# CRIS: Natural Language Processing of Crossword Clues

**Celia Abernathy**
Swarthmore College
caberna1@swarthmore.edu

**Lisa Bao**
Swarthmore College
lbao1@swarthmore.edu

## Abstract

We implement a single-module computational crossword solver applying natural language processing methodologies. The solver, called CRIS, first parses crossword puzzle clues using a unigram model to compute probabilities for possible answers to each clue from similar clues in a clue database. CRIS then executes one of three iterative algorithms to fill in the crossword puzzle grid with its best guesses: word-by-word, letter-by-letter, or combination fill methods. When tested on a selection of *New York Times* crossword puzzles from different days of the week, our results showed that CRIS could solve many puzzles well and a few puzzles perfectly using one of its three fill methods.

## 1 Introduction

American crossword puzzles[1] are a popular phenomenon which tests a solver's skill in the domains of both factual knowledge and linguistic intuition. A puzzle begins with a symmetric grid of solid black squares and numbered white squares, most often 15 by 15 squares in size, in which letters must be filled to form words or phrases that "cross" each other. Each numbered square corresponds to either or both of an "Across" clue and a "Down" clue, hinting at the respective word or phrase that should be filled into the grid starting at that number. Thus, a human solver may correctly answer only a portion of the puzzle based on the clues alone, but he or she is greatly aided by crossing letters. For instance, if the second letter of a five-letter answer is known to be R, then the probable options for the first letter are more limited than all 26 possibilities if one assumes that the answer is in standard English (although foreign words, abbreviations, and pop culture slang are fair game in solutions). More generally, crossword puzzles can be framed in two major parts: the clues must be interpreted and answered using natural language processing or information retrieval techniques, and then answers must be filled into the grid while preserving the crossing nature of the puzzle.

Perhaps the most well-known American crossword puzzles are those edited by Will Shortz and published daily in the *New York Times*, increasing in difficulty from Monday to Saturday, with a special larger puzzle on Sunday. Figure 1 illustrates one such puzzle, a themed Thursday crossword where the payoff theme entry is CHRISTMAS BONUS at 57A and 63A. This theme clue hints at the answers to the other theme clues at 17A, 22A, 34A, and 45A, where the respective letters from XMAS have been added as a "bonus" to the beginnings of each answer; the new answers are then clued with puns.

We chose to consider the 15x15-square Monday through Saturday puzzles from the *New York Times* as a testing data set. Our system, called Crossword Intelligence Solving (CRIS), uses a discounted unigram model to answer the natural language clues

[1] It is important to distinguish American crossword puzzles from another popular variant originating in Great Britain. The latter, which is also called a cryptic crossword, expects a very different style of cluing and slightly different grid constraints.


Figure 1: Complete solution grid to the Thursday, November 20, 2012 puzzle in the *New York Times*.

based on a database of previously occurring crossword clues, then tests several different solutions to the constraint satisfaction problem of filling in the crossword grid itself.

## 2 Related work

Although crossword puzzles had been previously studied as constraint satisfaction or information retrieval problems, Littman, et al's probabilistic solver called Proverb was the first to combine multiple expert modules into a centralized solver that then handles the crossword grid as a constraint satisfaction task (2002). First, a coordinator parses input and separates the crossword's clues from its grid. From the clues, each expert module produces a ranked list by probability of 0 to 10,000 candidate answers for each clue. Modules range from word-list matching to classical information retrieval or machine learning, as well as traditional natural language processing techniques. For instance, a Dijkstra module measures semantic relationships using graph distance, based on "the intuition that related words either co-occur with one another or co-occur with similar words"(Littman et al., 2002, 34). After all expert modules have returned, the merger renormalizes each probability list to create a single weighted list of possible answers to a given clue. This list is used by the final solver module to search for the best (maximum expected overlap with correct answers) solution to the whole puzzle while satisfying the grid constraints.

Littman, et al also utilize implicit distribution modules, which feed separately into the solver, to handle novel answers that are not found in the clue database. In these cases, the solver generates letter bigram probabilities after some letters of the answer have been filled in from crosses. The letter bigram module returns all remaining possible letter sequences of a given length, ranked by the same bigram probability distribution that is used by the solver during constraint satisfaction. Like most crossword puzzle solvers, Proverb is evaluated on daily newspaper puzzles from the *New York Times* and other sources, including difficult competition puzzles used in the annual American Crossword Puzzle Tournament. When tested on a selection of 70 *New York Times* puzzles divided into two categories, Monday through Wednesday and Thursday through Sunday, Proverb's performance declined as expected based on the *Times*' stated progression of increasing difficulty throughout the week.

