Cracking General Substitution Ciphers with an n-gram based isEnglish Heuristic

William Zhang[1]

[1]Montgomery Blair High School

October 8, 2025

Abstract

We define an isEnglish heuristic–which scores if a given text is likely to be english–using n-gram character frequency analysis from Google Books. We evaluate this heuristic against others in brute-force-attackable ciphers to validate its capability of differentiating English plaintext from English ciphertext. We then suggest multiple ways of implementing this heuristic to solve a general substitution cipher.

1 Introduction

It's often that perfect ciphers do not exist. Claude Shannon proved that one-time pads are a way to achieve perfect secrecy. To date, this is the only known perfectly secret cipher. However, it's often impractical to exchange unique keys larger than the size of the communication prior to any communication, and have to change them between communications. [Sha49] Shannon showed that other ciphers could be broken based on the entropy of the key and plaintext. [Sha48]

When the number of possible keys is small, the simplest way to solve a cipher is to brute force all possible plaintexts given a specific ciphertext. With a shift cipher, there are only 26 possible inputs to check, and a human can easily identify which text is likely plaintext English. To make this problem more difficult, we can send the ciphertext without any spacing information, to make cracking more difficult. This makes using wordlists less trivial. The subset of decryption problems that do not rely on spacing information is the problem we are addressing in this paper.

Substitution ciphers have been important throughout history. Going all the way back to Caesar, WWII's Enigma, to modern ciphers like AES, which implement many substitution ciphers at different points within their encryption processes. Most of these methods aren't vulnerable to the methods that are used here, which depend on the English language being the input to the substitution. The Enigma was solved with a similar method, hill climbing. It forms the basis for many local search methods in cryptography today. [LKW19] At a fundamental level, if we can thoroughly and efficiently break substitution ciphers, it doesn't break existing ciphers, but can cause implications throughout future cipher development.

The goal is to solve a general monoalphabetic substitution cipher automatically. This is difficiut because the key space is 26!, which is impossible to search in a reasonable amount of time.

Frequency analysis can solve many simple ciphers. If we know that E is the most common letter in the English alphabet, we can quickly figure out a likely candidate letter for what E encrypts to in the ciphertext. Our estimations only get better as text length increases, as the statistical averages converge. Shannon's reference to redundancy and entropy explains why this difference exists at all. We extend this concept of frequency analysis to n-gram frequency distributions, which count the frequency of continuous sequences of n letters in a given text.

Typically, automated frequency analysis implements a scoring algorithm, combined with a search algorithm, to find the key. Frequency analysis is simply only using the 1-gram, but we can do better. Past research has shown that using n higher than 1 can be more effective in solving the general substitution cipher. Wertheimer found that 2-gram analysis combined with an English dictionary is more effective than using 1-grams. Using a higher n in n-gram resulted in a higher difference between scores in a correct decryption versus a transposition of the correct decryption, which we replicate. Since we are using (1–9)-grams, we essentially have the same benefit as an English dictionary but with the added benefit of numerical frequency data. [Wer21]

Genetic algorithms have also been shown to work for solving simple substitution ciphers. Using digrams and monograms, Spillman, et al. showed that there was a high degree of success when using a genetic mating algorithm with fitness being scored as absolute differences with an exponential amplification. [SJNK93]

Prior results have shown a decent level of accuracy in automated substitution cipher analysis. However, these techniques relied on word breaks and punctuation to generate a "mask" for each unique pattern of letter co-occurrences. This method is computationally intensive and doesn't produce great results. [RAT93]

Transposition, a different problem, has been shown to be effectively solved with heuristics in combination with ant colony optimization. They use both bigrams and an English dictionary as heuristics, and build a key using an ant colony system. Since their problem boils down to a graph, essentially, it can be very efficient to search in this way. [RCS03] Similar to our search space, it grows factorially. Additionally, if we break down the substitution cipher into a problem of local search, we are also on a graph.

