Exploring Classical Cryptographic Ciphers and IS-ENGLISH Detection Techniques

Harshitha Poludas

Mentored by Professor William Gasarch

Department of Computer Science, University of Maryland

July 4, 2025

Abstract

The following is an important problem in cryptography: Alice wants to send a message to Bob so that (a) Bob can understand it, and (b) if Eve intercepts it, she cannot. This project explores several classical encryption schemes including the Shift Cipher, Affine Cipher, and other related methods. I implemented encryption, decryption, and cracking programs, with a special focus on English detection via frequency analysis and heuristic methods. This paper documents the findings, implementations, and future directions in cipher analysis and automated language detection.

A secondary topic of this project was testing whether the same `IS-ENGLISH()` detectors could also reject foreign-language texts. While fascinating, this direction is somewhat distinct from cipher cracking, and we will comment on it briefly in the paper and more extensively in the appendix.

1 Introduction

In classical cryptography, the goal is to securely transmit information even if an adversary intercepts it. My research focused on understanding and implementing cipher systems where the encryption is reversible, and cracking is possible using frequency analysis and linguistic patterns.

I studied how messages can be:

- Encrypted using known ciphers
- Decrypted when the key is known
- Cracked when the key is unknown

A central component of the project involved developing an `IS-ENGLISH` detector to determine whether a given decoded message is likely to be English, acting as a confirmation that the decoded text is correct. I spent several weeks designing, programming, and refining various detection methods, evaluating their performance on both long and short texts. The most striking finding was that the weighted common words method succeeded on short inputs (10–15 characters) where all other techniques failed, while on medium and long texts (200+ and 3000+ characters) most methods performed comparably well.

In addition to cipher analysis, I also explored a separate but related research question: how well do these detectors handle **foreign-language texts**? This is addressed in Appendix A, since it diverges from the main thread of cryptanalysis but provides valuable insight into the strengths and limitations of language-based detection methods.

Research Question

The central research question guiding this project is:

> How effective are different `IS-ENGLISH` detection methods at correctly identifying English plaintext when applied to the decryption and cracking of classical ciphers, and how does their performance vary across cipher types, text lengths, and input conditions?

2 Classical Ciphers

This project focused on two fundamental classical encryption schemes: the Shift Cipher and the Affine Cipher. Both are substitution ciphers that operate over the 26-letter English alphabet and serve as useful models for understanding key-based encryption and decryption processes.

2.1 Shift Cipher

The Shift Cipher (also known as the Caesar Cipher) shifts each letter in the plaintext by a fixed amount s in the alphabet. It is one of the oldest known encryption techniques and serves as a simple example of a monoalphabetic substitution cipher.

Encryption and Decryption

Let x be the numerical value of a character ($0 \leq x < 26$), and let s be the shift. Then:

$$\text{Encryption:} \quad E(x) = (x + s) \mod 26$$

$$\text{Decryption:} \quad D(y) = (y - s) \mod 26$$

Non-alphabetic characters are preserved during transformation.

Implementation

The encryption and decryption functions were implemented in Python, using modular arithmetic to wrap around the alphabet. A shift of $s = 5$, for example, maps “A” to “F”, “B” to “G”, and so on.

Cracking

To crack the cipher without knowing s, I tried all 26 possible shift values and evaluated each decryption candidate using my `IS-ENGLISH` detector. The correct decryption is identified as the first candidate that is likely to be English.

2.2 Affine Cipher

The Affine Cipher generalizes the Shift Cipher by introducing two parameters: a multiplier a and an offset b. Encryption and decryption are based on linear modular transformations.

Mathematical Formulation

For $x \in \{0, 1, \ldots, 25\}$:

$$\text{Encryption:} \quad E(x) = (ax + b) \mod 26$$

$$\text{Decryption:} \quad D(y) = a^{-1}(y - b) \mod 26$$

Here, a^{-1} is the modular inverse of a modulo 26. For the decryption to work, a must be relatively prime to 26. Valid values of a are 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, and 25. These are the numbers less than 26 that do not share any common factors with 26 other than 1. Since the prime factorization of 26 is 2×13, any number not divisible by 2 or 13 will be relatively prime to 26.

