Solving Substitution Ciphers

Sam Hasinoff
Department of Computer Science, University of Toronto
hasinoff@cs.toronto.edu

Abstract

We present QUIPSTER, an experimental system for the automatic solution of short substitution ciphers (Cryptoquotes). The system operates using an n-gram model of English characters and stochastic local search over the space of $26! \approx 4 \times 10^{26}$ possible keys. Experimental results show a median of 94% cipher letters correctly decoded, which is typically good enough for an unskilled human to finish decoding the cipher with minimal additional effort. Extensions incorporating a dictionary with word frequencies and a database of word patterns are also discussed.

1 Introduction

In Arthur Conan Doyle's short story "The Adventure of the Dancing Men" (1903), the protagonist Sherlock Holmes solves a murder mystery by realizing that a series of unusual stick figure drawings (Figure 1) are actually messages encoded using a substitution cipher [5]. The substitution cipher is a well-known classical cipher in which every plaintext character in all its occurrences in a message is replaced by a unique ciphertext character.

Figure 1: Dancing men ciphertext from "The Adventure of the Dancing Men" (1903).

Thus, each permutation of the 26 letters of the English alphabet (there are $26! \approx 4 \times 10^{26}$ in total) gives a unique key for encrypting a message. If a particular permutation is used to encrypt a message, then the inverse of that permutation can be used to decrypt it. A worked out example of a substitution cipher is given in Figure 2.

Decoding substitution ciphers is a popular activity among amateur cryptographers and people who enjoy word puzzles. Substitution ciphers of famous quotes appear in many newspapers (near the crossword puzzle and the jumble) under the title of Cryptoquotes or Aristocrats. For the purposes of this paper, we assume that punctuation is given (spaces and apostrophes are particularly helpful) and that capitalization is not preserved. While it is an unwritten rule of Cryptoquotes that no character encrypts to itself, for the sake of generality we do not make this assumption. Note that Cryptoquotes typically contain 50–250 characters of ciphertext.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ      alphabet
ZKLYQPNIFJDCUTVWSXMAGOEBHR      encryption key, e
TXLKWIUYHJBCSGVFEZQNMOPRDA      decryption key, d = e^-1

plaintext
THE MAN WHO DOES NOT READ BOOKS HAS NO ADVANTAGE
OVER THE MAN THAT CAN NOT READ THEM. --MARK TWAIN

ciphertext
AIQ UZT EIV YVQM TVA XQZY KVVDM IZM TV ZYOZTAZNQ
VOQX AIQ UZT AIZA LZT TVA XQZY AIQU. --UZXD AEZFT
```

Figure 2: Example substitution cipher.

In principle, substitution ciphers can be solved by exhaustively searching through the (astronomically large) key space for the key that produces the decrypted text most closely resembling meaningful English. Instead, human cryptographers exploit patterns and redundancy in the English language to greatly narrow their search. The information content of an English character has been estimated by various methods to be about 1.5 bits [10]. Hence the redundancy of English is about $\lg 26 - 1.5 \approx 3.2$ bits per character.

As the amount of available ciphertext increases, solving substitution ciphers becomes easier. The *unicity distance*, defined as the entropy of the key space divided by the per-character redundancy, is a theoretical measure of the minimum amount of ciphertext required by an adversary with unlimited computational resources. For substitution ciphers, the unicity distance is $\lg(26!)/3.2 \approx 28$ characters, which is roughly in agreement with the abilities of skilled human cryptographers [12].

Redundancy in English text manifests itself in a variety of forms:

- The statistics of characters strings are highly non-uniform. For example, `NT` is more frequent than `BT`, and `JX` never seems to occur.
- The vowels `AEIOUY` and consonants associate with special patterns.
- English text is constructed from limited vocabulary and words appear with highly non-uniform frequencies. Indeed, about half of all English text belongs to a miniature vocabulary consisting of the 135 most frequently occurring English words [7].
