Crosswords have always been a favorite pastime among Americans, and crossword puzzles of all shapes, sizes, difficulties, and topics are printed daily in newspapers and magazines across the country. Crossword puzzles are unique in that they are more than just logic puzzles – crosswords require a broad knowledge base, a firm grasp of the English language, and a good guess here or there to complete a puzzle. There is virtually no limit to the number of crossword puzzles that can be created, and each puzzle brings a new challenge to even an expert solver.

However, due to the complexity and vast array of knowledge required for solving crossword puzzles, a process for solving a crossword puzzle programmatically is difficult to develop. The only highly successful attempt at doing so came from Michael Littman and his colleagues at Duke University in 1999. Their final product, PROVERB (the "probabilistic cruciverbalist"), averaged 95.3% words correct and 98.1% letters correct on a sample of 370 puzzles taken from the New York Times and other puzzle sources. Their solution used a set of 30 different modules, pulling potential answers from various word lists, crossword clue databases, encyclopedias, and topic-specific database modules (music, literature, geography, etc.).

The purpose of the various modules was to span as many categories of knowledge and trivia as possible, in order to match the shallow and wide array of facts necessary for solving a crossword puzzle. In the years following PROVERB, however, many new resources for scouring the vast array of human knowledge have been developed and placed at our fingertips – e.g., the Google search engine. By harnessing the search capabilities of Google and other Internet search engines and databases, it seems quite possible to develop a crossword-solving application that relies primarily on Internet resources to determine answers to clues. Not only would this breed of solver be able to reference a wider array of trivia, but the solver's database would always be up-to-date and able to provide answers to more recent references in clues.

The nature of the Google search will be somewhat probabilistic. After sending the clue as a query to the search engine, the words that appear in the top results can be narrowed down to those that fit the puzzle, and those that appear most frequently. For example, the correct answer to the clue "Deutschland über _____" (Alles) would be expected to appear most frequently in the search results using a query containing the clue's text.

One difficulty that natural language poses is making the distinction between different word forms (i.e. singular vs. plural) or other aspects of the English language that crossword puzzles tend to utilize (for example, abbreviations). For this reason, the Google module will be supplied with a set of rules for manipulating clues based on certain keywords in the clue. For example, clues with the term "abbr." will have the word "abbreviation" appended to the query, and those with the words "for instance" will have the word "synonym" added to the search.

Although Google alone is a very powerful resource, there are other Internet resources that can reinforce and enhance the creation of word lists. OneLook, an Internet dictionary search engine, provides a few features that are suited for the stumped crossword solver. OneLook can handle single-character wildcard searches (i.e. "b??q??t" will return the word "banquet", among others) and can also search the full text of a dictionary definition and return words that fit the
description provided by the user. Consider an eight-letter clue beginning with E and ending in N. A simple wildcard search on that pattern (“e??????n”) would return a large list of words, but supplying one keyword can narrow the list down to one entry: "Albert" will narrow the word list down to the German physicist "Einstein", while "Alberta" will return "Edmonton", the capital city of the Canadian province of Alberta.

The last of the resources (and probably the most relevant of the three) is OneAcross.com, a crossword solving utility that uses an extensive clue database and advanced crossword logic to produce a word list sorted by relevance to the puzzle. One capability that OneAcross has that OneLook doesn’t is the ability to return multiple-word clues and other non-standard English responses that OneLook may miss. As with OneLook, OneAcross can perform wildcard searches, and even returns the source for the response that it gives. A response of “known clue” can help cement a few clues at the beginning of the solving process, providing for a firmer foundation as the solver ventures into other directions.

The primary goal of this project is to develop a crossword solver that utilizes Google, OneLook, OneAcross, and other Internet search engines and databases to work towards a solution for a given crossword puzzle. Each of the search modules will independently produce a candidate word list, and the most probable answer to the clue will be chosen from the candidates based on various criteria specific to each module. With Google, for example, the confidence value for a particular candidate word will be based on the relative frequency of that word in the search results compared to other words that could fit in the puzzle. When searching OneLook with the full-text definition search, the most relevant results will appear earlier in the word list.