Implementation

In my Python implementation, the program:

- Validates that a is invertible modulo 26
- Encrypts text using the affine formula
- Finds a^{-1} during decryption using a brute-force search

Cracking

To crack an affine-encrypted message without knowing the keys, the program tries all $12 \times 26 = 312$ valid (a, b) key pairs. Each decryption attempt is passed through several `IS-ENGLISH` detectors which will be discussed further in the next section. If the result is classified as English, the correct keys and plaintext are returned.

Results

Both ciphers were successfully implemented with full encryption, decryption, and cracking capabilities. For the Shift Cipher, testing all 26 possible shifts was efficient and reliably yielded the correct plaintext. The Affine Cipher required iterating over 312 possible (a, b) pairs, but the implementation handled this search effectively.

To determine whether a decryption was valid, each candidate plaintext was evaluated using the `IS-ENGLISH` detector. Initially, simpler methods such as the dot product and absolute frequency difference were used. As part of the internship research, I later developed a weighted common words detector. While this method performed comparably to others on medium and long texts, its unique strength was in handling very short inputs (10–15 characters), where all other approaches failed. This made it particularly valuable in scenarios where only small ciphertext fragments are available.

The performance and comparative effectiveness of all `IS-ENGLISH` methods are discussed in detail in the following section.

3 IS-ENGLISH Approaches

A key part of this project was building an effective `IS-ENGLISH` detector: an algorithm that could evaluate whether a decoded message is likely to be English. This was essential for cracking ciphers without knowing the key: if the program could identify when a decoded message "looked" like English, it could determine when decryption had succeeded. I developed and tested several methods of English detection throughout the internship, each with different strengths and weaknesses depending on the type of input.

3.1 Implemented Techniques

The following four techniques were implemented and tested:

1. Dot Product of Frequency Vectors
2. Absolute Difference of Frequency Vectors
3. Common Letter Heuristic
4. Common Words

3.2 Letter Frequency and the English Language

The English language follows statistical patterns. For example, the letter `E` appears much more often than the letter `Z`. These patterns remain relatively stable across most English texts.

Standard English Letter Frequencies

The following graph shows the standard frequency of each letter in the English language, based on large-scale text analysis:

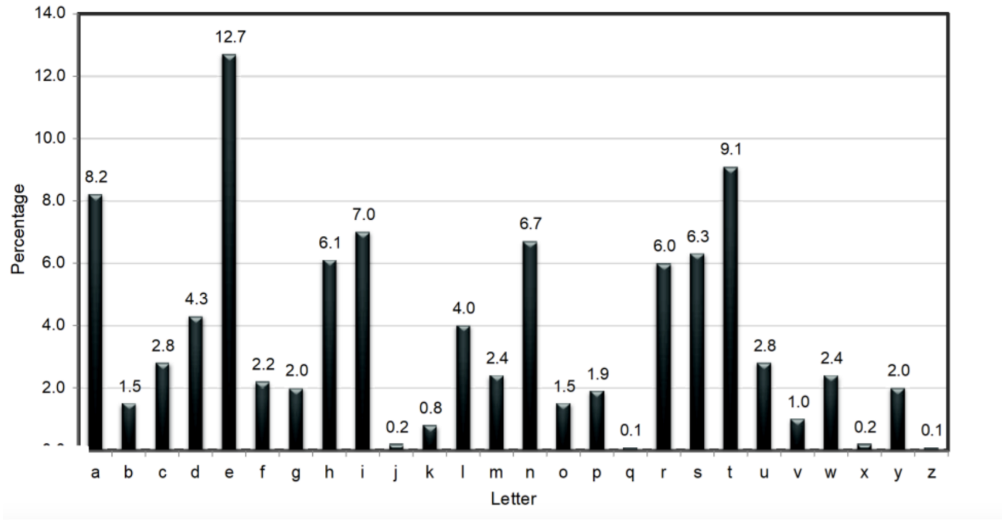

Figure 1: Standard English letter frequencies (source: Katz, Introduction to Modern Cryptography slides, 2023

Frequency Vectors and Measuring Proximity

Let T be a long text of length N, possibly encrypted.
Let N_a be the number of occurrences of the letter `a` in T,
Let N_b the number of occurrences of the letter `b` in T, and so on.