We would like to know if a given text is English or not. Instead of just using 1-grams or 2-grams, we go further to n-grams as our heuristic of choice. Given an n-gram character frequency analysis of the English language, we propose and evaluate multiple ways for its use on simpler ciphers, namely, the shift and affine ciphers. Given a Euclidean normalized vector of n-gram character frequencies, we test three scores: (1) sum of absolute differences, (2) mean squared error, and (3) cosine similarity. We evaluate each of these heuristics on correctness (i.e., identifying the correct plaintext), and differentiating capability (i.e., the ratio between the correctly identified plaintext and incorrect plaintexts). Lastly, to apply this to the general substitution cipher, we examine probabilistic techniques using the n-gram heuristic to generate a decryption dictionary. We evaluate the length of input text and its relation to performance.

A solution or an improvement in this space doesn't necessarily mean finding a singular method which is better. As previous methods have shown, sometimes a hybrid method is optimal to find the tradeoff between exploration and exploitation.

2 Data

We use Norvig's dataset of English letter frequencies. [Nor13] This dataset was assembled from the Google Books Ngrams dataset of word frequencies. The data was then cleaned and character-level n-gram analysis for n = 1–9. An example of the most frequent n-grams can be seen in Figure 1.

Of course, not *all* (1–9)-gram combinations of characters are found in actual English text.

n	**n-gram**	**Frequency**
1	E	445 155 370 175
2	TH	100 272 945 963
3	THE	69 221 160 871
4	TION	16 665 142 795
5	ATION	8 733 604 406
6	ATIONS	1 788 439 780
7	PRESENT	824 742 877
8	DIFFEREN	670 872 632
9	DIFFERENT	462 529 606

Figure 1: Examples of the most frequent character n-grams for $n = 1$–9.

We compare the actual character n-grams we see to the theoretical length (26^n). Figure 2 lists the lengths of each file.

n	**Length**	**Theoretical Length**	**% Covered**
1	26	26	100.0000%
2	669	676	98.9645%
3	8,653	17 576	49.2319%
4	42,171	456 976	9.2283%
5	93,713	1.19×10^7	0.7887%
6	114,565	3.09×10^8	0.0371%
7	104,610	8.03×10^9	0.0013%
8	82,347	2.09×10^{11}	0.0000%
9	59,030	5.43×10^{12}	0.0000%

Figure 2: Lengths of character n-grams for $n = 1$–9.

For test plaintexts, we choose English language texts of a variety of lengths. A list of texts used can be seen in Figure 3. We clean the data by removing all non-alphabetical characters, whitespace, and converting all characters into lowercase.

Text	Length (chars)
Cryptography*	2.6K
Decl. of Independence	6.6K
Bee Movie Script	37K
Romeo and Juliet	110K
A Tale of Two Cities	580K
Moby-Dick	960K

Figure 3: English test plaintexts used.
*Sourced from the Wikipedia Introduction

Additionally, we sourced a set of Wikipedia articles from English-Corpora and cleaned the data in the same manner. In this case, some HTML tag remnants are left behind, but this is acceptable because it is English text. The summary statistics for this set can be seen in Figure 4

Statistic	Value
N	4,396 articles
Min	26 chars
Q1	279 chars
Median	818 chars
Q3	2,100 chars
Max	87,677 chars

Figure 4: Summary statistics for Wikipedia texts.

Lastly, as another baseline measure, we sourced Google's 10,000 most frequently used words in English from Google's Trillion Word Corpus.

3 Shift Cipher

3.1 Methods

Given each input plaintext from the Wikipedia dataset, we shift it by a random amount. Then, we pass this ciphertext into one of two 'decryption' methods.

The first method is counting the total number of occurrences of each of the top 40 most frequently used words according to Google's 10,000 word list. The highest score is chosen.

The second method is to count each of the (1–9)-grams present in the text, then to compute the cosine similarity to the (1–9)-grams sourced from Norvig. We then sum each of the cosine similarities and pick the highest score.

