1 Introduction

1.1 Motivation

Puzzle solving is an interesting application of computer science. Some mathematically based problems are particularly well-handled by computers. Language, on the other hand, is substantially more complex and harder to translate into a machine-solvable problem. Normal crossword puzzle solvers, for example, consist mostly of clue databases, mapping a clue to previously observed solutions.

Cryptic crosswords are an interesting subset of language puzzles. The language problem involves translating a definition or “sense” into a word. This is known as a reverse dictionary lookup (normal dictionaries map words to definitions; reverse dictionaries map definitions to words) [1]. Since cryptic crosswords obey certain rules that we can model somewhat effectively, they are a rule-satisfying subset of the reverse dictionary lookup problem. In some cases these rules can be so straightforwardly implemented that our solver might be more effective than a human, which is ultimately the goal of a good puzzle solver.

Other applications of particularly good reverse dictionaries are areas that involve intent guessing. This is extremely relevant to interpreting web search queries.

1.2 Introduction to cryptic crossword puzzles

Much of the following section has been adopted from the wikipedia page [2]. Cryptic crosswords consist of a grid and clues, much like normal crosswords. However, there is substantially less overlap in the grid, and the clues take a particular, “cryptic” form. But for a few exceptional cases, crypticals generally follow certain rules:

1. A length for the solution is provided
2. A definition for the the solution word comes either at the end or the beginning of the clue
3. The solution word can be constructed by obeying a set of cryptic, wordplay rules with the “half” of the clue that is not part of the definition
4. The wordplay rules are often indicated by sets of indicator words
To allow the reader to understand the solution method, we now provide some examples of the forms of cryptic clues, organized alphabetically. Following each clue, we provide a level of difficulty of automated solution, on a scale of 1-5, 1 being easiest and 5 being hardest. This difficulty level is based on the following analysis of solution methods.

1.2.1 Anagrams

Rearrange part of the clue.

*Chaperone shredded corset (6)*

“shredded” is a word that indicates that some part of the clue should be anagrammed (literally shredded) to give a new word that satisfies the definition portion of the clue. Other anagram indicators include “shred,” “stir,” and other semantically similar words. In this case, corset can be rearranged to escort, which is a synonym of chaperone.

Anagrams can be complicated in the following ways:

- An anagram of multiple words is required, e.g.,
  *No heart gets broken by someone else (7)*
  is an anagram indicated by “broken.” An anagram of “no heart” is “another” which matches the definition “someone else.”

- An anagram involving initial letters and some words.

- An indirect anagram, where a synonym of a word in the solution should be anagrammed, e.g., in
  *Chew honeydew fruit (5)*
  We first take melon as a synonym of honeydew and “chew,” or anagram it to “lemon” to get a fruit.

The above form of anagram solution lends itself easily to a computed answer. A simple set of steps to solution might be:

1. If an indicator word exists
2. Anagram words, combinations of words, and combinations of initial letters and words
3. See if any of the valid anagrams (i.e. real English words) match the definition. From here on, this step is denoted as “Compare to definition”

*Complexity: 1* Anagrams have indicators, and computing anagrams should be straightforward, with a high probability of a definite solution

1.2.2 Charades

Combine words to get a solution.

*Outlaw leader managing money (7)*

Solutions involve taking synonyms of two or more words and joining those synonyms together. In this case, we join “ban,” a synonym of “outlaw” and “king,” a synonym of “leader,” to get “banking,” which can be defined as “managing money.”

Charades do not have indicator words, so possible steps to solution:

1. Find synonyms of words in clue
2. Look for combinations of synonyms that match the clue length and are valid English words
3. Compare word combinations to definition

*Complexity: 5* There are no indicators, and pairing lists of synonyms is $O(n^2)$ for two word combinations
1.2.3 Containers

Place some letters inside a word or other set of letters.

Apostle’s friend outside of university (4)

Put “pal,” from “friend,” outside of “U” for “university” to get “paul,” an apostle. Note here we encounter an additional level of complexity: we must find an instance of apostle rather than a definition. Another example of this sort of complication would be Turkey for country.

Containers do have indicator words, so steps to solution:

1. Find indicator word (note here this also clues how to compute the solution, e.g. “outside of”)
2. Get synonyms and initial letters and shuffle according to the clue word. Note that exactly following the clue word is probably more difficult than just computing anagrams.
3. Check definition

Complexity: 3-4 There are indicators, but we must find synonyms before anagramming.