The **frequency vector** $\vec{f_T}$ of the text is defined as:

$$\vec{f_T} = \left(\frac{N_a}{N}, \frac{N_b}{N}, \frac{N_c}{N}, \ldots, \frac{N_z}{N} \right)$$

Each component represents the proportion of that letter in the text.

I also define $\vec{f_E}$ to be the standard English frequency vector, derived from known statistics of English writing. This vector is assumed to represent "ideal" English text.

To determine whether T is likely to be English, we compare $\vec{f_T}$ and $\vec{f_E}$. If they are close, then the text likely resembles English. But how do we measure how close two vectors are?

Two methods are used in this project:

- The **dot product**, which is high when vectors are similar
- The **absolute difference**, which is low when vectors are similar

These two approaches are discussed in the following subsections.

3.3 Dot Product of Frequency Vectors

This method compares the letter frequency distribution of the input text with that of standard English using a dot product:

$$\text{score} = \sum_{i=0}^{25} f_{\text{text}}[i] \cdot f_{\text{English}}[i]$$

Here, f_{text} is the normalized frequency vector of the input text, and f_{English} is the reference frequency vector for English. The closer the vectors are aligned, the higher the dot product score. If the score exceeds a threshold (set to 0.06 in my implementation), the text is classified as English.

This method works best on longer texts where the letter frequency distribution has had enough room to "settle" and reflect real English statistics. It performs poorly on short or irregularly formatted inputs where frequency noise dominates.

Results and Evaluation

To evaluate this method, I divided test cases into three categories based on length:

- **Short texts** (10–15 characters): Examples include `Hello World`, `My name is Bob`, and `Hi how are you`. The method failed to detect English in 5 out of 6 short English texts, often returning dot products below the 0.06 threshold. This confirms the method's weakness on very short inputs.
- **Medium-length texts** (200+ characters): This included a poem, *Oh, the Places You'll Go*, and a few paragraphs from a Wikipedia article about sharks. The method correctly identified all of these as English, performing reliably in this range.
- **Long texts** (3000+ characters): These included full scripts and long-form articles (e.g., *Barbie Movie*, *Wikipedia article about food*). The dot product method correctly labeled every long English text. It also worked well on a research paper about IQ and sleep.

Failure Cases

Despite its success on medium and long English texts, this method showed a tendency to misclassify some foreign-language inputs as English. For example, Spanish and French versions of the *Shrek* script were incorrectly labeled as English, likely because their letter distributions partially overlap with those of English. It also incorrectly accepted lorem ipsum placeholder text (Latin) as English.

This limitation reveals that while the dot product is effective in controlled scenarios, it cannot reliably distinguish between similar-looking alphabets or detect meaningful language structure.

3.4 Absolute Difference of Frequency Vectors

This method calculates how different the input text is from standard English by summing the absolute differences between each letter's frequency:

$$\text{score} = \sum_{i=0}^{25} |f_{\text{text}}[i] - f_{\text{English}}[i]|$$

Here, f_{text} and f_{English} are the normalized letter frequency vectors of the input and English respectively. A lower score means the text's distribution is closer to English.

In my implementation, a text is classified as English if its score is less than or equal to 0.3. This threshold was found to work well through experimentation.

Results and Evaluation

I tested this method across the same three text length categories as before:

- **Short texts** (10–15 characters): Examples like `Hello World`, `Apple Banana Cat`, and `Hi I am going outside` were all incorrectly classified as not English. The reason is that in such short texts, even one or two letter imbalances can throw off the frequency vector significantly, leading to large absolute differences.
- **Medium-length texts** (200+ characters): This group included a poem, Dr. Seuss's *Oh, the Places You'll Go*, and paragraphs from a Wikipedia article about sharks. All were correctly identified as English, showing that the method becomes more reliable as text length increases.
- **Long texts** (3000+ characters): The method correctly identified every long English sample, including full movie scripts and long Wikipedia articles. Because the law of large numbers smooths out noise, the frequency patterns in these texts closely match those of standard English.

Strengths and Failure Cases

Where this method truly excels is in rejecting non-English inputs. Every single gibberish, foreign-language, and nonsense sample (including false-positive gibberish, pseudo-Latin placeholder text, and French or Spanish movie scripts) was correctly identified as not English. This is a major strength that the dot product method could not consistently match.