- The statistics of word strings are also non-uniform. For example, the string `LORD OF` appears more frequently than `LORD ELBOW`.
- Finally, there are constraints due to higher level semantics distinguishing meaningful English from nonsense. Human cryptographers excel at using this sort of information, but representing this for a computer is extremely difficult.

To give a concrete example of redundancy in English, single character frequencies (including the apostrophe) are shown for a sample corpus in Figure 3.

2 The QUIPSTER System

We developed an experimental system called QUIPSTER for automatically solving Cryptoquotes. The solver consists of two main components, a generic stochastic local search (SLS) method for navigating the key space, and a scoring function for evaluating the goodness of various keys. The scoring function for a particular key is defined as the log-likelihood of an n-gram language model applied to the ciphertext decrypted using that key.

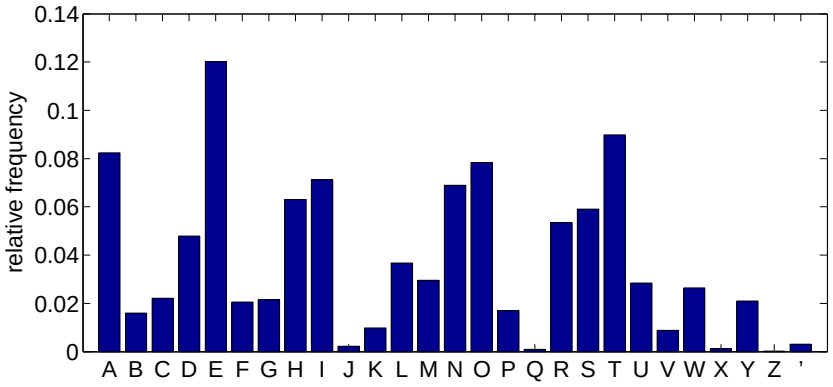

Figure 3: Relative character frequencies from *Great Expectations* (1860-1861).

2.1 Search Method

Our search method, given in Algorithm 1, is a simple form of stochastic local search [8] over the key space. The stochastic local modification step we use is to swap two letters of the key at random. To avoid being trapped in bad local minima, we restart from the beginning a fixed number of times and keep track of the best key overall. For future work, more sophisticated search strategies like tabu search, which explores several hypotheses simultaneously, may prove beneficial.

Algorithm 1: SOLVER($puzzle, num\_trials, num\_swaps, scoringFunction$)

input : substitution cipher $puzzle$, parameters num_trials and num_swaps controlling the amount of computation, and scoring function $scoringFunction$

output : best decryption key found $best\_key$ and its corresponding score $best\_score$, locally maximizing the scoring function

$best\_score \leftarrow -\infty$
for $i \leftarrow 1$ *to* num_trials **do**
  $key \leftarrow$ random permutation of the alphabet
  $best\_trial\_score \leftarrow -\infty$
  for $j \leftarrow 1$ *to* num_swaps **do**
    $new\_key \leftarrow key$ with two of its letters swapped randomly
    $score \leftarrow$ score $puzzle$ using $scoringFunction$ after decrypting it with new_key
    if $score > best\_trial\_score$ **then**
      $key \leftarrow new\_key$
      $best\_trial\_score \leftarrow score$
    endif
  end
  if $best\_trial\_score > best\_score$ **then**
    $best\_key \leftarrow key$
    $best\_score \leftarrow best\_trial\_score$
  endif
end
return $\{best\_key, best\_score\}$

2.2 N-gram Model

In computational linguistics, an n-gram model refers to an $(n-1)$-th order Markov model of language [10, 3]. Here we consider n-gram models of characters, but such models can also be constructed on the level of words. Most practical systems employ bigrams ($n = 2$) or trigrams ($n = 3$).

