Cryptic Crossword Clues: Generating Text with a Hidden Meaning

David Hardcastle
Open University, Milton Keynes, MK7 6AA
Birkbeck, University of London, London, WC1E 7HX
d.w.hardcastle@open.ac.uk
ahard04@dcs.bbk.ac.uk

Abstract

This paper discusses the generation of cryptic crossword clues: a task that involves generating texts that have both a **surface reading**, based on a natural language interpretation of the words, and a **hidden meaning** in which the strings that form the text can be interpreted as a puzzle. The process of clue generation realizes a representation of the hidden, puzzle meaning of the clue through the aggregation of chunks of text. As these chunks are combined, syntactic and semantic selectional constraints are explored, and through this language understanding task a meaningful surface reading is recovered. This hybrid language generation/language understanding process transforms a representation of the clue as a word puzzle into a representation of some meaningful assertion in the domain of the real world, mediated through the generated multi-layered text; a text which has two separate readings.

1 Introduction

This paper discusses a system called ENIGMA which generates cryptic crossword clues: fragments of text that also have a **hidden meaning**, quite different from the **surface reading**. This raises an interesting research question: how to generate text that has multiple layers of meaning based on different syntactic rules and different semantic interpretations. The input to the realizer is a production of a high-level cryptic clue grammar whose terminals are the strings that participate in the puzzle presented by the clue. These conceptualizations of possible crossword clues contain no implicit syntactic or semantic information, and so a mechanism is required to ensure that the resulting surface text is syntactically correct and semantically appropriate while the meaning of the text, derived directly from the input, is not disturbed during lexicalization. As with computational humour and poetry generation the process of generation is unusual in that the content is not specified in the input (Ritchie, 2001; Manurung, 2000), and this leads to tractability problems when considering the wide range of lexicalization options (see also Ritchie, 2005: 4) requiring a bespoke solution.

1.1 Cryptic Crossword Clues

The cryptic crossword clues generated by ENIGMA consist of two separate indications of the solution word, one of which is a definition, the other a puzzle based on its orthography. Consider, for example, the following simple clue for *noiseless*:

```
  Still wild lionesses (9)
```

Here *noiseless* is represented both by the synonym *still* (the definition) and a wordplay puzzle (an anagram of *lionesses*) indicated by the convention keyword *wild*. All of the clues generated by the system conform to Ximenean conventions (Macnutt, 1966), a set of guidelines that impose restrictions on inflection and word order to ensure that clues are ‘fair’ and also encourage the use of homographs and convention vocabulary to make them cryptic in nature.

It is important to note here that there are two separate readings of this clue: a surface reading in which the clue is also a fragment of English text, and the puzzle reading required to solve the clue. In the surface reading the word *still* is an adverb qualifying the adjective *wild*, while in the puzzle reading it is an adjective that is a synonym for *noiseless*.

There are many different types of crossword clue wordplay, including anagrams, homophones, writing words backwards, appending words together, and more besides. ENIGMA generates clues using seven of the eight main types listed in (Macnutt, 1966) and can also generate complex clues with subsidiary puzzles. This coverage combined with the richness of lexical choice in cryptic crossword convention vocabulary means that it is not uncommon for ENIGMA to generate several hundred valid clues for a single input word.

1.2 Requirements for Generation

Given a particular solution word, such as *noiseless*, the first step for the system is to determine the different ways in which the letters of the solution could be presented as a puzzle. For example, the basis of the clue could be that *noiseless* is an anagram of *lionesses*, or that it can be formed by running *noise* and *less* together, or that it is composed of a river (*Oise*) followed by the letter *l* all placed inside the word *ness*. In its present form ENIGMA locates 154 such formulations for the input word *noiseless*. Each of these formulations can be represented as a clue tree under ENIGMA's domain grammar for cryptic crosswords in which the terminal elements are the strings used to compose the solution word. The sample clue tree in Figure 1 represents the fact that *noiseless* can be formed by running *noise* and *less* together, a puzzle type known as a Charade (Macnutt, 1966; Manley, 2001).

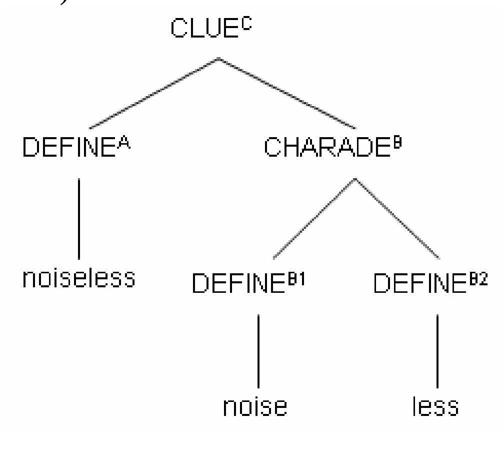

Figure 1. A clue tree that represents appending *noise* and *less* to form *noiseless*.