However, the tradeoff is that it was far too strict on short English phrases. In my tests, it failed all 6 short English inputs, even ones as simple as `My Name is Bob` or `Hi how are you`. This suggests that while the absolute difference method is highly accurate for filtering out bad candidates, it's unreliable for short or casual English messages.

Conclusion

Overall, this method is extremely effective for detecting non-English inputs and performs strongly on medium to long texts. However, its rigidity makes it a poor standalone detector for short messages. In future comparative analysis, it will be valuable to consider combining this method with others that are more lenient on short text.

3.5 Common Letter Heuristic

This approach is based on a very simple idea: certain letters occur much more frequently in English than others. For example, according to Figure 1, the most common letters in English are:

`E, T, A, O, I, N, S, H, R, D, L, U, ...`

Because letters like `E`, `A`, and `T` appear so often, we can use their presence in a text as a rough signal that the text may be English.

This method examines the frequency of letters in the input text and selects the top three most frequent ones. If at least two of those three letters are among the three most common English letters (`E`, `A`, `T`), the text is classified as English.

Results and Evaluation

I tested this heuristic on the same categories of input as the other methods:

- **Short texts** (10–15 characters): Unsurprisingly, the method failed every short English test case. Very basic English sentences like `Hello World`, `My name is Bob`, and `Hi how are you` did not pass the check. This is because, in such small samples, the top three letters may not include many of `E`, `A`, or `T` due to randomness or low counts.
- **Medium-length texts** (200+ characters): This method had partial success here. It correctly identified texts like *Oh, the Places You'll Go* and a shark-related Wikipedia paragraph, but failed on others. While the presence of common letters becomes more stable, the heuristic is still unreliable overall.
- **Long texts** (3000+ characters): At this length, the letter frequency distribution tends to align better with standard English, so the method performed reasonably well. Most long English texts were accepted correctly.

False Positives and Limitations

The biggest issue with this method is that it incorrectly classifies many foreign texts as English. For example, Spanish and French scripts often contain high frequencies of letters like `A` and `E`, which are also common in English. As a result, this method misclassified several foreign-language inputs, such as the *Shrek* script in Spanish and both Spanish test cases, as English.

It also failed to reject non-English gibberish like `Altarbade Fertalrian Gernvdaeh`, because those nonsense words coincidentally contained high-frequency letters like `A` and `E`.

Conclusion

This heuristic is fast and simple but lacks the depth needed for accurate classification. It may offer value as part of a larger ensemble of detectors or for early filtering in large-scale systems. However, by itself, it is not reliable for short texts, foreign text rejection, or precise English detection.

3.6 Common Words

The idea behind this approach was to detect English based on the presence of real English words, rather than relying only on alphabet frequency. The key motivation was that random-looking strings or foreign languages might happen to mimic English letter patterns, but they wouldn't contain recognizable English vocabulary.

In this method, the input text is first cleaned by removing all spaces, punctuation, and numbers, and converting all letters to lowercase. Then, the program extracts all substrings of length 3 to 8 from the cleaned text and compares each one to a large dictionary of common English words.

Let n be the number of substrings generated. The input is classified as English if at least $\sqrt{n}$ of the substrings appear in the dictionary. The intuition is that if a reasonable number of substrings look like English words, even when the original spacing is lost, then the original text likely contains English content.

Results and Evaluation

As with the other methods, I tested the Common Words method across short, medium, and long texts:

- **Short texts** (10–15 characters): The method failed all six English test cases. Because the number of substrings is small, the square root threshold is high relative to what the text can achieve. For instance, the input `Hello World` contains only a handful of substrings, and very few match dictionary entries, resulting in a false negative.
- **Medium-length texts** (200+ characters): This method performed well in this range. Inputs like a poem, Dr. Seuss's *Oh, the Places You'll Go*, and paragraphs from a Wikipedia article were all correctly classified as English.
- **Long texts** (3000+ characters): This method also correctly identified all long English inputs, including movie scripts and scientific writing. Because longer texts generate more substrings, and thus more word matches, they tend to exceed the $\sqrt{n}$ threshold comfortably.

False Positives and Limitations

The biggest weakness of this approach was its inability to distinguish English from foreign languages. It classified nearly all non-English inputs as English (including Spanish, French, and German scripts) because many substrings in those languages overlap with English word fragments. For example, common letter sequences like `ent`, `ion`, or `est` may appear in multiple languages, inflating the match count.

