Decoding Anagrammed Texts Written in an Unknown Language and Script

Bradley Hauer and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, Canada
{bmhauer,gkondrak}@ualberta.ca

Abstract

Algorithmic decipherment is a prime example of a truly unsupervised problem. The first step in the decipherment process is the identification of the encrypted language. We propose three methods for determining the source language of a document enciphered with a monoalphabetic substitution cipher. The best method achieves 97% accuracy on 380 languages. We then present an approach to decoding anagrammed substitution ciphers, in which the letters within words have been arbitrarily transposed. It obtains the average decryption word accuracy of 93% on a set of 50 ciphertexts in 5 languages. Finally, we report the results on the Voynich manuscript, an unsolved fifteenth century cipher, which suggest Hebrew as the language of the document.

1 Introduction

The Voynich manuscript is a medieval codex[1] consisting of 240 pages written in a unique script, which has been referred to as the world's most important unsolved cipher (Schmeh, 2013). The type of cipher that was used to generate the text is unknown; a number of theories have been proposed, including substitution and transposition ciphers, an *abjad* (a writing system in which vowels are not written), steganography, semi-random schemes, and an elaborate hoax. However, the biggest obstacle to deciphering the manuscript is the lack of knowledge of what language it represents.

Identification of the underlying language has been crucial for the decipherment of ancient scripts, including Egyptian hieroglyphics (Coptic), Linear B (Greek), and Mayan glyphs (Ch'olti'). On the other hand, the languages of many undeciphered scripts, such as Linear A, the Indus script, and the Phaistos Disc, remain unknown (Robinson, 2002). Even the order of characters within text may be in doubt; in Egyptian hieroglyphic inscriptions, for instance, the symbols were sometimes rearranged within a word in order to create a more elegant inscription (Singh, 2011). Another complicating factor is the omission of vowels in some writing systems.

Applications of ciphertext language identification extend beyond secret ciphers and ancient scripts. Nagy et al. (1987) frame optical character recognition as a decipherment task. Knight et al. (2006) note that for some languages, such as Hindi, there exist many different and incompatible encoding schemes for digital storage of text; the task of analyzing such an arbitrary encoding scheme can be viewed as a decipherment of a substitution cipher in an unknown language. Similarly, the unsupervised derivation of transliteration mappings between different writing scripts lends itself to a cipher formulation (Ravi and Knight, 2009).

The Voynich manuscript is written in an unknown script that encodes an unknown language, which is the most challenging type of a decipherment problem (Robinson, 2002, p. 46). Inspired by the mystery of both the Voynich manuscript and the undeciphered ancient scripts, we develop a series of

[1]The manuscript was radiocarbon dated to 1404-1438 AD in the Arizona Accelerator Mass Spectrometry Laboratory (*http://www.arizona.edu/crack-voynich-code*, accessed Nov. 20, 2015).

Transactions of the Association for Computational Linguistics, vol. 4, pp. 75–86, 2016. Action Editor: Regina Barzilay.
Submission batch: 12/2015; Published 4/2016.

algorithms for the purpose of decrypting unknown alphabetic scripts representing unknown languages. We assume that symbols in scripts which contain no more than a few dozen unique characters roughly correspond to phonemes of a language, and model them as monoalphabetic substitution ciphers. We further allow that an unknown transposition scheme could have been applied to the enciphered text, resulting in arbitrary scrambling of letters within words (*anagramming*). Finally, we consider the possibility that the underlying script is an abjad, in which only consonants are explicitly represented.

Our decryption system is composed of three steps. The first task is to identify the language of a ciphertext, by comparing it to samples representing known languages. The second task is to map each symbol of the ciphertext to the corresponding letter in the identified language. The third task is to decode the resulting anagrams into readable text, which may involve the recovery of unwritten vowels.

The paper is structured as follows. We discuss related work in Section 2. In Section 3, we propose three methods for the source language identification of texts enciphered with a monoalphabetic substitution cipher. In Section 4, we present and evaluate our approach to the decryption of texts composed of enciphered anagrams. In Section 5, we apply our new techniques to the Voynich manuscript. Section 6 concludes the paper.

2 Related Work

In this section, we review particularly relevant prior work on the Voynich manuscript, and on algorithmic decipherment in general.

2.1 Voynich Manuscript

Since the discovery of the Voynich manuscript (henceforth referred to as the VMS), there have been a number of decipherments claims. Newbold and Kent (1928) proposed an interpretation based on microscopic details in the text, which was subsequently refuted by Manly (1931). Other claimed decipherments by Feely (1943) and Strong (1945) have also been refuted (Tiltman, 1968). A detailed study of the manuscript by d'Imperio (1978) details various other proposed solutions and the arguments against them.

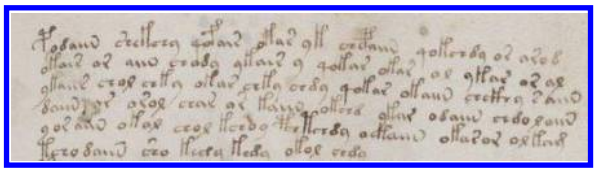

Figure 1: A sample from the Voynich manuscript.