These clue trees contain no linguistic information - the terminals should be thought of as strings not as words. To lexicalize this data the system must construct a fragment of natural language that can be reinterpreted - through the resolution of homographs and a knowledge of special conventions - as a valid cryptic clue puzzle based on this non-linguistic structure. Along the way the syntax and semantics of the puzzle reading must not be disturbed or the clue will lose its hidden meaning. At the same time, the natural language syntactic and semantic information that is missing from the input data must be imposed on the clue so that a valid surface reading is achieved.

2 Chunk by Chunk Generation

A complete clue does not need to be a sentence, or even a clause, it can be any valid fragment of text, and ENIGMA takes advantage of this fact to simplify the generation algorithm. The clue tree shown in Figure 1 is realized through a process of composition. First the symbol labeled A is realized. Next B1 and B2 are realized individually and then combined to form B. Now, A and B can themselves be combined to form the clue. Each realization is a fragment of text, and I refer to each of these fragments as a *chunk*, although I note that they are rather different from chunks based on major heads (Abney, 1989), for the reasons set out below. To implement this process the system needs to be able to do two things: create chunks for each terminal in the clue tree, and merge chunks into successively larger ones until the root of the tree is reached. This recursive process enables ENIGMA to construct complex clues with subsidiary puzzles using the same implementation it uses for simple puzzles.

2.1 Word Order

When chunks are combined together they cannot interleave or nest. The reason for this is that each chunk represents a part of the hidden meaning of the clue, and word order is central to its interpretation. This is why ENIGMA uses a flat structure rather than a tree structure to build up the clue.

3 Implementation

Each chunk can attach to another chunk to its left, to its right, or via an intermediary word or phrase, such as a conjunction, something I call 'upward attachment'. The grammar that underpins these attachments is encoded as a set of three *extension points*[1] to each chunk: one specifying the relationships that can occur to the left, another those that can occur to the right, and the third specifying upward attachments. For example the chunk *wild lionesses* has, amongst many others, an extension point to the left indicating that it can attach as direct object to a verb, one to the right indicating that it can attach to a verb as subject and an upward attachment through which it can attach via a coordinating conjunction to another noun.

In addition to specifying the relationship and target type each extension point also specifies an *erasure*[2] for the chunk to which it belongs - this erasure indicates a word in the chunk that can stand in for the chunk as a whole when determining attachment. It is important to note that the erasure is not equivalent to the syntactic head and that different extension points on the same chunk may have different erasures. For example, in addition to looking for a verb to the left, the chunk *wild lionesses* also has an extension point looking for an adverb to the left, since an adverb could qualify the adjective *wild*. Therefore, the extension point looking for a verb erases *wild lionesses* to *lionesses*, so that a verb chunk looking for a noun to its right as direct object will accept it, whereas the extension point looking for an adverb erases this same chunk to *wild*, so that an adverb looking to its right for an adjective to qualify could also accept it. In this way the concept of erasure makes it possible for a wider variety of syntactic dependencies to be encoded in the same way on a single chunk, enabling polymorphic behaviour.

In some respects the extension points and associated erasures encoded onto each chunk act like the categories on functors in Combinatory Categorial Grammar (Clark et al, 2002), or edges on the agenda used in chart generation (Kay, 1996) as they specify the type and directionality of the arguments available and the type of the result. However, in addition to this syntactic information the grammar also provides the mechanism through which semantic selectional constraints are enforced. The erasures do not just specify a type (such as noun or adjective) but also a **member** of the chunk: *wild* or *lionesses* in this case. This enables the erasures to be used to determine which semantic checks are required to validate the attachment, adding to ENIGMA's implementation of chunks the semantic constraints that Abney notes as missing from his formulation (1989: 15). For example, if the chunk *wild lionesses* attaches to a chunk to its left that erases to a verb and is looking for a direct object then the extension point governing this attachment enforces syntactic correctness, but this is not enough. Since the clue tree only contains information about crossword conventions a separate semantic check is now required to ensure that it makes sense for the verb to take the noun *lionesses* as its direct object, and this semantic check will be performed using the relation and the erasures as arguments.

So, for example, when the chunk *still* is combined to the left of *wild lionesses* the system performs a semantic check to ensure that *still* can qualify *wild*. If the verb *calm* (an alternative homograph of a synonym for *noiseless*) is attached to the left then the system checks that *lionesses* can be the direct object of *calm*, and so on.