Each of these methods are then evaluated on their accuracy rate and their differentiating capability. If the metric is a higher-is-better metric, we compute its differentiating capability as the highest value divided by the second highest value. If the metric is a lower-is-better metric, we compute its differentiating capability as the second lowest value divided by the lowest value. This way, all ratios are greater than one and comparable.

Finally, we plot this differentiating capability versus the length of the text.

Additionally, we checked the value of using higher n in the n-grams. We used a composite score of the sum of cosine similarities for n up to a certain amount. We used the first $n = 100$ Wikipedia articles and tested them each on using only 1-grams, using 1-grams and 2-grams, all the way up to including (1–9)-grams.

3.2 Results

For correct and incorrect results, we measure both the mean differentiating capability (M. Dif. Cap.) and the mean length (M. Len.). See Figure 5.

Method	Common Words	N-gram
Correct	4,384	4,394
M. Dif. Cap.	2.091	2.630
M. Len.	1978.1	1972.5
Incorrect	12	2
M. Dif. Cap.	1.074	1.059
M. Len.	65.7	2768.5

Figure 5: Results for shift ciphers.

Note that there were 2 incorrect results for the N-gram method. One text was length 96 and the other was length 5,441, resulting in a high mean length. Length 96 makes sense as it's hard to determine whether a short piece of text is correctly decrypted. The text with length 5441 was a highly irregular Wikipedia article list of United States Navy Landing Craft Infantry, with lots of repeats of the text "USS LCI(L)-(#)" where # is a number.

We record the distribution of correct score ratios in Figure 6. We observe that the distribution for the N-gram scores is consistently higher than the distribution for the Common Words scores. Since the N-gram method both has less incorrect results and provides a higher differentiating capability for correct answers, we choose the N-gram analysis for further study in the Affine method.

For the incremental addition of higher n, all composite scoring methods achieved 100%

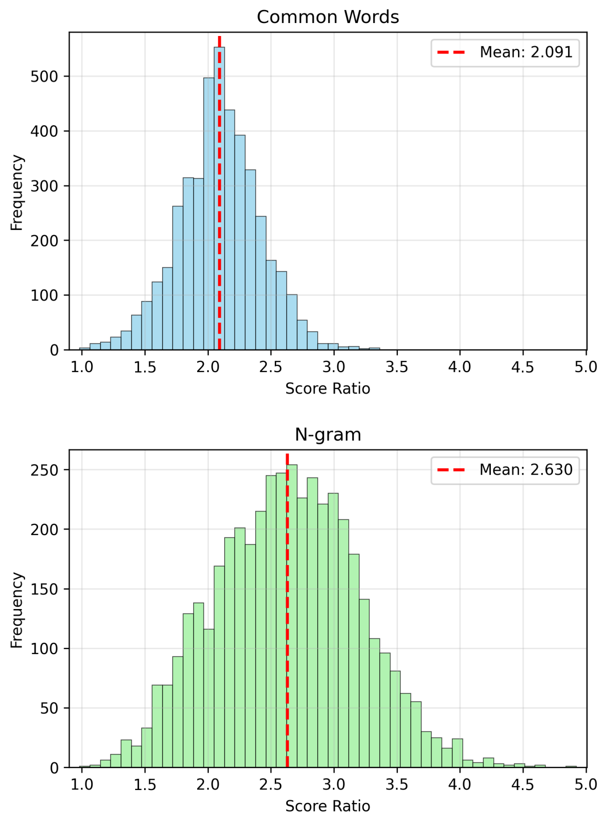

Figure 6: Distributions of correct shift cipher differentiating capability scores.

N	M. Dif. Cap
1–1	1.413
1–2	1.895
1–3	2.400
1–4	2.666
1–5	2.837
1–6	2.940
1–7	3.013
1–8	3.067
1–9	3.104

Figure 7: Incremental scoring for shift ciphers.

correctness. However, we can still measure their differentiating capability in Figure 7. We can see that adding higher n increases the differentiation, but it levels off towards higher n. Overall, the n-gram heuristic is measurably better than the common words heuristic, and only improves as we increase n.