Numerous languages have been proposed to underlie the VMS. The properties and the dating of the manuscript imply Latin and Italian as potential candidates. On the basis of the analysis of the character frequency distribution, Jaskiewicz (2011) identifies five most probable languages, which include Moldavian and Thai. Reddy and Knight (2011) discover an excellent match between the VMS and Quranic Arabic in the distribution of word lengths, as well as a similarity to Chinese Pinyin in the predictability of letters given the preceding letter.

It has been suggested previously that some anagramming scheme may alter the sequence order of characters within words in the VMS. Tiltman (1968) observes that each symbol behaves as if it had its own place in an "order of precedence" within words. Rugg (2004) notes the apparent similarity of the VMS to a text in which each word has been replaced by an alphabetically ordered anagram (*alphagram*). Reddy and Knight (2011) show that the letter sequences are generally more predictable than in natural languages.

Some researchers have argued that the VMS may be an elaborate hoax created to only appear as a meaningful text. Rugg (2004) suggests a tabular method, similar to the sixteenth century technique of the Cardan grille, although recent dating of the manuscript to the fifteenth century provides evidence to the contrary. Schinner (2007) uses analysis of random walk techniques and textual statistics to support the hoax hypothesis. On the other hand, Landini (2001) identifies in the VMS language-like statistical properties, such as Zipf's law, which were only discovered in the last century. Similarly, Montemurro and Zanette (2013) use information theoretic techniques to find long-range relationships between words and sections of the manuscript, as well as between the text and the figures in the VMS.

2.2 Algorithmic Decipherment

A monoalphabetic substitution cipher is a well-known method of enciphering a plaintext by converting it into a ciphertext of the same length using a 1-to-1 mapping of symbols. Knight et al. (2006) propose a method for deciphering substitution ciphers which is based on Viterbi decoding with mapping probabilities computed with the expectation-maximization (EM) algorithm. The method correctly deciphers 90% of symbols in a 400-letter ciphertext when a trigram character language model is used. They apply their method to ciphertext language identification using 80 different language samples, and report successful outcomes on three ciphers that represent English, Spanish, and a Spanish abjad, respectively.

Ravi and Knight (2008) present a more complex but slower method for solving substitution ciphers, which incorporates constraints that model the 1-to-1 property of the key. The objective function is again the probability of the decipherment relative to an n-gram character language model. A solution is found by optimally solving an integer linear program.

Knight et al. (2011) describe a successful decipherment of an eighteenth century text known as the Copiale Cipher. Language identification was the first step of the process. The EM-based method of Knight et al. (2006) identified German as the most likely candidate among over 40 candidate character language models. The more accurate method of Ravi and Knight (2008) was presumably either too slow or too brittle for this purpose. The cipher was eventually broken using a combination of manual and algorithmic techniques.

Hauer et al. (2014) present an approach to solving monoalphabetic substitution ciphers which is more accurate than other algorithms proposed for this task, including Knight et al. (2006), Ravi and Knight (2008), and Norvig (2009). We provide a detailed description of the method in Section 4.1.

3 Source Language Identification

In this section, we propose and evaluate three methods for determining the source language of a document enciphered with a monoalphabetic substitution cipher. We frame it as a classification task, with the classes corresponding to the candidate languages, which are represented by short sample texts. The methods are based on:

1. relative character frequencies,
2. patterns of repeated symbols within words,
3. the outcome of a trial decipherment.

3.1 Character Frequency

An intuitive way of guessing the source language of a ciphertext is by character frequency analysis. The key observation is that the relative frequencies of symbols in the text are unchanged after encipherment with a 1-to-1 substitution cipher. The idea is to order the ciphertext symbols by frequency, normalize these frequencies to create a probability distribution, and choose the closest matching distribution from the set of candidate languages.

More formally, let P_T be a discrete probability distribution where $P_T(i)$ is the probability of a randomly selected symbol in a text T being the i^{th} most frequent symbol. We define the distance between two texts U and V to be the Bhattacharyya (1943) distance between the probability distributions P_U and P_V:

$$d(U, V) = -\ln \sum_i \sqrt{P_U(i) \cdot P_V(i)}$$

The advantages of this distance metric include its symmetry, and the ability to account for events that have a zero probability (in this case, due to different alphabet sizes). The language of the closest sample text to the ciphertext is considered to be the most likely source language. This method is not only fast but also robust against letter reordering and the lack of word boundaries.

3.2 Decomposition Pattern Frequency

Our second method expands on the character frequency method by incorporating the notion of decomposition patterns. This method uses multiple occurrences of individual symbols within a word as a clue to the language of the ciphertext. For example, the word *seems* contains two instances of 's' and 'e', and one instance of 'm'. We are interested in capturing the relative frequency of such patterns in texts, independent of the symbols used.

Formally, we define a function f that maps a word to an ordered n-tuple $(t_1, t_2, \ldots t_n)$, where $t_i \geq t_j$ if $i < j$. Each t_i is the number of occurrences of the i^{th} most frequent character in the word. For example, $f(\text{seems}) = (2, 2, 1)$, while $f(\text{beams}) = (1, 1, 1, 1, 1)$. We refer to the resulting tuple as the *decomposition pattern* of the word. The decomposition pattern is unaffected by monoalphabetic letter substitution or anagramming. As with the character frequency method, we define the distance between two texts as the Bhattacharyya distance between their decomposition pattern distributions, and classify the language of a ciphertext as the language of the nearest sample text.

