Homework 4: The Mini-Project

15-381 - Artificial Intelligence - Spring 2000

Handed out on March 23, 2000
Proposal due at the beginning of class April 6, 2000
Final write up due at the beginning of class on April 27, 2000

1 Overview

1.1 Introduction

This last homework is a mini-project exploring a topic of your choosing. The mini-project is to be a more substantive bit of work than a standard homework, so it is a good idea to get started early. A mini-project is an opportunity to further explore one of the topics covered which you found interesting. Mini-projects typically involve coding up some algorithm covered in class and evaluating its performance on some example problem or data set. You can either select a project from the list which follows, or propose your own.

Mini-projects can be done individually or in teams of 2 or 3 people – your choice. However, we highly recommend working in teams. All team members will receive the same grade for the project; it is up to you to divide up the work. Also, recall that if you have a cumulative numerical grade corresponding to an A- or better, including the mini-project, you will receive an A for the course an do not need to take the final exam. Each project team (or individual) will be assigned a TA as mentor, with whom you will interact and to whom you will turn in your project materials.

1.2 Project Write Up

At the end of your mini-project you will need to turn in a 5-to-7 page written report in to your project mentor TA which includes the following:

- a description of the algorithms / methods implemented
- a description and analysis of the results obtained
- a description of the lessons learned / conclusions
- printouts of code and sample runs

If you are working in a group, only one report needs to be handed in for the entire group. Please confer with your TA as to how much and in what form each of the above should take in your final report, as there is necessarily some variation across projects.

1.3 Project Proposal

Before starting your project you will need to hand in a one page project proposal, which should include the following:

- a list of the members of your team
- a description of the topic you plan to explore
- a description of what you plan to do
- a description of the scope what you plan to do
- a general description of how you will go about doing it

Based on your project proposal, you will be assigned a TA as a mentor who has some expertise in that area. Your mentor TA will review your proposal and provide you with feedback.

1.4 Questions

If you have any questions about the mini-project itself or the candidate project topics, please contact Chuck Rosenberg, `chuck@cs.cmu.edu` or talk to one of the TA's during office hours. Once you have been assigned a TA who will be your mentor for your mini-project, you should contact that TA directly with any questions.

2 Candidate Project Topics

The following is a list of sample project topics. You may choose from one of the topics in this list, come up with a variant, or come up with a completely new idea of your own.

2.1 Comparative analysis of two or three text classifiers

Go to a web site such as Yahoo that has web pages categorized into a taxonomy. Select 10 to 20 categories in the taxonomy, and for each category ensure that you have at least 30 test web pages with text on them (e.g. NOT mostly GIF or Javascript), Divide the collection randomly into a training set, a validation set, and a test set. Then implement two or three text categorizers (e.g. each group member implements one), train them on the training set (with category labels) and test them on the validation and test sets (without labels) to see how well they predict the category label. Use accuracy, precision, recall, and/or F1 (from the class notes) as your measures. Which classifier performs best? Why might this be so? Try different random splits of the data and analyze which is more sensitive to the selection of training/testing data? [You may use Naive Bayes, k-Nearest-Neighbors, Decision Trees or any other such classifiers.]

2.2 Parsing Natural Language queries into SQL database queries

You may wonder how certain web sites can process natural language queries. Int his project you will use either ATN or case-frame instantiation methods to parse NL queries (questions and commands such as those below) into well-formed SQL queries that can be issued to a DBMS (such as Oracle, Access, Sybase, etc.). Use database semantics to disambiguate if needed for syntactically-ambiguous sentences. Build yourself a dictionary and grammar (either ATN or Case frames) that is separate from the ATN interpreter code or the case-frame interpreter code. The interpreters should be able to work with a different grammar for a different DB or a different task. It is usually best to interpret the language and build internal data structures (e.g. case frames) in step one, and map these into SQL as a separate step 2. Make sure that you handle at least one difficult NL structure (such as multiple prepositional phrases, conjoined phrases and/or relative clauses). Which NL phenomenon is hardest to handle and why? Example NL inputs for an employee DB for company web.com, with offices in Atlanta and Pittsburgh follow. Assume the DB is a single relational table (e.g. no "joins" or other difficult DB operations). List the employees in Pittsburgh. Which people work in Pittsburgh? Which employees make over $60,000? Give me the employees who make more than $60,000. How many employees in Pittsburgh make more than $60,000? List the web.com people in Atlanta 3. Build a clause-form resolution interpreter. I.e. given a knowledge base of facts in skolemized clause form, and a new fact to be resolved against that (small) knowledge base, write an interpreter that outputs a sequence of unifications and resolutions to prove it true by resolving its negation with the knowledge base. If it is not resolvable, the interpreter should output that fact, and if the interpreter runs over an effort bound (e.g. max number of attempted unifications, or max time), it should output a time-out message. You should develop some heuristics of how to manage the best-first search – i.e. what to unify if more than one possibility is present – for instance, the search branch that leaves the least to be resolved after one-step lookahead is often the best one to pursue next. If you are a group, especially a 3-person group, you may also want to investigate converting FOL into clause form automatically as a pre-processing step. The interpreter and the knowledge base should be separate (i.e. the same interpreter should work on a different knowledge base without requiring surgery). The new clause to be resolved should be an input.