By the Markov model assumption, the probability of a character depends only on the previous $n-1$ characters. Using w_i^j to denote the string $w_i \cdots w_j$ we have:

$$p(w_1^l) = \prod_{i=1}^{l} p(w_i | w_1 \cdots w_{i-1}) \approx \prod_{i=1}^{l+1} p(w_i | w_{i-n+1}^{i-1}). \quad (1)$$

To make $p(w_i | w_{i-n+1}^{i-1})$ meaningful for $i < n$, we pad the beginning of every word with $n-1$ distinguished symbols (\$). We also pad every word with a distinguished symbol (ˆ) at the end. For example, `RINGS` is padded to become `$$RINGS^` for a trigram model.

The maximum likelihood estimate for the probability $p(w_i | w_{i-n+1}^{i-1})$ is simply the relative frequency:

$$p_{ML}(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})}, \quad (2)$$

where $c(w_{i-n+1}^{i})$ is the raw count of the n-gram w_{i-n+1}^{i} over some representative corpus of English text. Our vocabulary of characters V includes the alphabet, the apostrophe, and the two distinguished symbols, for a total of $|V| = 29$. The lexer picks out all valid words from the corpus, possibly with internal apostrophes.

We can use an n-gram model to evaluate the likelihood of a novel piece of text (generated independently of the training corpus), and obtain a rough measure of its English-ness. Thus, we define the scoring function for a particular key as the log-likelihood of our n-gram model applied to the ciphertext decrypted using that key. Note that the n-gram model can also be used generatively to produce English-like text, but this will produce mainly nonsense words.

2.2.1 Smoothing techniques

One problem with standard n-gram models is that low-frequency n-grams may be missing entirely from the training corpus. Thus, the occurrence of even one such zero-frequency n-gram would cause the entire test set to be assigned a likelihood of zero.

Smoothing addresses this problem by re-estimating the probabilities of all $|V|^n$ n-grams, shifting some of the probability from the higher-frequency n-grams to the low- and zero-frequency n-grams. A thorough survey of smoothing techniques for n-grams (on the level of words) was presented in [3] and we have adopted their notation.

For this project, we implemented three different smoothing methods: Witten-Bell smoothing, absolute discounting, and a simple *ad hoc* smoothing method.

Witten-Bell smoothing (method C, as originally described in [16]) is an elegant smoothing technique first developed for text compression. The key concept is to use a count of n-grams seen at least once to re-estimate the count of the unseen n-grams. Witten-Bell smoothing is defined recursively as a linear interpolation of the maximum-likelihood estimate and the lower-order $(n-1)$-gram model. As a base case, for the 0-gram model, we take the uniform distribution.

In absolute discounting [13], instead of multiplying by some discount factor, we subtract a fixed discount $D \leq 1$ from every non-zero count. We estimate the optimal fixed D as $D = \frac{n_1}{n_1 + 2n_2}$, where n_k is the number of n-grams seen k times.

The *ad hoc* method we devised doesn't even form a proper Markov transition matrix, as the conditional probabilities do not sum to 1. For zero-frequency n-grams we take the probability as $\exp(-8.0)$, otherwise we simply take the original maximum-likelihood estimate.

3 Experimental Results

QUIPSTER was written entirely in C++ and the n-gram scoring function was somewhat optimized. Execution speed is about 2 seconds per puzzle (with 15 trials of 10^4 swaps) on a mid-level Pentium 4.

The n-gram model was trained on the text of *Great Expectations* (1860-1861) by Charles Dickens [4] with smoothing as previously described. All substitution ciphers were generated from Faisal Jawdat's large personal collection of quotes [9]. This eclectic collection comprises 5236 quotes, not all suitable as official Cryptoquotes because of their highly varying lengths and occasional vulgarity. The collection is a good mixture of popular sayings as well as classic, political, and literary quotes. It contains plenty of wordplay and jokes (many of which might be classified as geek humour).

As shown in Figure 4, the effect of the different smoothing methods was almost negligible. Moreover, there appear to be no benefits to using high order n-gram models with $n > 4$. For the remainder of this paper, the smoothing method was set to absolute discounting and a trigram model was used.