It is worth noting that this method requires word separators to be preserved in the ciphertext. In fact, the effectiveness of the method comes partly from capturing the distribution of word lengths in a text. On the other hand, the decomposition patterns are independent of the ordering of characters within words. We will take advantage of this property in Section 4.

3.3 Trial Decipherment

The final method that we present involves deciphering the document in question into each candidate language. The decipherment is performed with a fast *greedy-swap* algorithm, which is related to the algorithms of Ravi and Knight (2008) and Norvig (2009). It attempts to find the key that maximizes the probability of the decipherment according to a bigram character language model derived from a sample document in a given language. The decipherment with the highest probability indicates the most likely plaintext language of the document.

The greedy-swap algorithm is shown in Figure 2. The initial key is created by pairing the ciphertext and plaintext symbols in the order of decreasing frequency, with null symbols appended to the shorter of the two alphabets. The algorithm repeatedly attempts to improve the current key k by considering the "best" swaps of ciphertext symbol pairs within the key (if the key is viewed as a permutation of the alphabet, such a swap is a transposition). The best swaps are defined as those that involve a symbol occurring among the 10 least common bigrams in the decipherment induced by the current key. If any such swap yields a more probable decipherment, it is incorporated in the current key; otherwise, the algorithm terminates. The total number of iterations is bounded by m, which is set to 5 times the size of the alphabet. After the initial run, the algorithm is restarted 20 times with a randomly generated initial key, which often results in a better decipherment. All parameters were established on a development set.

1: $k_{max} \leftarrow$ InitialKey
2: **for** m iterations **do**
3: $\quad k \leftarrow k_{max}$
4: $\quad \mathcal{S} \leftarrow$ best swaps for k
5: $\quad$ **for** each $\{c_1, c_2\} \in \mathcal{S}$ **do**
6: $\quad\quad k' \leftarrow k(c_1 \leftrightarrow c_2)$
7: $\quad\quad$ **if** $p(k') > p(k_{max})$ **then** $k_{max} \leftarrow k'$
8: $\quad$ **if** $k_{max} = k$ **then return** k_{max}
9: **return** k_{max}

Figure 2: Greedy-swap decipherment algorithm.

3.4 Evaluation

We now directly evaluate the three methods described above by applying them to a set of ciphertexts from different languages. We adapted the dataset created by Emerson et al. (2014) from the text of the Universal Declaration of Human Rights (UDHR) in 380 languages.[2] The average length of the texts is 1710 words and 11073 characters. We divided the text in each language into 66% training, 17% development, and 17% test. The training part was used to derive character bigram models for each language. The development and test parts were separately enciphered with a random substitution cipher.

Table 1 shows the results of the language identification methods on both the development and the test set. We report the average top-1 accuracy on the task of identifying the source language of 380 enciphered test samples. The differences between methods are statistically significant according to McNemar's test with $p < 0.0001$. The random baseline of 0.3% indicates the difficulty of the task. The "oracle" decipherment assumes a perfect decipherment of the text, which effectively reduces the task to standard

[2]Eight languages from the original set were excluded because of formatting issues.

Method	Dev	Test
Random Selection	0.3	0.3
Jaskiewicz (2011)	54.2	47.6
Character Frequency	72.4	67.9
Decomposition Pattern	90.5	85.5
Trial Decipherment	94.2	97.1
Oracle Decipherment	98.2	98.4

Table 1: Language identification accuracy (in % correct) on ciphers representing 380 languages.

language identification.

All three of our methods perform well, with the accuracy gains reflecting their increasing complexity. Between the two character frequency methods, our approach based on Bhattacharyya distance is significantly more accurate than the method of Jaskiewicz (2011), which uses a specially-designed distribution distance function. The decomposition pattern method makes many fewer errors, with the correct language ranked second in roughly half of those cases. Trial decipherment yields the best results, which are close to the upper bound for the character bigram probability approach to language identification. The average decipherment error rate into the correct language is only 2.5%. In 4 out of 11 identification errors made on the test set, the error rate is above the average; the other 7 errors involve closely related languages, such as Serbian and Bosnian.

The trial decipherment approach is much slower than the frequency distribution methods, requiring roughly one hour of CPU time in order to classify each ciphertext. More complex decipherment algorithms are even slower, which precludes their application to this test set. Our re-implementations of the dynamic programming algorithm of Knight et al. (2006), and the integer programming solver of Ravi and Knight (2008) average 53 and 7000 seconds of CPU time, respectively, to solve a single 256 character cipher, compared to 2.6 seconds with our greedy-swap method. The dynamic programming algorithm improves decipherment accuracy over our method by only 4% on a benchmark set of 50 ciphers of 256 characters. We conclude that our greedy-swap algorithm strikes the right balance between accuracy and speed required for the task of cipher language identification.

4 Anagram Decryption

In this section, we address the challenging task of deciphering a text in an unknown language written using an unknown script, and in which the letters within words have been randomly scrambled. The task is designed to emulate the decipherment problem posed by the VMS, with the assumption that its unusual ordering of characters within words reflects some kind of a transposition cipher. We restrict the source language to be one of the candidate languages for which we have sample texts; we model an unknown script with a substitution cipher; and we impose no constraints on the letter transposition method. The encipherment process is illustrated in Figure 3. The goal in this instance is to recover the plaintext in (a) given the ciphertext in (c) without the knowledge of the plaintext language. We also consider an additional encipherment step that removes all vowels from the plaintext.