4 Affine Cipher

4.1 Methods

Since Affine ciphers are much more complex (312 vs. 26 possible ciphers), it is more computationally expensive to use the Wikipedia dataset. Thus, we use the first 5 of the 6 test plaintexts as a stand-in. We randomly choose a combination (a, b) such that the encrypted text is $E(x) = (ax + b) \bmod n$. We do not choose $a = 1$ as a possibility, because that would just be a shift cipher. We independently run each plaintext 10 times per method ($N = 10 \times 5 \times 3 = 150$).

We explore different scoring. First, we use the cosine similarity of the vectors produced by n-gram analysis. Second, we use the absolute differences of each component of the n-gram vectors. Third, we use the squared differences of each component of the n-gram vectors. We separately evaluate the differentiating capability for each of the (1–9)-grams.

4.2 Results

We display the accuracy (for $N = 10$ per text-method combination) and the mean differentiating capability (M. Dif. Cap.) for each method in Figure 8.

Looking into the results more closely, no matter what we encrypt the specific affine text with, we will *always* get back the same text with the highest score (the plaintext), but we will also get back the same second-best and third-best text. Even though each of the 10 runs per plaintext have different keys, and therefore ciphertexts, the set plaintexts we get back each time are identical.

Note that the differentiating capability increases significantly as we use higher n for n-gram analysis for cosine similarity, while they decrease with difference-based metrics. As we increase n, we get larger and larger word fragments that can only be a part of actual English. That means for incorrect decryptions, we may get almost-zero scores, but for difference-based methods, there is less penalty.

This means that increasing n in n-grams only helps the differentiation when we are using the cosine similarity method. This analy-

Method	Accuracy	M. Dif. Cap.
Cosine Similarity		
Composite	100.0%	2.99
1	100.0%	1.27
2	100.0%	2.09
3	100.0%	6.95
4	100.0%	21.39
5	100.0%	71.64
6	100.0%	281
7	100.0%	2448
8	100.0%	35 947
9	100.0%	174 445
Absolute Differences		
Composite	100.0%	1.13
1	100.0%	5.41
2	100.0%	3.02
3	100.0%	1.93
4	100.0%	1.43
5	100.0%	1.20
6	100.0%	1.11
7	100.0%	1.07
8	100.0%	1.04
9	100.0%	1.03
Squared Differences		
Composite	100.0%	1.52
1	100.0%	27.98
2	100.0%	7.63
3	100.0%	3.76
4	100.0%	1.90
5	100.0%	1.41
6	100.0%	1.23
7	100.0%	1.16
8	100.0%	1.11
9	100.0%	1.08

Figure 8: Results for affine ciphers.

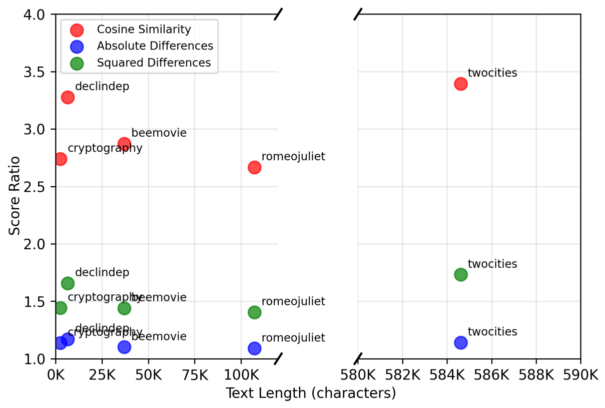

Figure 9: Score ratios of affine scoring methods vs. text length.

sis without combining the scores shows the differing value of different n-gram lengths. Since all methods produce different levels of differentiation at different lengths, we conclude that using n-grams of more than one length are useful.

Additionally, we evaluate the effect of text length and type on encryption success for each of the three methods. Using the composite score, we plot score ratio vs. text length in Figure 9. We additionally find no statistically significant correlation between text length and score ratio ($p \gg 0.05$).

5 General Substitution Cipher

5.1 Methods

We randomly encrypt a general substitution cipher by creating a random permutation of 26 letters. Like previous methods, we strip information about whitespace and punctuation.

