Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 / MoSIG M1 Final Exam - May 2016

Conditions: You have the right to use any notes or written material. You may answer questions in English or in French. When appropriate, illustrate your answer with mathematics. Your written answers must be clear and legible. Illegible text will not be graded. Duration: 3 hours.

1) (4 points) Define knowledge. Identify several forms of knowledge and explain how each form can be represented. For each form of knowledge, describe tasks for which the form may be used, explain how the knowledge can be acquired, and discuss limitations.

2) (2 points) Provide a definition and an explanation for an ROC curve? How is it calculated? How is it be used?

3) (6 points) Temporal interval logic provides a tool to reason about the relations between intervals. The relations used in interval logic can be seen as predicates defined over intervals of time. In this exercise we will use upper case letters (A, B, C...) to stand for intervals and lower case letters to represent relations (m, mi, d, ...). Because the symbols ">", "<", and "=" correspond to CLIPS keywords these will be replaced with "b" (before), "bi" (before-inverse) and "e" (equals). For example, if A can be before or meets or overlaps B then in interval logic, this is written

A --(b, m, o)--> B.

As predicates this would be written $b(A, B) \vee m(A, B) \vee o(A, B)$.

A predicate is a function that returns a likelihood value. We can note this as a set of couples (r, p) where r is a temporal relation and the p is the corresponding likelihood.

To apply transitivity, you should compute the product of the probabilities for each possible pair of relations. Note that with transitivity, when R_{AB} and R_{BC} have multiple relations, the same relation may become possible in multiple ways in R_{AC}. This will increase its likelihood. This can be modeled by summing the probabilities for each occurrence of a relation in the set produced by transitivity. To use these new likelihoods as probabilities, the likelihoods for the set of possible relations must then be renormalized to sum to 1.

a) In the absence of any knowledge of relations between intervals A and B, all relations are possible and equally likely. Write out the set of possible relations R_{AC} with the associated probabilities.

b) Suppose we have "A overlaps B" and "B overlaps or meets C". Write our the set of relations and probabilities for R_{AB} and R_{BC}.

c) (2 points) What are the probabilities for the possible relations by Transitivity between A and C?

d) (2 points) Suppose you have multiple paths for the relations between two intervals (e, g, R_{AC}^{B} and R_{AC}^{D}). How would you combine the probabilities of the relations in the two lists to obtain a unique list.

4) (6 points) The goal of this question is to write CLIPS rules that will use the frequency of words and word pairs to predict the next word as a user types each word of a text.

Your system uses following templates for Words and Word-Pairs.

```
(deftemplate Word ; structure for counting words (1-Grams for words)
      (slot WORD1 (type STRING))
      (slot COUNT (type INTEGER) (default 0))  ; Number of instances of word
)

(deftemplate WordPair   ; structure for counting Word Pairs (2-Grams of words)
      (slot WORD1 (type STRING))
      (slot WORD2 (type STRING))
      (slot COUNT (type INTEGER) (default 0))  ; Number of instances of word
pair
)
```

As each word is typed by the user, your system will create a Fact of the form

```
      (NewWord ?w)
```

where ?w is the most recent word. Your system will read words using a special function (read-word) that returns the most recent word as a string each time the user types a space. If the user types a carriage return, read-word returns as nil.

a) If there is no Fact of type NewWord, then the system should prompt the user to type a word. Write a rule named "ReadWord" that prints a question mark and then uses the function read-word to read the next word and create a fact of type NewWord.

b) For each new word, the system will generate a set of hypotheses, represented by a Fact of the form (NextWord ?w ?p), where ?w is a possible next word and ?p is the probability that this will be the next word based on the frequency of Words and WordPairs. Write a rule named NextWordHypothesis to create possible hypotheses and their probabilities.

c) Once all of the hypotheses for the next word have been generated, the system should print out the most likely word. Write a rule named PredictWord that prints out the most likely next word. Make sure that this rule has lower priority than NextWordHypothesis so that it only executes after all hypotheses have been generated.