Answer:** case(X.Q) T : predict X.C==T; F : predict X.C==F

Accuracy: 23/30.

B. What is the Naive Bayes prediction for C for an instance where Q=T, R=F, S=F?

**Answer:**

Ignoring the normalization factor, we estimate Prob(C=T—Q=T,R=F,S=F) as

\[ P(Q=T—C=T) \cdot P(R=F—C=T) \cdot P(S=F—C=T) \cdot P(C=T) = \frac{8}{10} \cdot \frac{2}{10} \cdot \frac{2}{10} \cdot \frac{10}{30} = \frac{320}{30,000} = \text{about 1/100.} \]

we estimate Prob(C=F—Q=T,R=F,S=F) as

\[ P(Q=T—C=F) \cdot P(R=F—C=F) \cdot P(S=F—C=F) \cdot P(C=F) = \frac{5}{20} \cdot \frac{2}{20} \cdot \frac{3}{20} \cdot \frac{20}{30} = \frac{600}{24,000} = 1/400. \]

The prediction is therefore C=T.

C. In using a Naive Bayes classifier on text or on data sets with many attributes, it is common to use the logarithms of the probabilities involved rather than the probabilities themselves. Why?

**Answer:**

Multiplying together many probabilities less than 1 quickly leads to floating point underflow.

Problem 6: 10 points

One problem with the k-nearest neighbors algorithm is the problem of the comparative scales of different dimensions. For instance, suppose that we have a table of people recording their height in feet, their weight in pounds, and whether they are overweight or not.

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Overweight?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>150</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>180</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>280</td>
</tr>
</tbody>
</table>

Now, if we consider a new person who is 5.9 feet and 160 pounds and we use the Euclidean distance, then the “distance” to person #2 is \( \sqrt{(5.9 - 5.0)^2 + (160 - 150)^2} = 10.04 \) whereas the “distance” to person #3 is \( \sqrt{(5.9 - 6)^2 + (160 - 180)^2} = 20.0002 \). That is, the height is essentially completely ignored. Intuitively, though, the new person should certainly resemble person 3 more than person 2.

Propose a method for fixing this kind of problem (in general, not just in the specific case of feet vs. weight). You do not have to recalculate the results for the above example.
Answer: Rescale the values based on their range. That is, rather than measure height in feet and weight in pounds, define a new scale where each attribute ranges a fixed numeric range (0 to 1, or -1 to 1, or something of the kind). This can be done in a number of ways, with various pros and cons:

A. Compute the range as the maximum value of the attribute minus the minimum value, and divide all the values through by this range.

B. Compute the standard deviation, and divide through by the standard deviation.

C. Use percentiles, rather than absolute values. (Note that this may be a non-linear scale.)

For instance applying method A to weights in the above table, since the range is 180 lbs, one would measure weights in units of 180 lbs and derive the new table:

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight 180/180</th>
<th>Overweight?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>5</td>
<td>100/180 = 0.56</td>
</tr>
<tr>
<td>2.</td>
<td>5</td>
<td>150/180 = 0.83</td>
</tr>
<tr>
<td>3.</td>
<td>6</td>
<td>180/180 = 1</td>
</tr>
<tr>
<td>4.</td>
<td>6</td>
<td>280/180 = 1.56</td>
</tr>
</tbody>
</table>

The test example would then have height=5.1, weight=160/180 = .88. Thus it is obviously closer to (2) than to (3).

Problem 7: 10 points

A. Describe the use of a training set and a test set in supervised learning.

Answer: Given a corpus of labelled examples, one divided it into a training set and a test set. The training set is used as the input to the supervised learner. The classifier (or other task executor) is then held fixed and run on the examples in the test set, to measure the quality of its output by comparing the computed answers to the ones recorded in the training set.

B. Why is it important to keep these separate?

Answer: So that it is possible to distinguish between an executor which is genuinely able to carry out the task on unseen examples, and an executor that has been overfit to the data, so it only works on examples in the training set.

Problem 8: 5 points + extra credit

The k-gram model of tagging makes two independence assumptions:

1. The ith element, $E_i$ depends only on the ith tag $T_i$ and is conditionally independent of all the other elements and the other tags given $T_i$.

2. The ith tag $T_i$ depends only on the previous $k – 1$ tags and is conditionally independent of all the earlier tags given $T_{i+1-k} \ldots T_{i-1}$.

Consider the case where the elements are words and the tags are parts of speech.

A. (5 points) In class, I stated that assumption (1) is completely and obviously untrue. Explain why, using an example.
**Answer:** The assumption that one word in the sentence is independent of all the other words in the sentence, given the tags, is obviously false. For instance, if there are two nouns in the sentence, then, given that the first is "vase", it is much more likely that the second is "flowers" than that it is "pancakes".

B. (5 points extra credit). Assumption (2) is also untrue, but this is considerably more subtle. Give an argument to show that assumption (2) is untrue.

**B. Answer:** Suppose that we are making the $k$-gram assumption and there is a $k + 1$ word sentence, each of which can be nouns. Consider the tagging in which all of them are noun. Since each subsequence of $k$ tags is possible (there can be compound noun groups of arbitrary length at either the beginning or the end of the sentence) the $k$-gram model will assign this tagging a non-zero probability. However, the actual probability of the tagging

\[
\text{<StartSentence>} \text{<Noun>} ... \text{<Noun>} \text{<EndSentence>}
\]

is 0 because of the constraint that every sentence has a verb.

**Hint:** Assume, which is true, that the NLP system marks the beginning and end of sentences by adding a special "StartSentence" and "EndSentence" flag both as elements and as tags. So the sentence “John wears pyjamas” is treated as though it were “StartSentence John wears pyjamas EndSentence” with tags “<StartSentence> <Name> <Verb> <Noun> <EndSentence>” Now think about the constraint that every sentence must contain a verb, and argue that for some sentences this constraint implies that the independence assumption must be false.