Our solution is composed of a sequence of three modules that address the following tasks: language identification, script decipherment, and anagram decoding. For the first task we use the decomposition pattern frequency method described in Section 3.2, which is applicable to anagrammed ciphers. After identifying the plaintext language, we proceed to reverse the substitution cipher using a heuristic search algorithm guided by a combination of word and character language models. Finally, we unscramble the anagrammed words into readable text by framing the decoding as a tagging task, which is efficiently solved with a Viterbi decoder. Our modular approach makes it easy to perform different levels of analysis on unsolved ciphers.

4.1 Script Decipherment

For the decipherment step, we adapt the state-of-the-art solver of Hauer et al. (2014). In this section, we describe the three main components of the solver: key scoring, key mutation, and tree search. This is followed by the summary of modifications that make the method work on anagrams.

The scoring component evaluates the fitness of each key by computing the smoothed probability of the resulting decipherment with both character-level and word-level language models. The word-level models promote decipherments that contain

(a) organized compositions through improvisational music into genres
(b) fyovicstu dfnrfecpcfie pbyfzob cnryfgcevpcfivm nzecd cipf otiyte
(c) otvfusyci cpifenfercfd bopbfzy fgyiemcpfcvrcnv nczed fpic etotyi
(d) adegiknor ciimnooopsst ghhortu aaiiilmnooprstv cimsu inot eegnrs
(e) adegiknor compositions through aaiiilmnooprstv music into greens

Figure 3: An example of the encryption and decryption process: (a) plaintext; (b) after applying a substitution cipher; (c) ciphertext after random anagramming; (d) after substitution decipherment (in the alphagram representation); (e) final decipherment after anagram decoding (errors are underlined).

in-vocabulary words and high-probability word n-grams, while the character level models allow for the incorporation of out-of-vocabulary words.

The key mutation component crucially depends on the notion of *pattern equivalence* between character strings. Two strings are pattern-equivalent if they share the same pattern of repeated letters. For example, *MZXCX* is pattern-equivalent with *there* and *bases*. but not with *otter*. For each word unigram, bigram, and trigram in the ciphertext, a list of the most frequent pattern equivalent n-grams from the training corpus is compiled. The solver repeatedly attempts to improve the current key through a series of transpositions, so that a given cipher n-gram maps to a pattern-equivalent n-gram from the provided language sample. The number of substitutions for a given n-gram is limited to the k most promising candidates, where k is a parameter optimized on a development set.

The key mutation procedure generates a tree structure, which is searched for the best-scoring decipherment using a version of beam search. The root of the tree contains the initial key, which is generated according to simple frequency analysis (i.e., by mapping the n-th most common ciphertext character to the n-th most common character in the corpus). New tree leaves are spawned by modifying the keys of current leaves, while ensuring that each node in the tree has a unique key. At the end of computation, the key with the highest score is returned as the solution.

In our anagram adaptation, we relax the definition of pattern equivalence to include strings that have the same decomposition pattern, as defined in Section 3.2. Under the new definition, the order of the letters within a word has no effect on pattern equivalence. For example, *MZXCX* is equivalent not only with *there* and *bases*, but also with *three* and *otter*, because all these words map to the $(2, 1, 1, 1)$ pattern. Internally, we represent all words as alphagrams, in which letters are reshuffled into the alphabetical order (Figure 3d). In order to handle the increased ambiguity, we use a letter-frequency heuristic to select the most likely mapping of letters within an n-gram. The trigram language models over both words and characters are derived by converting each word in the training corpus into its alphagram. On a benchmark set of 50 ciphers of length 256, the average error rate of the modified solver is 2.6%, with only a small increase in time and space usage.

4.2 Anagram Decoder

The output of the script decipherment step is generally unreadable (see Figure 3d). The words might be composed of the right letters but their order is unlikely to be correct. We proceed to decode the sequence of anagrams by framing it as a simple hidden Markov model, in which the hidden states correspond to plaintext words, and the observed sequence is composed of their anagrams. Without loss of generality, we convert anagrams into alphagrams, so that the emission probabilities are always equal to 1. Any alphagrams that correspond to unseen words are replaced with a single 'unknown' type. We then use a modified Viterbi decoder to determine the most likely word sequence according to a word trigram language model, which is derived from the training corpus, and smoothed using deleted interpolation (Jelinek and Mercer, 1980).

4.3 Vowel Recovery

Many writing systems, including Arabic and Hebrew, are abjads that do not explicitly represent vowels. Reddy and Knight (2011) provide evidence that the VMS may encode an abjad. The removal of vowels represents a substantial loss of information,

and appears to dramatically increase the difficulty of solving a cipher.

In order to apply our system to abjads, we remove all vowels in the corpora prior to deriving the language models used by the script decipherment step. We assume the ability to partition the plaintext symbols into disjoint sets of vowels and consonants for each candidate language. The anagram decoder is trained to recover complete in-vocabulary words from sequences of anagrams containing only consonants. At test time, we remove the vowels from the input to the decipherment step of the pipeline. In contrast with Knight et al. (2006), our approach is able not only to attack abjad ciphers, but also to restore the vowels, producing fully readable text.