Although Google and OneLook will perform the majority of the searches, other modules will be implemented as necessary, and as time permits. Possible secondary sources include Wikipedia and IMDb (the Internet Movie Database).

The solver will be supported by a crossword database similar to the one used by Littman et al. in PROVERB. The solver's database will contain clue/answer pairs from over 6500 puzzles (compared to approximately 5000 puzzles supporting PROVERB), with more clues added to the database every day. The crossword database will serve two purposes. First, it will supplement OneLook as a means of building word lists of potential matches for the puzzle, independent of the clue text itself. Second, the database will (most likely) have clues that appear in the puzzle to be solved, and those solutions can be pulled directly from the database into the solution set. The crossword database will also prove to be useful when the response is an abbreviation or some other uncommon English construction that has appeared in previous crossword puzzles. Such answers probably can’t be found in OneLook or a sufficient number of results in Google to be considered candidates for the puzzle. Littman reported that approximately 30% of sample puzzle clues were found in some form in his crossword database, a number significant enough to greatly affect the performance of the new solver. I also hope to obtain a crossword clue database from Stan Newman, the crossword editor at Newsday, with whom I have already worked closely to produce Yale’s first crossword puzzle tournament.
Since the solver is largely probabilistic, the solver will employ a backtracking mechanism to evaluate the various outcomes that result from taking different paths in the solving process (affected by both the order of the clues to be solved and the words chosen for each clue). A major challenge is determining how good is good enough and how far the solver needs to go before realizing that there is an error in the solution.

Another complication to consider is the order in which clues are evaluated. Since the clues will need to be parsed and manipulated to create better queries anyway, part of the parsing process will assign a priority to each of the clues in the puzzle. Since fill-in-the-blank clues have the greatest chance of being found in Google (by using a quoted phrase search and providing a wildcard for the missing word), they will be solved first by the solver. Other qualities of each of the clues will be evaluated to determine how easy it will be to solve a clue with little additional information (such as intersecting words). Once more words are filled in, a clue can be solved using only the crossword solution database and a filtered word list from OneLook.

A solver that even comes close to PROVERB’s success rate is unrealistic for this project’s time frame, but I hope to develop a solver that performs reasonably well given a relatively straightforward crossword puzzle (i.e. with few answers involving puns or esoteric trivia that wouldn’t appear dominantly in a Google search). A lot of the project’s plans are open to change, depending on how successful the first runs prove to be. Once the first runs are completed, I plan on making incremental improvements to the code, hopefully improving the success rate with each improvement over time.

I plan on writing all of the source code using C# and .NET 2.0. First, I find it very easy to integrate the various modules and functionalities with one framework, and Microsoft’s IDE, Visual Studio 2005, is a perfect environment for developing such a multi-component application. My second reason is on a more personal level, since the majority of the programming that I will be doing for my employer next fall involves C# and .NET, and using the .NET framework for this project will help me become accustomed to the technology. The .NET framework has a powerful regular expressions library that I will be using to parse the HTML from the results pages, and GUI design in Visual Studio is incredibly simple.

Rather than developing a particular format for storing and importing crossword puzzles, I decided to leave all of my source puzzles in Across Lite format, the de facto standard for crossword puzzle distribution over the Internet. The format is easy to parse and relatively compact, and using this format reduces the processing overhead necessary for converting 6500+ puzzles once they are downloaded. A subset of the puzzles are distributed in PDF format, but preliminary experimentation with a PDF parsing utility leads me to believe that these puzzles could also be used with the solver. I plan on writing a utility for converting these puzzles over to Across Lite, so the final product only has to deal with one format.

The GUI that I plan on developing will illustrate and drive the entire solving process: the GUI will display each of the intermediate solving states and the word lists evaluated for each clue, and it will provide the ability to manually select which clues to evaluate at a given time (in order to experiment with clue priorities).
My deliverables at the end of the project will consist of the source code for all parts of the crossword project (GUI, modules, crossword class definitions, etc.), an operational solver GUI with sample crossword puzzles demonstrating the optimal functionality of the application (along with any compiled databases), and a final write-up of my strategies and findings. I will also have an oral presentation prepared which will highlight and summarize the content presented in the write-up.