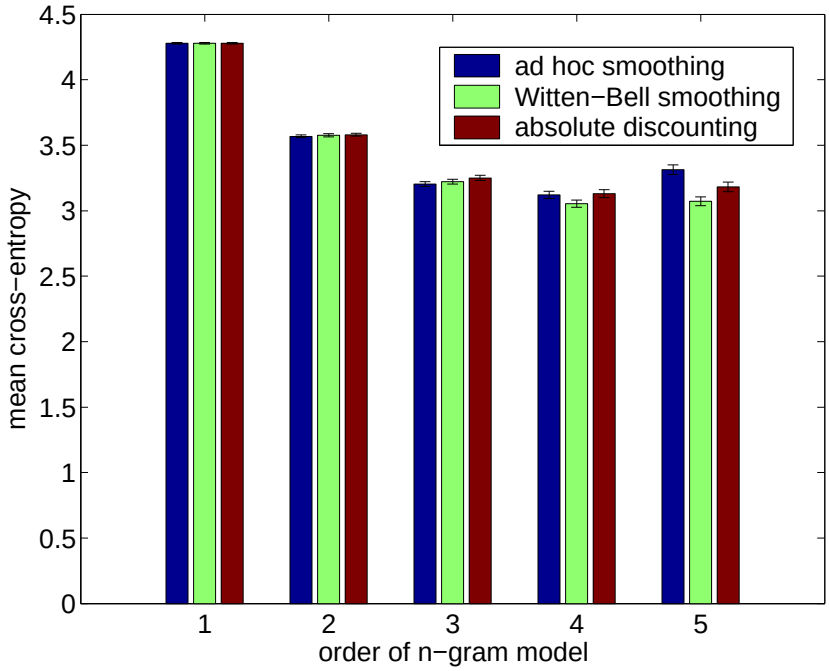

Figure 4: Mean cross-entropy (lower is better) measured over a test set of 1250 puzzles, for different orders of n-grams and different smoothing methods. Error bars indicate 95% confidence intervals.

The search method is controlled by parameters that determine how much computation time

to spend. We investigated the sensitivity of performance to these parameters, as shown in Figure 5. Performance levels off sharply with increasing numbers of swaps per trial (with the number of trials fixed at 15), and there appears to be little benefit in performing more than 10^4 swaps per trial. Increasing the number of trials (with the number of swaps fixed at 10^4) was more fruitful, but the performance beyond 80 trials seems to level off, and results that are nearly as good can be obtained with only 30-40 trials.

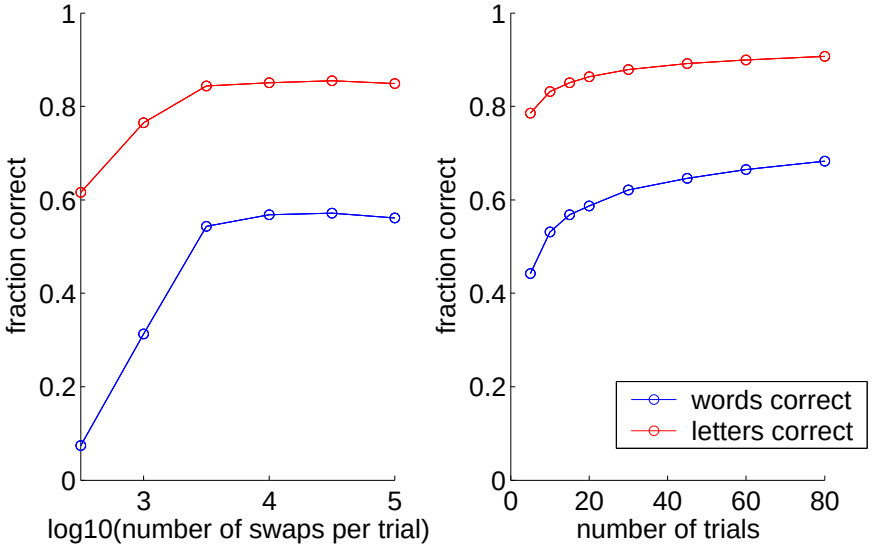

Figure 5: Performance as a function of compute time on a test set of 1250 puzzles ($n = 3$). In the left plot the number of trials is held fixed at 15, and in the right plot the number of swaps is fixed at 10^4.