4.4 Evaluation

In order to test our anagram decryption pipeline on out-of-domain ciphertexts, the corpora for deriving language models need to be much larger than the UDHR samples used in the previous section. We selected five diverse European languages from Europarl (Koehn, 2005): English, Bulgarian, German, Greek, and Spanish. The corresponding corpora contain about 50 million words each, with the exception of Bulgarian which has only 9 million words. We remove punctuation and numbers, and lowercase all text.

We test on texts extracted from Wikipedia articles on art, Earth, Europe, film, history, language, music, science, technology, and Wikipedia. The texts are first enciphered using a substitution cipher, and then anagrammed (Figure 3a-c). Each of the five languages is represented by 10 ciphertexts, which are decrypted independently. In order to keep the running time reasonable, the length of the ciphertexts is set to 500 characters.

The first step is language identification. Our decomposition pattern method, which is resistant to both anagramming and substitution, correctly identifies the source language of 49 out of 50 ciphertexts. The lone exception is the German article on technology, for which German is the second ranked language after Greek. This error could be easily detected by noticing that most of the Greek words "deciphered" by the subsequent steps are out of vocabulary. We proceed to evaluate the following steps assuming that the source language is known.

	Step 2	Step 3	Both	Ceiling
English	99.5	98.2	97.7	98.7
Bulgarian	97.0	94.7	91.9	95.3
German	97.3	90.6	88.7	91.8
Greek	95.7	96.6	92.7	97.2
Spanish	99.1	98.0	97.1	99.0
Average	97.7	95.7	93.8	96.5

Table 2: Word accuracy on the anagram decryption task.

The results in Table 2 show that our system is able to effectively break the anagrammed ciphers in all five languages. For Step 2 (script decipherment), we count as correct all word tokens that contain the right characters, disregarding their order. Step 3 (anagram decoding) is evaluated under the assumption that it has received a perfect decipherment from Step 2. On average, the accuracy of each individual step exceeds 95%. The values in the column denoted as *Both* are the actual results of the pipeline composed of Steps 2 and 3. Our system correctly recovers 93.8% of word tokens, which corresponds to over 97% of the in-vocabulary words within the test files, The percentage of the in-vocabulary words, which are shown in the *Ceiling* column, constitute the effective accuracy limits for each language.

The errors fall into three categories, as illustrated in Figure 3e. Step 2 introduces *decipherment errors* (e.g., deciphering 's' as 'k' instead of 'z' in "organized"), which typically preclude the word from being recovered in the next step. A *decoding error* in Step 3 may occur when an alphagram corresponds to multiple words (e.g. "greens" instead of "genres"), although most such ambiguities are resolved correctly. However, the majority of errors are caused by *out-of-vocabulary* (OOV) words in the plaintext (e.g., "improvisational"). Since the decoder can only produce words found in the training corpus, an OOV word almost always results in an error. The German ciphers stand out as having the largest percentage of OOV words (8.2%), which may be attributed to frequent compounding.

Table 3 shows the results of the analogous experiments on abjads (Section 4.3). Surprisingly, the removal of vowels from the plaintext actually improves the average decipherment step accuracy to 99%. This is due not only to the reduced number of

	Step 2	Step 3	Both	Ceiling
English	99.9	84.6	84.5	98.7
Bulgarian	99.1	71.1	70.4	95.0
German	98.5	73.7	72.9	91.8
Greek	97.7	65.4	63.6	97.0
Spanish	99.8	73.7	73.3	99.0
Average	99.0	73.8	73.1	96.4

Table 3: Word accuracy on the abjad anagram decryption task.

distinct symbols, but also to the fewer possible anagramming permutations in the shortened words. On the other hand, the loss of vowel information makes the anagram decoding step much harder. However, more than three quarters of in-vocabulary tokens are still correctly recovered, including the original vowels.[3] In general, this is sufficient for a human reader to understand the meaning of the document, and deduce the remaining words.

5 Voynich Experiments

In this section, we present the results of our experiments on the VMS. We attempt to identify the source language with the methods described in Section 3; we quantify the similarity of the Voynich words to alphagrams; and we apply our anagram decryption algorithm from Section 4 to the text.

5.1 Data

Unless otherwise noted, the VMS text used in our experiments corresponds to 43 pages of the manuscript in the "type B" handwriting (VMS-B), investigated by Reddy and Knight (2011), which we obtained directly from the authors. It contains 17,597 words and 95,465 characters, transcribed into 35 characters of the Currier alphabet (d'Imperio, 1978).

For the comparison experiments, we selected five languages shown in Table 4, which have been suggested in the past as the language of the VMS (Kennedy and Churchill, 2006). Considering the age of the manuscript, we attempt to use corpora that correspond to older versions of the languages, including King James Bible, Bibbia di Gerusalemme, and Vulgate.

[3]The differences in the Ceiling numbers between Tables 2 and 3 are due to words that are composed entirely of vowels.