1.2.4 Deletions

Remove a letter from a word (beginning, end, or middle) to get a solution.

Beheaded celebrity is sailor (3)

Substitute “star” for “celebrity” and behead “star” to get “tar,” a synonym for “sailor.”

Solution steps:

1. Find indicator
2. Get synonyms and perform beheadment, curtailment, or internal deletion. Note that the first two are computationally easier.
3. Check definition

Complexity: 3 The steps to solution are very similar to those for Containers, but we avoid having to combine different sets of letters and words.

1.2.5 Double definitions

These clues lack wordplay. Instead, the solution will match a definition at the beginning and end of the clue.

Bird country (6)

Gives “turkey,” which is both a bird and a country.

These clue types lend themselves to reverse dictionary lookups [1].

Complexity: 4 Depending on the efficiency of our reverse dictionary and semantic analysis, we might need to split the clue in many positions (total possible split locations is \((\text{number of words in clue}) - 1\) and attempt \(2 * (\text{number of split locations}) = 2 * (\text{words} - 1)\) reverse lookups. Without an efficient reverse dictionary, this is extremely costly.

1.2.6 Hidden words

The solution is hidden in the letters of the clue.

Cook some lunch effectively (4)

“chef” occurs at lunch effectively and matches “cook.” “Some” functions as a hidden indicator.

Hidden solutions can be complicated by taking initial letters, or odd/even letters.

Solutions steps:

1. Find indicator word
2. Search for valid words in string of text (Note not all strings should be considered, and the possible complications mentioned above must be handled carefully, though in a straightforward computational fashion.)

3. Check definition

   Complexity: 1  Indicator words and the presence of the solution word in the letters of the clue lend this solution type to computation.

1.2.7 Other types

   • Homophones: Taking a homophone to get a solution. E.g., We hear twins shave (4) gives twins\(\rightarrow\)pair\(\rightarrow\)pare, matches shave. “Hear” indicates homophone.

   • Reversal: Flip a word to get a solution. E.g., Returned beer fit for a king (5) gives beer\(\rightarrow\)lager\(\rightarrow\)regal, matches king. “Returned” indicates reversal.

1.2.8 More complex types

   A discussion of the following clue types can be found on the cryptic crosswords wikipedia page [2]. Combinations involve combinations of different clue types, as the name suggests.

   • Cryptic definition
   • Spoonerism
   • Literal
   • Combinations

1.3 Basic implementation

   Barring the consideration of combination clue types, we can follow a relatively straightforward clue solving technique:

   ```python
   for solver in solver set:
       if (clue doesn’t have indicator word):
           continue
       if (solver requires initial synonimization):
           clue = clue.synonimize()
       possible solutions = clue.manipulate()  # anagram, search for hidden, reverse, etc.
       for soln in possible solutions:
           if soln is not valid English:
               continue
           if clue.checkDefinition(soln)  # i.e., does this solution match the definition
               add soln to list of possible solutions
   return list of possible solutions
   ```

   We proceed to implement as follows:

   1. Class Clue: An incoming clue string is parsed into a string of terms and a length

   2. Class Term: A Clue is made up of Terms that contain a word and a synonym / definition set (syn-def set) for that word. We compute syn-def sets only once per term in the clue.

   3. Class Solver: implements method getSolutions(Clue). We have a solver for each clue type that we implement. The anagram solver will compute valid anagrams, the hidden solver will look for valid words hidden in the clue text, etc.

   4. Class Solution: A solution object tracks the formation of a given solution word, including the type of Solver, the indicator word, and notes on how the solution word was reached.

   5. Algorithm: Deriving syn-def sets. We make use of Python’s Natural Language Toolkit (NLTK) and a thesaurus database we generate per the algorithm detailed below.
6. Algorithm: Syn-def intersection scoring

7. Algorithm: Checking solutions. We use a set similarity algorithm akin to a cross-product to compute the nearness of a solution word to a clue’s probable definition. This is a comparison of multiple syn-def sets.

**Deliverables**

1. An anagram solver
2. A hidden solver
3. Possible attempts at other solvers depending on success and timeliness of the first two techniques
4. Testing interface