Another automatic crossword puzzle solver–albeit for Italian-language cryptic crosswords–is Gori, et al's WebCrow project at the University of Siena (Gori et al., 2006). WebCrow's high-level architecture is similar to that of Proverb, which the authors took as inspiration, but information retrieval from the World Wide Web is used as the solver's primary source of knowledge. Additional modules, such as a word dictionary, specific structural rules, and a small clue database, are also passed into a merger module which combines the lists of candidate answers for a given clue based on both individual candidate probabilities and the confidence of each module's list. Finally, a constraint satisfaction solver is also required to find an admissible solution that maximizes the total number of correct answers.

Gori, et al place their primary focus on the WebSearch module of WebCrow. WebSearch's internal architecture consists of four components: retrieving relevant documents from the Web, extracting candidate answers, scoring and filtering the can-

didate lists, and estimating list confidence. The first component, retrieving relevant documents using Google, is also the most time-intensive, taking up "easily over 90% of time in the entire clue-answering process" (Gori et al., 2006, 272). Second, a list of unique words of the correct length, as well as bigrams of the correct length that occur more than once, are passed through a statistical information-retrieval filter and a morphological machine-learning filter. These candidate answers are ranked according to merged scores from each filter; since high recall is more important than precision in order to avoid eliminating a possible answer, low-scoring candidates are never pruned from the list. Finally, the module estimates the probability that its candidate list contains the correct answer to the clue using a trained neural network; this confidence probability is used by the merger to combine candidate lists from different modules.

Finally, cruciverbalist Matthew Ginsberg's recent 2011 publication on a solver called Dr. Fill represents a major milestone in the development of crossword-solving computer programs (Ginsberg, 2011). Dr. Fill converts a given crossword puzzle to a weighted constraint satisfaction problem, using novel heuristics based on the projected cost of filling a square of the grid with a particular level to solve the problem. Drawing on his own expert knowledge of the problem domain, Ginsberg exploits some of the numerous crossword conventions specific to this constraint satisfaction domain. For instance, Dr. Fill has modules dedicated to identifying a multiple-word answer, to identifying rebus[2] puzzles, to analyze part-of-speech in order to use the substitution test[3] that is convention for crossword clues, and to check for abbreviations in clues, which will indicate an abbreviation as the answer. However, other significant crossword puzzle conventions like puns and other wordplay–often signaled by a question mark–are ignored by the system.

Using a large dictionary of candidate answers and their probabilities, Dr. Fill aims to maximize the probability of correct fill in the puzzle grid using the probability of a candidate answer given a clue and answer length. Since some crossword fill, such as "MMYY for [Credit card exp. date format]", is not a natural language word or word sequence, the domain of possible answers should include all $26^N$ unique strings for an answer of length $N$ with a cost function to discourage the use of unlikely strings (Ginsberg, 2011, 862). Given such a large domain, the system employs limited forward propagation of crossing letters and uses value- and variable-selection heuristics to find the best solution. The heuristic scoring system includes five criteria: an exact or partial match for the clue, part of speech analysis, the human-evaluated merit[4] of the fill, abbreviations, and fill-in-the-blank clues. Words are filled into the puzzle in order of increasing total cost, with preference for later words that result in minimal cost increases of earlier words. Ultimately, Ginsberg achieves the impressive result of performance equivalent to the top 50 human solvers at the 2010 American Crossword Puzzle Tournament.

## 3 Methods

CRIS draws on a database of past clues and answers to model how clues indicate different answers. It uses this to estimate, for each clue in the puzzle, the probability of different possible answers. Then it considers these probabilities to determine how to fill in the puzzle, which is done iteratively.