As there are 26! possible permutations of the alphabet, and thus, 26! possible keys, it's not feasible to bruteforce every key.

Our heuristics allow us to measure the 'goodness' of any particular 'decryption' text. Thus, we randomly swap characters of the decryption dictionary according to a predetermined strategy. We propose four different strategies.

First, we use beam search with a beam width of 10, maintaining a candidate search space of 10 at a given time. Using a frequency-ordered list of the characters, we iteratively try each key. Each of the 10 keys is tacked on with one of the remaining letters, then, the 10 best keys are selected to move on to the next round. At each round, the partial keys are scored with the log probability of the fully mapped n-grams. Since not every character is mapped at each round, we only score based on contiguous groups of fully mapped n-grams.

The other three methods use local search. We start off with a random permutation and iteratively improve on each of them one swap at a time. We still score using the log probability with all (1–9)-grams.

Hill climbing will only accept a random change if it improves the score.

Simulated annealing starts with an initial temperature and cools over time, always accepting improving changes and randomly accepting worse changes according to current temperature. We initially used parameters $T_0 = 10$ and $\alpha = 0.9995$. The probability we accept a worse change is $P = e^{-\Delta score/T}$, and temperature is updated according to $T_n = T_{n-1} * \alpha$

Monte Carlo Markov Chains (MCMC) is similar to simulated annealing, but it does not depend on a temperature. A worse change is accepted with probability $P = e^{-\Delta score}$.

We precompute the log probability of different n-grams in English using Norvig. Then, with sliding windows of size $n = 1$–9, we add the log probability $\log count/total$ of that specific n-gram to the total for the text. If an n-gram isn't found, we assign it a low floor value of $\log 0.01/total$ where total represents sum of all frequencies of n-grams of that particular size (thus, 0.01 is the 'count' for 0). Thus, we gain the benefit of all n-grams. We chose this method over cosine similarity, as it is much less computationally expensive compared to the vector-based methods. Even though cosine similarity is very powerful, initial tests showed that log probability is as good a heuristic as cosine similarity (mean differentiating capability in between cosine similarity and squared differences).

We used a standard number of iterations as n=10 000 for each of the three local search methods. We present both average execution time, and also an adjusted execution time. This execution time is adjusted by the last iteration where there was an improvement in score $time_{adj} = time * iters/10000$

We measure success of a given 'decryption' by comparing the original plaintext to the decrypted plaintext. Comparing decryption keys is insufficient, as it's possible for multiple characters to not be used in the original plaintext, making it impossible for there to be a single correct answer.

We ran these on the first $n = 1000$ Wikipedia article texts.

Based on these intial results, we then modified the protocol by testing a variety of parameters. Instead of initially setting the decryption dictionary with just a random mapping, we start local search using the result from beam search. We then pass it into MCMC, as it explored the largest search space. We did collected data on the first $n = 100$ articles from the Wikipedia dataset, and individually analyze inaccuracies.

We also collected data on checking if increasing using higher n for the n-gram scoring was valuable on the same $n = 100$ dataset. We evaluated both the accuracy and the speed of convergence, measured in mean iterations before convergence on the final answer. We choose parameters for the beam search + MCMC method based on the results from the previous mini-study.

5.2 Results

Hill climbing, MCMC, and Simulated Annealing deliver broadly similar results with similar runtimes, as seen in Figures 11 and 12. Beam search stands out as the most accurate method and overall fastest method, when not adjusting times.

Accuracy improves with character length, as is visible in Figures 10 and 11. Additionally, in Figure 10, we see that after using all four methods, only 19.5% of results were not correctly decrypted by any of them.

Based on the belief that MCMC explored the largest search space, we further used MCMC with a starting dictionary from beam search. We tested beam widths of 10 or 20, and 2000, 5000, or 10000 iterations, with results shown in Figure 13. We are able to get significantly better performance with less runtime than local search methods with the 10/2000 combination, and are able to achieve even higher performance with 20/10000.