2.3 Game Playing: build an alpha-beta minimax search engine and apply it to playing othello or checkers

You may be curious about the details of the internals of game playing programs like Deep Blue. (A descendant of CMU's own Deep Thought.) In this project you will build an alpha-beta minimax search engine and apply it to playing othello or checkers. For your implementation, you will need a proper evaluation function for othello (e.g. the total number of white / black pieces, pieces at particular key locations in the board) or for checkers (e.g. number of men on each side, mobility, center control, queening, etc.). Selecting these features and the evaluation function requires the consultation of someone who knows how to play othello or checkers really well. This could be a member of your group or an external "expert". You should also build a (simple) end-game library to know how to press home a win (e.g. 2 or 3 against 1). The product should be an othello or checker playing program with just enough UI that someone – such as the TA – can use it reliably. Note that this project may seem easy, but will take more time than you may at first estimate, even given this warning. In lieu of othello or checkers, you may also want to consider getting the computer to play other games which you find interesting. If you decide to try some other game, consult a TA before your proposal.

2.4 Scheduling / Constraint Propagation

Remember those candy bars ending up in the trash? Here's your chance to save more chocolate for humanity. Given a set of scheduling constraints and an evaluation function for a schedule, implement a program that performs hill climbing and/or simulated annealing to come up with a good schedule. If the search space is small enough to do an exhaustive search to determine the optimal schedule, compare the schedules generated by hill climbing/simulated annealing to the optimal schedule using the evaluation function. Examples include:

1. scheduling exams to blocks in the exam period given room and time constraints
2. assigning office workers to cubicles based on preferences such as proximity to clueless boss, proximity to annoying co-workers, whether cubicle is a window one or not, proximity to coffee machine, proximity to copier, etc,
3. scheduling production flow in a factory.

2.5 Resolution Interpreter

Write code that, given a database of first order logic statements in clause form and a statement, can determine whether that statement is consistent or not with the database.

2.6 Natural Language

Given a set of data consisting of children's misreadings of words (collected by the University of Colorado over a multiple-year twin study), learn rules that predict their misreadings based on spelling and/or pronunciation. The data looks like:

- KITTY -> CAT
- KITTY -> /k uh t iy/
- WAS -> /s aw/ ...

(The actual data is in a slightly different format.)

2.7 Machine Vision and Machine Learning

Apply at least two machine learning techniques to a machine vision task like face detection, car detection, or head pose detection. Analyze and compare the performance / features utilized by the machine learning algorithms. Some possible choices of machine learning algorithms which could be applied here are neural networks and Naive Bayes.

2.8 Bayes Networks

In the machine learning assignment you experimented with very simple Bayesian Network structures. Some other structures are only a bit more complex but are very useful. Write code to construct simple tree structured Bayes Networks from data and use them for prediction on a data set selected with your mentor. Compare the performance of the tree structured network to Naive Bayes and Joint Bayes on either the full feature set or a subset of the feature set. One possible data set is a text classification data set. A TA can give you a pointer to a specific paper which can be used as a reference for this project.

2.9 Frame / Semantic-net interpreter

Implement a frame/semantic-net interpreter with pushers and pullers as well as simple inheritance.

2.10 Choose your own topic – *Be Creative!*

Define your own project. Select some cool AI-related topic that you found really interesting or exciting during the semester and you would like to learn more about. Think about what you want to do (it needs to be a mini-project sized chunk of work) and describe the topic and the work you would like to do in your one page proposal. You should discuss your idea with one of the TA's before writing it up.