### 3.1 Database

CRIS uses a database of clues and answers created by Matthew Ginsberg and found at `http://www.otsys.com/clue`. The database includes over 3.8 million clues, about half of which are unique, drawn from numerous published crosswords from sources including but not limited to *The New York Times*. More information about the database can be found in (Ginsberg, 2011), which uses it for Dr. Fill. For CRIS's purposes, we divide the database depending on the length of the answers, from 3 to 15

[2]Rebus crossword puzzles permit a single square in the grid to hold more than one letter simultaneously. Typically one word appears in a single square throughout the puzzle.

[3]Ginsberg explains this test as follows: "It must be possible to construct a sentence in which the clue appears, and so that the meaning of the sentence is essentially unchanged if the clue is replaced with the fill" (2011, 861).

[4]The relative merit of a crossword answer is based on subjective criteria such as lower-frequency letters, unusual combinations of letters, or non-obscurity of a word that meets other criteria. Approximately 100 crossword constructors scored 500 words each on "merit", and the remaining words were evaluated with a linear model matching these human evaluations (Ginsberg, 2011, 865).

letters. The number of clues for each answer length vary, with a greater number of clues for smaller answers, of which there are more in any given puzzle. There are the greatest number of clues, 460058, for 4-length answer, and the smallest number, 10868, for 14-length answers.

### 3.2 Generating answer probabilities

Any given crossword grid is non-unique, in that there are various combinations of words that would fit the layout (Ginsberg, 2011). The clues that accompany each puzzle grid determine which fill words are correct for that particular puzzle. Thus, it is important that CRIS be able to evaluate the probabilities of different possible answers to clues.

For each clue number–we use *number* here to refer to a set of squares that need answering, like "1A" or "13D", specifying a single clue and a single correct answer associated with it–CRIS creates a list of possible answers by checking the database of old clues for answers of matching length. Possible answers are assigned a score that can be incremented for each old clue connected to that answer. If the old clue matches the current clue exactly, a large bonus is added to that answer's score. CRIS also uses a unigram model to compute similarity between clues: every word in an old clue that appears in the current clue will increment the score for that answer. The amount of this increment is inversely proportional to the frequency of the word in the clue database as a whole, so that very frequent words like *of* do not indicate a great match between the current clue and old clue, whereas infrequent words like proper names will be adequately rewarded.

In formal terms, for a num $n$ with a clue $c_n$, the clue-based probability $P_c(w_n)$ of the answer to $n$ being $w$ is calculated using the set of old clues $K$ that apply to $w_n$. For each clue $k$ in $K$ we can define the set of unigram matches $U_k$, that is, the set of words that appears in both $c_n$ and $k$. We can also use a $count_K(u)$ to count the frequency of $u$ in $K$. Thus, each clue-based answer probability is:

$$P_c(w_n) = \sum_{k \in K} ( \sum_{u \in U_k} \frac{1}{count_K(u)} + (1 \text{ if } k = c_n))$$

For many of the answers in the database, the clues to them will have nothing in common with the current clue. However, we still want to consider these answers, since they could potentially be correct but clued in a novel way. The only words and phrases CRIS considers as candidate answers are from the clue database, so we want to cast as wide a net as possible. Thus, we smooth the scores using simple $+k$ smoothing, adding a very small amount to each score to give every answer a non-zero probability. Once a score has been assigned to each possible answer, the values are normalized to give the answer probabilities for that number.

### 3.3 Fill algorithms

Just like a human solver[5], CRIS fills in a puzzle iteratively, choosing a few letters or words that it is most sure about and writing them in, then reassessing the puzzle given these letters. We hypothesized that using an iterative algorithm would allow CRIS to fill in puzzles more quickly than a more computationally intensive algorithm like constraint satisfaction. We explored several different ways for CRIS's fill algorithm to proceed:

**Word by word**

In a naive word-by-word approach to crossword solving, CRIS picks the answer with the highest probability and writes it into the grid. Then it eliminates any potential answers that conflict with what is already in the grid, that is, "crossing" answers whose letter doesn't match the answer previously written in. CRIS normalizes the answer probabilities for each number. Then it repeats this procedure, choosing the answer that it is most sure about until the entire grid is full or it has no possible answers left. We consider this approach naive because it does not take into account how words will cross each other in selecting answers: an answer may seem probable based solely on the clue, but if it has a low probability of fitting with any of the possible crossing words, then the probability of entering it into the grid should be lower.