Language	Text	Words	Characters
English	Bible	804,875	4,097,508
Italian	Bible	758,854	4,246,663
Latin	Bible	650,232	4,150,533
Hebrew	Tanach	309,934	1,562,591
Arabic	Quran	78,245	411,082

Table 4: Language corpora.

5.2 Source Language

In this section, we present the results of our ciphertext language identification methods from Section 3 on the VMS text.

The closest language according to the letter frequency method is Mazatec, a native American language from southern Mexico. Since the VMS was created before the voyage of Columbus, a New World language is an unlikely candidate. The top ten languages also include Mozarabic (3), Italian (8), and Ladino (10), all of which are plausible guesses. However, the experiments in Section 3.4 demonstrate that the frequency analysis is much less reliable than the other two methods.

The top-ranking languages according to the decomposition pattern method are Hebrew, Malay (in Arabic script), Standard Arabic, and Amharic, in this order. We note that three of these belong to the Semitic family. The similarity of decomposition patterns between Hebrew and the VMS is striking. The Bhattacharyya distance between the respective distributions is 0.020, compared to 0.048 for the second-ranking Malay. The histogram in Figure 4 shows Hebrew as a single outlier in the leftmost bin. In fact, Hebrew is closer to a sample of the VMS of a similar length than to any of the remaining 379 UDHR samples.

The ranking produced by the the trial decipherment method is sensitive to parameter changes; however, the two languages that consistently appear near the top of the list are Hebrew and Esperanto. The high rank of Hebrew corroborates the outcome of the decomposition pattern method. Being a relatively recent creation, Esperanto itself can be excluded as the ciphertext language, but its high score is remarkable in view of the well-known theory that the VMS text represents a constructed language.[4] We hypoth-

[4]The theory was first presented in the form of an ana-

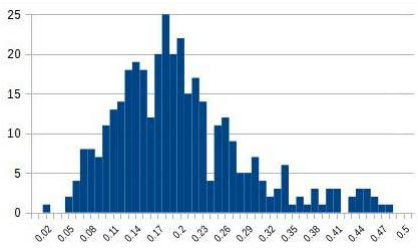

Figure 4: Histogram of distances between the VMS and samples of 380 other languages, as determined by the decomposition pattern method. The single outlier on the left is Hebrew.

esize that the extreme morphological regularity of Esperanto (e.g., all plural nouns contain the bigram 'oj') yields an unusual bigram character language model which fits the repetitive nature of the VMS words.

In summary, while there is no complete agreement between the three methods about the most likely underlying source language, there appears to be a strong statistical support for Hebrew from the two most accurate methods, one of which is robust against anagramming. In addition, the language is a plausible candidate on historical grounds, being widely-used for writing in the Middle Ages. In fact, a number of cipher techniques, including anagramming, can be traced to the Jewish Cabala (Kennedy and Churchill, 2006).

5.3 Alphagrams

In this section, we quantify the peculiarity of the VMS lexicon by modeling the words as alphagrams. We introduce the notion of the *alphagram distance*, and compute it for the VMS and for natural language samples.

We define a word's alphagram distance with respect to an ordering of the alphabet as the number of letter pairs that are in the wrong order. For example, with respect to the QWERTY keyboard order, the word *rye* has an alphagram distance of 2 because it contains two letter pairs that violate the order: (r, e) and (y, e). A word is an alphagram if and only if its alphagram distance is zero. The maximum alphagram distance for a word of length n is equal to the number of its distinct letter pairs.

gram (Friedman and Friedman, 1959). See also a more recent proposal by Balandin and Averyanov (2014).

In order to quantify how strongly the words in a language resemble alphagrams, we first need to identify the order of the alphabet that minimizes the total alphagram distance of a representative text sample. The decision version of this problem is NP-complete, which can be demonstrated by a reduction from the path variant of the traveling salesman problem. Instead, we find an approximate solution with the following greedy search algorithm. Starting from an initial order in which the letters first occur in the text, we repeatedly consider all possible new positions for a letter within the current order, and choose the one that yields the lowest total alphagram distance of the text. This process is repeated until no better order is found for 10 iterations, with 100 random restarts.

When applied to a random sample of 10,000 word tokens from the VMS, our algorithm yields the order `4BZOVPEFSXQYWC28ARUTIJ3*GHK69MDLN5`, which corresponds to the average alphagram distance of 0.996 (i.e., slightly less than one pair of letters per word). The corresponding result on English is `jzbqwxcpathofvurimslkengdy`, with an average alphagram distance of 2.454. Note that the letters at the beginning of the sequence tend to have low frequency, while the ones at the end occur in popular morphological suffixes, such as $-ed$ and $-ly$. For example, the beginning of the first article of the UDHR with the letters transposed to follow this order becomes: *"All ahumn biseng are born free and qaule in tiingdy and thrisg."*

To estimate how close the solution produced by our greedy algorithm is to the actual optimal solution, we also calculate a lower bound for the total alphagram distance with any character order. The lower bound is $\sum_{x,y} \min(b_{xy}, b_{yx})$, where b_{xy} is the number of times character x occurs before character y within words in the text.

Figure 5 shows the average alphagram distances for the VMS and five comparison languages, each represented by a random sample of 10,000 word tokens which exclude single-letter words. The *Expected* values correspond to a completely random intra-word letter order. The *Lexicographic* values correspond to the standard alphabetic order in each language. The actual minimum alphagram distance is between the *Lower Bound* and the *Computed Minimum* obtained by our greedy algorithm.