In Figure 6 we show performance as a function of ciphertext length. Beyond 200 characters of ciphertext, the method performs almost perfectly, and with as few as 75 characters of ciphertext the method is still typically usable. Over a test set of 1250 puzzles, a median of 94% of cipher letters and 65% of cipher words were correctly decoded.

4 Discussion and Future Work

A preliminary investigation was undertaken into modifying the scoring function to incorporate a dictionary of English words. We must be careful not to give too much weight to words appearing in the dictionary, otherwise the search could easily get trapped in bad local minima. To give a specific example, if 85% of the decrypted words appear in the dictionary, the search is probably on the right track, but if only 15% appear in the dictionary, these words could just as easily be nonsense. Moreover, we should probably give less weight to less common words (e.g. TORSEL) as well as shorter words more likely to have been generated at random (e.g. TAM).

Another approach to improving performance is to better exploit our knowledge about the problem. Since the subsitution ciphers we consider are always quotes, the last few words often represent an attribution (which we may want to discount if proper names tend not to appear in our dictionary). Sometimes this structure is even made explicit with the convention of the em-dash ("—") punctuation at the end of the puzzle. For that matter, we may

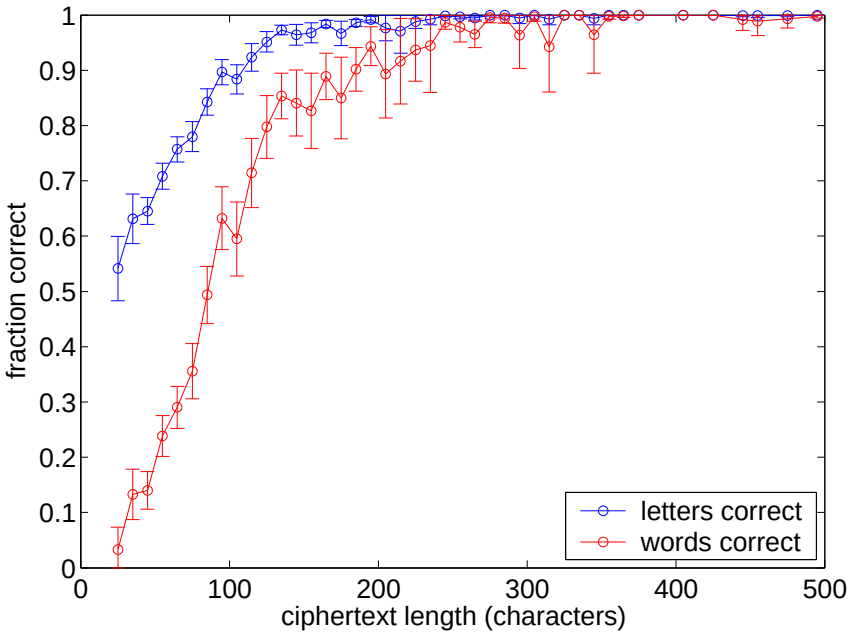

Figure 6: Performance as a function of ciphertext length on a test set of 1250 puzzles ($n = 3$, 15 trials, 10^4 swaps per trial). Error bars indicate 95% confidence intervals.

wish to seach a large reference like *Bartlett's Quotations* to see if any common quotations match the pattern of the ciphertext as a whole (or in part).

Perhaps the most promising unexplored avenue is to more narrowly direct the search by exploiting internal word patterns. For example, the only dictionary words consistent with the pattern ABCADECFCG are CONCERNING, CRUCIBULUM, and AQUASCUTUM. Furthermore, only the first of these words is common enough to appear in our corpus. To give another example, 178 words in the dictionary are consistent with the pattern ABCA, but only 16 of these appear in the corpus. This method entails a large combinatorial search, but many possibilities can be pruned because certain combinations of word choices will be mutually inconsistent.