**Letter by letter**

The letter-by-letter algorithm fills in each square of the grid based on letter probabilities. Letter probability is calculated by summing the

[5] At least, one foolish or confident enough to solve in pen.

probabilities of all the possible answers that could lead to that letter. For example, the probability of *d* in the top left square is the sum of the probabilities of all the words that could go in 1A which begin with *d*, added to the sum of the probabilities of all the words that could go in 1D which begin with *d*.

For any given square $s$, there are two numbers whose answers will intersect in $s$: an across number $a$ and a down number $d$. For a letter $l$ and a square $s$, let $A$ be the set of possible answers for $a$ which have $l$ positioned so that it would be in square $s$, and let $B$ be the same for $b$. We can define the probability $P(l_s)$ of a $l$ being in square $s$ as:

$$P(l_s) = \sum_{w_a \in A} P(w_a) + \sum_{w_d \in D} P(w_d)$$

An alternative method for calculating $P(l_s)$ would multiply the across and down probability sums instead of adding. However, when we tried this, CRIS did very poorly. We speculate that because CRIS only has probabilities for answers from the clue database, it may be missing data for some answers that are in the actual puzzle. Thus, the sum of probabilities in one of the directions may result in 0. Multiplying the across and down probabilities means that the total letter probability might be 0, which causes problems if incorrect.

After normalizing each square's letter probabilities, the 5 letters with the highest probabilities are written into the grid. Any possible answers that conflict with these letters are eliminated and answer probabilities are renormalized for numbers that were affected by this elimination. Then the algorithm starts again by filling in the new top 5 letters. This approach takes into account the nature of crosswords, in which information from both across and down answers is important in determining any square. However, the algorithm is clearly not very much like a human solver, who would take crossing information into account but then fill in whole words.

**Combination**

The combination algorithm represents a combination of the letter- and word-based approaches. First, it calculates a normalized letter probability for each square. Then, it calculates new probabilities for each possible answer by multiplying the probabilities of each letter involved.

For a given word $w$ and number $n$, let $P(l_{w,n,i})$ be the probability of the $i$th letter of $w$ appearing in the appropriate square. Using $P_c(w_n)$ as the original clue-based probability, we can calculate a new probability for $P(w_n)$ using:

$$P(w_n) = P_c(w_n) * \prod_{i=0}^{length(w)} P(l_{w,n,i})$$

P(w) = Pc(w) * P(l0) * P(l1) * * P(ln) where P(lx) is the probability of the letter in position x of w appearing in the appropriate square

Now the new word probabilities are normalized, and the most likely word is chosen and written into the grid. Any conflicting answers are eliminated and the algorithm repeats until the grid is full or it gets stuck because it doesn't know any words that will fit into the current grid.

## 4 Results

We tested CRIS on 30 *New York Times* crossword puzzles from 5 successive weeks, from Monday 11/12/2012 to Saturday 12/10/2012. However, we omitted the Sunday puzzles because they are typically 23x23 squares in size rather than the most common 15x15. For each puzzle, we first calculated the answer probabilities using the clue database, and then passed these probabilities to each of the three different fill algorithms.

Figure 2 shows the average performance for each algorithm on each of the days of the week that we looked at. Performance is measured by the percentage of non-black squares filled with the correct letter. Hence, blanks and incorrect letters are judged equally. While this measure is not identical to the judging rules used in crossword competitions, where incorrect letters typically result in a much greater

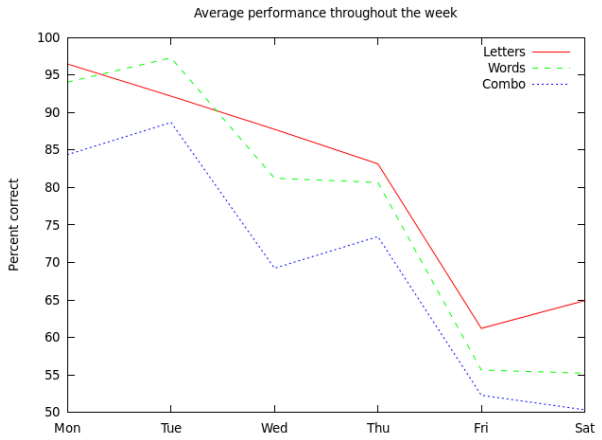


Figure 2: Average percentage correct for each of the three algorithms: letter-by-letter, word-by-word, and combination methods.