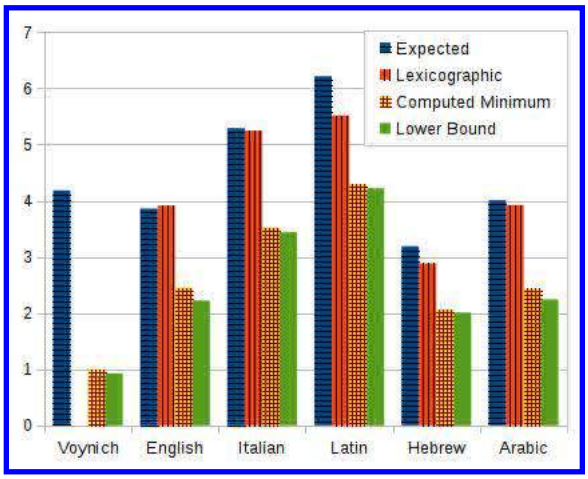

Figure 5: Average word alphagram distances.

The results in Figure 5 show that while the expected alphagram distance for the VMS falls within the range exhibited by natural languages, its minimum alphagram distance is exceptionally low. In absolute terms, the VMS minimum is less than half the corresponding number for Hebrew. In relative terms, the ratio of the expected distance to the minimum distance is below 2 for any of the five languages, but above 4 for the VMS. These results suggest that, if the VMS encodes a natural language text, the letters within the words may have been reordered during the encryption process.

5.4 Decipherment Experiments

In this section, we discuss the results of applying our anagram decryption system described in Section 4 to the VMS text.

We decipher each of the first 10 pages of the VMS-B using the five language models derived from the corpora described in Section 5.1. The pages contain between 292 and 556 words, 3726 in total. Figure 6 shows the average percentage of in-vocabulary words in the 10 decipherments. The percentage is significantly higher for Hebrew than for the other languages, which suggests a better match with the VMS. Although the abjad versions of English, Italian, and Latin yield similar levels of in-vocabulary words, their distances to the VMS language according to the decomposition pattern method are 0.159, 0.176, and 0.245 respectively, well above Hebrew's 0.020.

None of the decipherments appear to be syntactically correct or semantically consistent. This is expected because our system is designed for pure monoalphabetic substitution ciphers. If the VMS indeed represents one of the five languages, the amount of noise inherent in the orthography and the transcription would prevent the system from producing a correct decipherment. For example, in a hypothetical non-standard orthography of Hebrew, some prepositions or determiners could be written as separate one-letter words, or a single phoneme could have two different representations. In addition, because of the age of the manuscript and the variety of its hand-writing styles, any transcription requires a great deal of guesswork regarding the separation of individual words into distinct symbols (Figure 1). Finally, the decipherments necessarily reflect the corpora that underlie the language model, which may correspond to a different domain and historical period.

Nevertheless, it is interesting to take a closer look at specific examples of the system output. The first line of the VMS (`VAS92 9FAE AR APAM ZOE ZOR9 QOR92 9 FOR ZOE89`) is deciphered into Hebrew as **ועשה לה הכהן איש אליו לביחו ו עלי אנשיו המצות**.[5] According to a native speaker of the language, this is not quite a coherent sentence. However, after making a couple of spelling corrections, Google Translate is able to convert it into passable English: *"She made recommendations to the priest, man of the house and me and people."* [6]

Even though the input ciphertext is certainly too noisy to result in a fluent output, the system might still manage to correctly decrypt individual words in a longer passage. In order to limit the influence of context in the decipherment, we restrict the word language model to unigrams, and apply our system to the first 72 words (241 characters)[7] from the "Herbal" section of the VMS, which contains drawings of plants. An inspection of the output reveals several words that would not be out of place in a medieval herbal, such as **הצר** '*narrow*', **איכר** '*farmer*', **אור** '*light*', **אויר** '*air*', **אש** '*fire*'.

The results presented in this section could be interpreted either as tantalizing clues for Hebrew as

[5] Hebrew is written from right to left.

[6] https://translate.google.com/ (accessed Nov. 20, 2015).

[7] The length of the passage was chosen to match the number of symbols in the Phaistos Disc inscription.

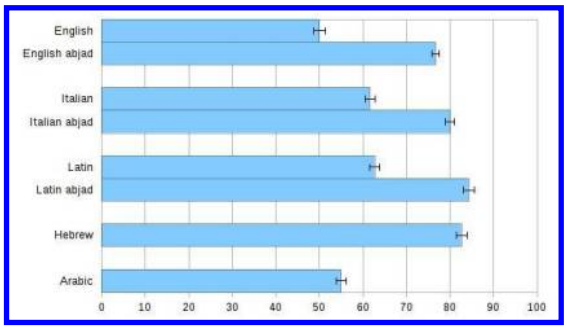

Figure 6: Average percentage of in-vocabulary words in the decipherments of the first ten pages of the VMS.

the source language of the VMS, or simply as artifacts of the combinatorial power of anagramming and language models. We note that the VMS decipherment claims in the past have typically been limited to short passages, without ever producing a full solution. In any case, the output of an algorithmic decipherment of a noisy input can only be a starting point for scholars that are well-versed in the given language and historical period.