5 Previous Work

There are perhaps a dozen publications on on the automatic solution of substitution ciphers scattered over the last twenty-five years. Some of the authors working on this problem do not appear to be aware of the work of their predecessors and most only achieve good results on the far easier problem where lots of ciphertext (over 300 characters) is available. Possibly the area has not garnered more interest because substitution ciphers are known to be insecure, and automatic methods take the fun out of solving these for yourself.

Carrol and Martin [2] developed an expert system approach to solving substitution ciphers using hand-coded heuristics, but their results were marginal. Forsyth and Safavi-Naini recast the problem as combinatorial optimization and tackled it with a simulated annealing algorithm [6]. The automated solution of very long texts (5000 characters) is straightforward with their method, but performance is much worse with less ciphertext.

Perhaps closest to our method, Spillman *et al.* used character bigram and unigram statistics to evaluate the goodness of a key, and developed a genetic algorithm approach analogous to our stochastic local search method [15]. Unfortunately, they only present results in terms of their goodness function and do not describe any characteristics of their test set.

Another pair of papers use character trigram statistics and a relaxation scheme to iterate towards the most probable key [14, 1]. Both methods appear to require at least 200 characters of ciphertext for good results.

Lucks based his approach on searching over a word pattern dictionary with the constraint that all ciphertext characters must decrypt to the same plaintext character [11]. Hart later improved on this method by directing this combinatorial search towards more frequent English words [7]. Neither of these methods typically give complete (or unique) solutions because some words may not appear in the dictionary and others may decrypt to multiple possibilities. However, both methods do perform well at decoding small amounts of ciphertext and giving a human cryptographer a good foothold to complete the task. Given a word pattern solution as a starting point, an automated system like QUIPSTER would likely also be able to fill in any remaining gaps.

References

[1] D. Bahler and J. King. An implementation of probabilistic relaxation in the cryptanalysis of simple substitution systems. *Cryptologia*, 16(3):219–225, 1992.

[2] J. Carrol and S. Martin. The automated cryptanalysis of substitution ciphers. *Cryptologia*, 10(4):193–209, 1986.

[3] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, 1996.

[4] C. Dickens. *Great Expectations*. Project Gutenberg Release #1400 (July 1998), 1860–1861. http://www.gutenberg.org.

[5] A. C. Doyle. The adventure of the dancing men. *The Strand Magazine*, 26, December 1903.

[6] W. S. Forsyth and R. Safavi-Naini. Automated cryptanalysis of substitution ciphers. *Cryptologia*, 17(4):407–418, October 1993.

[7] G. W. Hart. To decode short cryptograms. *Communications of the ACM*, 37(9):102–108, September 1994.

[8] H. H. Hoos. *Stochastic Local Search - Methods, Models, Applications*. PhD thesis, Technische Universität Darmstadt, 1998.

[9] F. Jawdat. Personal collection of quotes, 1991–2002. http://www.faisal.com/quotes.

[10] D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, 2000.

[11] M. Lucks. A constraint satisfaction algorithm for the automated decryption of simple substitution ciphers. In *CRYPTO 1988*, pages 132–144, 1988.

[12] A. J. Menezes, P. C. van Oorshot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. http://www.cacr.math.uwaterloo.ca/hac.

[13] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences on stochastic language modelling. *Computer Speech & Language*, 8(1):1–38, 1994.

[14] S. Peleg and A. Rosenfeld. Breaking substitution ciphers using a relaxation algorithm. *Communications of the ACM*, 22(11):598–605, November 1979.

[15] R. Spillman, M. Janssen, B. Nelson, and M. Kepner. Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. *Cryptologia*, 17(1):31–44, January 1993.

[16] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events on adaptive text compression. *IEEE Transaction of Information Theory*, 4(37):1085–1094, 1991.