4 Sample output

Figure 2 depicts system output from ENIGMA representing the sample clue given in the introduction. Since so many clues are generated the system also generates a list of justifications which it uses to determine a score and rank the clues. The output shown in Figure 2 only includes information that relates to this paper; the full listing also contains information about the structure of the clue and the difficulty of the clue as a word puzzle. All of the explanatory text is generated using templates.

```
Clue for [noiseless]
Clue [Still wild lionesses (9)]
POS [still/AV0 wild/AJ0 lionesses/NN2]
Homograph pun: to solve the clue 'still'
must be read as Adjective but has surface
reading Adverb
Sense: dependency fit 'wild lionesses' of
type Adjective Modifier characterized as
'inferred'
```

[1] The term extension point is more commonly used to define the interfaces to plug-in components in extensible computer systems.

[2] In Object-Oriented Programming an erasure is a simplification or genericisation of a type through some interface, see for example (Bracha et al, 2001).

```
Attachment: 'wild lionesses' attached via
type Attributive Adjective Modifier
Sense: 'still wild' sense-checked for In-
tensifying Adverb attachment using the
lexicon
Attachment: 'still wild' attached via
type Adverbial Qualifier (of Adjective)
Thematic Fit: 'wild' and 'lionesses'
share common thematic content.
```

Figure 2. Sample output from ENIGMA.

5 Discussion

ENIGMA constructs clues using their hidden meaning as the starting point. Lexical choice is very unrestricted, while word order is quite tightly constrained. This leads to combinatorial explosion in lexical choice, but of the intractably large number of possible productions for each clue very few also function as viable fragments of natural language. ENIGMA's approach is to work through the structure of the hidden clue and determine constraints on the surface reading on the fly. The compositional process reins in the combinatorial explosion by pushing language constraints down to the most local level at which they can operate.

ENIGMA uses various generic language understanding resources built specifically for the application during the generation process to ensure that the syntactic relationships behind the clue's surface reading are semantically supported.

- A Collocational Semantic Lexicon mined from British National Corpus and augmented using WordNet determines if a proposed dependency relation between two words is semantically probable (Hardcastle 2007). This lexicon is used to impose selectional constraints on syntactic dependency relations, such as between a verb and its direct object.
- A Word Association Measure based on a distributional analysis of data in the British National Corpus is used to evaluate the thematic coherence of the clue (Hardcastle 2005).
- A Phrase Dictionary derived from the Moby Compound word list[3] is used to identify aggregations that result in the creation of multi-word units such as compound nouns or phrasal verbs.

The resulting clue texts are syntactically and semantically valid under the symbolic language grammar of the domain, and at the same time are plausible fragments of natural language. I plan to perform a mix of qualitative and quantitative evaluations on a set of generated clues, a reference set of clues published in newspapers and a set of control clues generated with no syntactic or semantic constraints, grouped into subsets that share the same solution word.

[3] http://www.dcs.shef.ac.uk/research/ilash/Moby/.

References

S. Abney. (1989). Parsing by Chunks. In *The MIT Parsing Volume*, edited by C. Tenny. The MIT Press.

G. Bracha, N. Cohen, C. Kemper, S. Mark, M. Odersky, S.-E. Panitz, D. Stoutamire, K. Thorup, and P. Wadler. (2001). *Adding generics to the Java programming language: Participant draft specification*. Technical Report, Sun Microsystems.

S. Clark, J. Hockenmaier, and M. Steedman. (2002). Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, pages 327–334.

D. Hardcastle. (2007). *Building a Collocational Semantic Lexicon*. Technical Report BBKCS-07-02. Birkbeck, London.

D. Hardcastle. (2005). An examination of word association scoring using distributional analysis in the British National Corpus: what is an interesting score and what is a useful system? In *Proceedings of Corpus Linguistics*, Birmingham.

M. Kay. (1996). Chart Generation. In *Proceedings of the 34th Meeting of the ACL*, pages 200-204.

D. S. Macnutt. (1966). *On the Art of the Crossword*. Swallowtail Books.

D. Manley. (2001). *Chambers Crossword Manual*. Chambers.

H. Manurung, G. Ritchie and H. Thompson. (2000). *Towards a Computational Model of Poetry Generation.* In *Proceedings of AISB Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, pages 79-86.

G. Ritchie. (2001). Current Directions in Computational Humour. In *Artifical Intelligence Review 16(2)*, pages 119-135.

G. Ritchie. (2005). Computational Mechanisms for Pun Generation. In *Proceedings of the 10th European Natural Language Generation Workshop*, pages 125-132.