6 Conclusion

We have presented a multi-stage system for solving ciphers that combine monoalphabetic letter substitution and unconstrained intra-word letter transposition to encode messages in an unknown language.[8] We have evaluated three methods of ciphertext language identification that are based on letter frequency, decomposition patterns, and trial decipherment, respectively. We have demonstrated that our language-independent approach can effectively break anagrammed substitution ciphers, even when vowels are removed from the input. The application of our methods to the Voynich manuscript suggests that it may represent Hebrew, or another abjad script, with the letters rearranged to follow a fixed order.

There are several possible directions for the future work. The pipeline approach presented in this paper might be outperformed by a unified generative model. The techniques could be made more resistant to noise; for example, by softening the emission model in the anagram decoding phase. It would also be interesting to jointly identify both the language and the *type* of the cipher (Nuhn and Knight, 2014), which could lead to the development of methods to handle more complex ciphers. Finally, the anagram decoding task could be extended to account for the transposition of words within lines, in addition to the transposition of symbols within words.

[8] Software at https://www.cs.ualberta.ca/~kondrak/.

Acknowledgements

We thank Prof. Moshe Koppel for the assessment of the Hebrew examples. We thank the reviewers for their comments and suggestions.

This research was supported by the Natural Sciences and Engineering Research Council of Canada, and by Alberta Innovates – Technology Futures and Alberta Innovation & Advanced Education.

References

Arcady Balandin and Sergey Averyanov. 2014. The Voynich manuscript: New approaches to deciphering via a constructed logical language.

A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109.

Mary E. d'Imperio. 1978. The Voynich manuscript: An elegant enigma. Technical report, DTIC Document.

Guy Emerson, Liling Tan, Susanne Fertmann, Alexis Palmer, and Michaela Regneri. 2014. Seedling: Building and using a seed corpus for the human language project. In *Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 77–85.

Joseph Martin Feely. 1943. *Roger Bacon's Cypher. The Right Key Found*. Rochester, NY.

William F. Friedman and Elizebeth S. Friedman. 1959. Acrostics, anagrams, and Chaucer. *Philological Quarterly*, 38(1):1–20.

Bradley Hauer, Ryan Hayward, and Grzegorz Kondrak. 2014. Solving substitution ciphers with combined language models. In *COLING*, pages 2314–2325.

Grzegorz Jaskiewicz. 2011. Analysis of letter frequency distribution in the Voynich manuscript. In *International Workshop on Concurrency, Specification and Programming (CS&P'11)*, pages 250–261.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. *Pattern recognition in practice*.

Gerry Kennedy and Rob Churchill. 2006. *The Voynich manuscript: The mysterious code that has defied interpretation for centuries*. Inner Traditions/Bear & Co.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *COLING/ACL*, pages 499–506.

Kevin Knight, Beáta Megyesi, and Christiane Schaefer. 2011. The Copiale cipher. In *4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 2–9.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, volume 5, pages 79–86.

Gabriel Landini. 2001. Evidence of linguistic structure in the Voynich manuscript using spectral analysis. *Cryptologia*, 25(4):275–295.

John Matthews Manly. 1931. Roger Bacon and the Voynich MS. *Speculum*, 6(03):345–391.

Marcelo A. Montemurro and Damián H. Zanette. 2013. Keywords and co-occurrence patterns in the Voynich manuscript: An information-theoretic analysis. *PloS one*, 8(6):e66344.

George Nagy, Sharad Seth, and Kent Einspahr. 1987. Decoding substitution ciphers by means of word matching with application to OCR. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):710–715.

William Romaine Newbold and Roland Grubb Kent. 1928. *The Cipher of Roger Bacon*. University of Pennsylvania Press.

Peter Norvig. 2009. Natural language corpus data. In Toby Segaran and Jeff Hammerbacher, editors, *Beautiful data: The stories behind elegant data solutions*. O'Reilly.

Malte Nuhn and Kevin Knight. 2014. Cipher type detection. In *EMNLP*, pages 1769–1773.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *EMNLP*, pages 812–819.

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *NAACL*, pages 37–45.

Sravana Reddy and Kevin Knight. 2011. What we know about the Voynich manuscript. In *5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 78–86.

Andrew Robinson. 2002. *Lost languages: The enigma of the world's undeciphered scripts*. McGraw-Hill.

Gordon Rugg. 2004. An elegant hoax? A possible solution to the Voynich manuscript. *Cryptologia*, 28(1):31–46.

Andreas Schinner. 2007. The Voynich manuscript: Evidence of the hoax hypothesis. *Cryptologia*, 31(2):95–107.

Klaus Schmeh. 2013. A milestone in Voynich manuscript research: Voynich 100 conference in Monte Porzio Catone, Italy. *Cryptologia*, 37(3):193–203.

Simon Singh. 2011. *The code book: The science of secrecy from ancient Egypt to quantum cryptography*. Anchor.

Leonell C Strong. 1945. Anthony Askham, the author of the Voynich manuscript. *Science*, 101(2633):608–609.

John Tiltman. 1968. *The Voynich Manuscript, The Most Mysterious Manuscript in the World*. Baltimore Bibliophiles.