It also misclassified the well-known Latin placeholder *lorem ipsum* text as English for the same reason. These false positives highlight the limitations of using only word shape and substring matching without weighing how likely each word is in real English (an idea that will be revisited in the next version of this method).

Weighted Substrings: A New Approach

To address the issues with false positives and inflexibility on short texts, I developed an improved version of the Common Words method using weighted scoring.

Instead of treating all dictionary words equally, I assigned each word a weight based on its frequency in English. For example, the word `the` was given a full weight of 1.0, while less common words like `downtown` received lower scores (e.g., 0.6778). Additionally, I filtered out all words under three letters or those without vowels, removing noise and abbreviations from the dataset.

The input text is processed similarly; all punctuation, spaces, and digits are removed, and all substrings of length 3 to 8 are extracted. For each substring that matches a word in the weighted dictionary, its corresponding score is added to a total. The final score is computed as the sum of matched weights divided by the total number of substrings:

$$\text{normalized score} = \frac{\sum \text{matched word weights}}{\text{total substrings}}$$

This normalized score is then compared to a threshold. Initially, I experimented with dynamic thresholds based on text length, but through data analysis, I discovered a simpler and more robust solution: a flat threshold of 0.045 worked for all tested input lengths.

Results and Evaluation

The updated weighted method achieved perfect classification accuracy across all test cases:

- **Short texts**: Unlike every other method tested, the weighted common words approach correctly classified all short English texts, including `Hello World`, `Hi how are you`, and `My name is Bob`, which are notoriously difficult due to their limited context.
- **Medium-length texts**: As expected, the method performed flawlessly on this category, continuing to correctly identify texts such as poems, children's literature, and encyclopedia entries.

- **Long texts**: Long English inputs like the *Barbie Movie Script*, Wikipedia articles, and research papers were also correctly labeled as English. Despite the presence of formatting irregularities and varied vocabulary, the weighted approach remained consistent and accurate.

Foreign Text and Gibberish Rejection

Most notably, this method also correctly rejected every non-English input. Foreign-language texts such as French, Spanish, and German, which previously produced many false positives, now scored well below the 0.045 threshold. The lowest-scoring non-English sample was the German test case (score = 0.0235), and the highest was the French *Shrek* script (score = 0.0307), both well below the threshold. Even inputs with no real words, like `zqx jrk blr mnv` and numeric strings, were safely rejected.

Summary

The addition of word frequency weighting, combined with a flat universal threshold, dramatically increased the effectiveness of the Common Words method. It outperformed all previous approaches by accurately identifying all English texts (regardless of length, formatting, or spacing) while simultaneously rejecting all non-English samples. This made it the most robust, generalizable, and precise `IS-ENGLISH()` method developed in this project.

4 Performance of `IS-ENGLISH()` Methods on Ciphers

To evaluate how well the different `IS-ENGLISH()` techniques function in practical cipher-cracking, I tested them on two representative classical ciphers: the Shift Cipher and the Affine Cipher. Each was applied to both a short English passage (about 50 words) and a longer one (about 350 words). This allowed me to assess not only the inherent strengths and weaknesses of each detection method, but also how performance varied with text length and cipher structure.

4.1 Shift Cipher

For the Shift Cipher, results were largely consistent with the theoretical expectations of each method. The dot product method struggled, failing to identify English in both the short and long passages. This can be explained by the fact that frequency vectors derived from short samples exhibit significant noise, and even in longer passages, the method's fixed threshold proved too strict.

By contrast, the absolute difference method performed reliably. Because it directly penalizes deviations in frequency distributions rather than requiring precise alignment, it was more tolerant of the small statistical irregularities that appear in decrypted text, successfully detecting English in both cases.

The common words approach also proved highly effective on the Shift Cipher. Since shifting does not alter word boundaries or internal letter order, once the correct key is applied, English words appear intact, making dictionary-based detection straightforward.

Finally, the common letters heuristic showed mixed reliability: when restricted to the top two letters, it frequently produced false negatives, but with a threshold of three frequent letters it became more dependable. This reflects the fact that letter frequencies stabilize only at a certain text length, and that English's most common letters (`E`, `T`, `A`) provide a stronger signal when considered together.