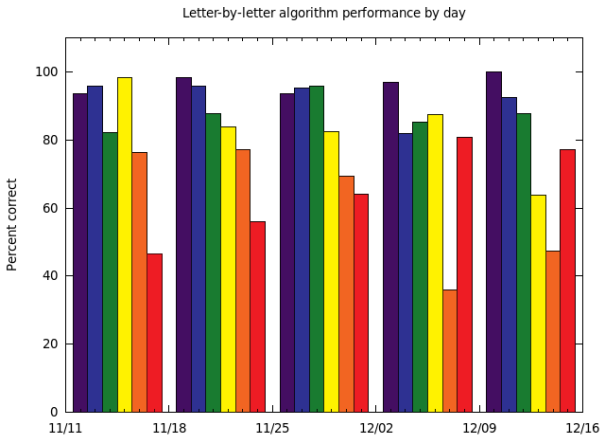


Figure 3: Percentage of squares correct by the letter-by-letter algorithm on each graph. Each day of the week is shown in a different color.

penalty compared to blanks, it makes for an easy-to-grasp metric for CRIS's success.

Overall, it is evident that CRIS's performance declines as the week goes on. The combination method performed least well compared to the other two methods, and the letter method improved over the word method. Specifically, the combination method averaged 69.733% correct over all 30 puzzles, the word method averaged 77.364% correct, and the letter method averaged 80.951% correct. However, as we can see with the Tuesday average, this relative performance is not accurate in all cases. There are many puzzles in which the word or combo method, or both, outperformed the letter method. In fact, if we only count which method was the most accurate for each puzzle, we find that the letter method is the most accurate for only half of the puzzles. For both the word and combination methods, there was one puzzle for which they were 100% correct and the other two methods made errors: Tuesday, Nov. 20, for the word method and Thursday, Nov. 15, for the combination method). All three methods did achieve 100% performance on one puzzle: Monday, Nov. 10.

Clearly, there is significant variability between puzzles, even on the same day of the week. For example, consider Figure 3, which plots the performance of CRIS's letter-by-letter algorithm on each of the 30 puzzles[6]. Although most weeks show an overall trend of decreasing performance, this trend is not wholly consistent due to individual variability.

## 5 Discussion

Although we were interested in seeing how all three algorithms perform when solving puzzles, we predicted that the word method will do worse than the other two methods because it does not account for crossing answers, a central feature of crosswords. Also, since CRIS can't backtrack, if it makes a mistake then the effect may cascade into a serious error. Filling the grid letter-by-letter not only takes into account crosses, but also cannot make mistakes as large as word-by-word. On the other hand, just filling in letters may prove shortsighted: even in a case where many crossing answers contain a certain letter, it could be that another letter is the only one that will eventually allow successful crosses on other parts of the words. Thus, we hypothesized that the combination method, which considers the interactions between letter and word choice, would be most successful. Furthermore, we predicted that CRIS's performance would decline throughout the week since later *New York Times* puzzles are harder than earlier ones, with tricky clues, novel answers, and clever themes which could throw CRIS off.

The data agrees with our intuition that CRIS's per-

[6]Although the puzzles are mostly shown in order, note that we had to skip Thursday, Nov. 22's puzzle, which was 16x16, and use Thursday, Nov. 8's instead; however, we plot this replacement puzzle as Nov. 22 in Figure 3 for aesthetic reasons

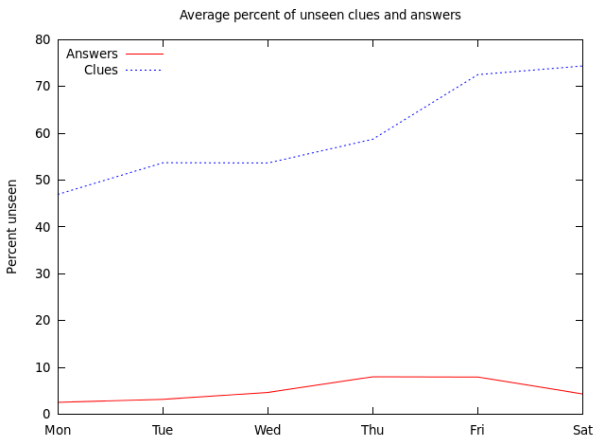


Figure 4: Average percentage of answers and clues that are not in the database.