# Methods Correct	**Count**	**Average Length**
0	195	736.8
1	123	1533.3
2	184	2853.9
3	289	4648.3
4	209	7032.8
Total	1000	3670.6

Figure 10: Overall correctness results for general substitution ciphers.

Method	**Accuracy (>1k chars)**
Hill Climb	53.8% (67.6%)
MCMC	51.8% (67.0%)
Sim. Anneal.	55.4% (70.0%)
Beam Search	58.4% (71.9%)
n	1000 (676)

Figure 11: Individual correctness results for general substitution ciphers.

Method	**Time (Adjusted)**
Hill Climb	305.2 (57.2)
MCMC	303.4 (59.3)
Sim. Anneal.	303.8 (57.2)
Beam Search	164.9

Figure 12: Average time (in seconds) per article.

Of the 12 articles that no method got correct, 10/12 had a higher log probability score of the plaintext than the proposed plaintext. This indicates that we would be able to significantly improve accuracy without changing our heuristic, we just need to be able to search a larger space.

Beam Width	**Iterations**	**Accuracy**
10	2000	80%
10	5000	83%
10	10000	83%
20	2000	84%
20	5000	85%
20	10000	85%

Figure 13: Beam search + MCMC results.

For the incremental study, we choose the beam width 10 and 2000 iterations. This has an acceptable level of performance, while being quick to run. Additionally, since the performance is not clustered towards the top, hopefully, it'll allow us to see a larger differetiation between the incremental n additions.

We display the results of our incremental scoring in Figure 14. Note that for the first row, simple frequency analysis never got it right, which is why its mean iterations towards convergence was zero: the beam search just creates the frequency key, and the MCMC isn't able to improve the score. There is an increase in accuracy up until around 6. Additionally, there is a decrease in iterations until convergence up until around 5. There is a clear increase in accuracy as we add higher n, but there is decreasing marginal value in adding n higher than 6 for balancing accuracy and compute time.

N	**Accuracy**	**Mean Iterations**
1–1	0%	0
1–2	47%	472.6
1–3	54%	464.2
1–4	68%	469.7
1–5	76%	403.6
1–6	80%	407.1
1–7	79%	396.1
1–8	81%	407.6
1–9	80%	413.7

Figure 14: Incremental scoring for general substitution ciphers.

6 Conclusion

We show that initial results are promising for the use of n-gram analysis as a heuristic for solving the general substitution cipher.

Accuracy of cryptanalysis methods is still highly correlated with text length. Our hybrid approach, seeding MCMC with beam search produced the highest success rate of 85%.

We believe that the main improvement we can make in future iterations is to find ways to efficiently expand the search space to produce better results.

References

[LKW19] George Lasry, Nils Kopal, and Arno Wacker. Cryptanalysis of enigma double indicators with hill climbing. *Cryptologia*, 43(4):267–292, July 2019.

https://doi.org/10.1080/01611194.2018.1551253.

[Nor13] Peter Norvig. English letter frequency counts: Mayzner revisited or etaoin srhldcu, 2013. https://norvig.com/mayzner.html.

[RAT93] R S Ramesh, G Athithan, and K Thiruvengadam. An automated approach to solve simple substitution ciphers. *Cryptologia*, 17(2):202–218, April 1993. https://doi.org/10.1080/0161-119391867872.

[RCS03] M.D. Russell, J.A. Clark, and S. Stepney. Making the most of two heuristics: breaking transposition ciphers with ants. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 4, pages 2653–2658 Vol.4, 2003. https://doi.org/10.1109/CEC.2003.1299423.

[Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[Sha49] C E Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, October 1949. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x.

[SJNK93] Richard Spillman, Mark Janssen, Bob Nelson, and Martin Kepner. Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. *Cryptologia*, 17(1):31–44, January 1993.

[Wer21] Phil Wertheimer. On the difficulty of breaking substitution ciphers. *Digital Repository at the University of Maryland*, 2021. https://doi.org/10.13016/zlm4-mkns.