4.2 Affine Cipher

The Affine Cipher presented greater challenges, since its transformations distort frequency distributions in a non-linear way. Unsurprisingly, the dot product again failed, as the mapping between plaintext and ciphertext letters is less uniform than in the Shift Cipher, further lowering similarity scores. In contrast, the absolute difference method retained its robustness: although frequencies are warped, they remain within tolerable bounds, allowing the method to succeed on both short and long samples.

The common words method again performed consistently well, as the structural integrity of words is preserved once the correct key is applied. However, the common letters heuristic was more fragile under the

Affine Cipher than under the Shift Cipher. Because multiple (a, b) key pairs can coincidentally produce the same set of high-frequency letters, merely checking for partial overlap with English's frequent letters was insufficient. Reliability improved only when requiring all three of the top English letters to appear among the most frequent in the decrypted text, especially for longer passages.

4.3 Comparative Observations

Overall, these experiments highlight important distinctions between the methods. The dot product is too brittle to be of general use, particularly for affine transformations or short texts, though it might still serve as a coarse filter when applied to very long samples. The absolute difference method stands out as a reliable baseline, since it accommodates moderate distortions and remains effective across cipher types. The common words approach emerges as the strongest technique in practice: it is resilient to distortions, sensitive to actual linguistic structure, and consistently identifies valid decryptions across both ciphers and text lengths. By contrast, the common letters heuristic is only situationally useful, offering occasional success when applied with stricter thresholds, but prone to false positives and negatives when used in isolation.

Taken together, these findings suggest that while frequency-based statistical methods offer some utility, dictionary-driven approaches such as common words (particularly in its weighted form) are both more discriminative and more generalizable. This aligns with the intuition that language detection cannot rely on frequency patterns alone, but must incorporate structural and semantic signals to reliably distinguish correct decryptions from spurious ones.

5 Conclusion

This project examined the effectiveness of several `IS-ENGLISH()` detection methods in the context of cracking classical substitution ciphers. Across both the Shift and Affine Ciphers, frequency-based techniques such as the dot product and absolute difference demonstrated that letter statistics can be useful signals, but also revealed their fragility: they struggled with short texts, noisy distributions, and foreign-language false positives. The common letters heuristic offered simplicity and speed, but its reliability was limited to specific scenarios and larger samples.

The weighted common words method proved to be the most robust overall. Its key advantage was on short texts, where every other approach failed but it still reliably detected English. On medium and long texts, it performed on par with the other strong methods (dot product, absolute difference, and common words), but without any loss of accuracy. This ability to succeed on short, noisy inputs while remaining competitive on longer samples makes it the most versatile technique developed in this project. This robustness suggests that incorporating linguistic structure, rather than relying solely on statistical similarity, is essential for accurate English detection in cryptanalysis.

Looking ahead, future work could extend these methods beyond classical ciphers. Applying weighted word detection to more complex encryption schemes, testing against larger and more diverse corpora, or combining statistical and linguistic features into a hybrid ensemble model could provide even stronger performance. These directions reflect the broader challenge of language recognition in cryptography: developing methods that are both sensitive enough to catch short, irregular inputs and discriminative enough to reject plausible but incorrect candidates.

A Foreign Text Experiments

A fascinating secondary topic explored in this project was testing `IS-ENGLISH()` methods on **foreign-language texts**. The motivation was to evaluate whether detectors that perform well on English plaintext also correctly reject inputs in Spanish, French, German, and other languages.

The results highlighted important differences:

- The **dot product** method often misclassified foreign-language texts (especially Spanish and French) as English, since their letter frequency distributions overlap significantly with English.

- The **absolute difference** method excelled here, correctly rejecting all foreign-language samples by detecting even small deviations in letter frequencies.
- The **common words** method misclassified nearly all foreign texts as English, due to substring overlaps with English word fragments (`ent`, `ion`, etc.).
- The **weighted common words** method correctly rejected every foreign-language text, scoring all of them well below the 0.045 threshold.

These experiments confirmed that foreign-text rejection is a qualitatively different problem from cipher cracking, but also demonstrated that weighted common words provide a powerful general-purpose discriminator. While not the main research focus of this project, this secondary investigation reinforced the importance of combining frequency-based and vocabulary-based methods for robust language detection.