formance will decline because the puzzles become harder as the week progresses. A harder puzzle may mean more obscure answers or answers that are combinations of other words or puns; more punny or thematic clues; or simply harder clues (for example, to clue the answer ALEX, the crossword constructor can choose between a well-known Alex from popular culture or a more obscure one). CRIS is detrimentally affected by all of these. If the correct answer is too obscure or is totally novel, it may not appear in the clue database, meaning that CRIS won't consider it at all. If the clues are hard or tricky, even if the answer appears in the clue database, CRIS may not make a strong connection between the current clue and old, different clues for the same answer. This failing contrasts with clues whose exact match appears in the database, for which CRIS is very confident about the answer.

We investigated this problem of unseen answers and clues by counting, for each of the 30 puzzles, how many of their answers and clues could not be found in the clue database. Figure 4 shows the average unseen answer and clue percentages for each day of the week. As expected, the number of unseen answers and clues tends to increase throughout the week. However, keep in mind that counting how many clues do not have an exact match in the database is only one factor in clue difficulty; of those without exact matches, there are some with very similar old clues and some without any.

Although all three fill algorithms show the same trend of decreasing performance over time, there remains a clear difference among their respective performance scores, as we saw in Figure 2. Surprisingly, the combination method tended to do the worst; but as we predicted, the letter method performed better than the naive word method. We suspect that filling in word-by-word, which both the original word method and the combination method do, is simply more error-prone. It is not clear why the combination method tends to be less accurate than the word method. Also, for many of the puzzles, the word or combination methods outperformed the letter method. There may be a certain quality to puzzles that makes one method better at solving them, but if so, this quality is not immediately apparent.

Both the word-by-word and combination method show an interesting pattern of good performance on Tuesday and Thursday relative to the other days. Tuesday and Thursday are days with a strong theme, often with a pun or trick element to them. However, the clues for shorter entries are often simple answers crossing the theme answers and tend to be more straightforward. Thus, it may be easier for CRIS to guess these answers because they are more likely to occur as exact matches in the database of old clues.

## 6 Conclusions and future work

CRIS solves crossword puzzles in two main steps. First, it gathers a list of possible answers to each clue and their probabilities by using a unigram model to find similar clues in a clue database. Then it executes an iterative algorithm that fills in the grid with its best guesses. We compared three different grid-filling algorithms: a naive word-by-word fill based on answer probabilities, a letter-by-letter method that determines the probability of different letters based on the answers that cross there, and a combination method that takes the letter probabilities and uses them to generate new word probabilities. We tested CRIS on 20 *New York Times* crossword puzzles which varied in difficulty based on the day of the week and found that the letter method tended to do best, averaging 80.951% of the letters correct throughout the week, with better results for earlier (easier) puzzles. There were 3 puzzles which

CRIS solved completely correctly. These results are promising, given that CRIS is much simpler than previous crossword solving systems like Dr. Fill (Ginsberg, 2011).

Since different grid-filling algorithms perform better on different puzzles, it would be interesting to explore what makes certain puzzles better suited for one or the other. Perhaps the algorithms – especially the combination, which appeared so promising but did not outperform either the word or letter methods on their own – can be refined to produce better results. A future extension of CRIS might also include other methods of generating possible answer lists. Using information retrieval to actually look up the answer to clues, rather than relying solely on previous clues, would certainly be helpful. Hence, CRIS could evolve into an architecture with various modules to deal with different types of clues, a design used by other major crossword solving systems.

## 7 Acknowledgments

We would like to thank Professor Richard Wicentowski for his guidance in shaping and implementing this project. We are also grateful to Matthew Ginsberg, creator of Dr. Fill, for making his extensive crossword clue database available in the public domain, thus allowing us to create CRIS.

## References

M.L. Ginsberg. 2011. Dr. fill: crosswords and an implemented solver for singly weighted csps. *Journal of Artificial Intelligence Research*, 42(1):851–886.

M. Gori, M. Ernandes, and G. Angelini. 2006. Cracking crosswords: the computer challenge. *Reasoning, Action and Interaction in AI Theories and Systems*, pages 265–286.

M.L. Littman, G.A. Keim, and N. Shazeer. 2002. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1):23–55.