# DECIPHERMENT OF HISTORICAL MANUSCRIPTS

by

Nada Aldarrab

---

A Thesis Presented to the

FACULTY OF THE USC GRADUATE SCHOOL

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

MASTER OF SCIENCE

(COMPUTER SCIENCE)

May 2017

# Acknowledgments

As I write these words, I am overwhelmed by the number of people I have been lucky to have throughout this journey. Apparently, one page of acknowledgments is not enough to thank all these amazing people.

I believe no words can express my gratitude to my wonderful parents, Munirah Alzamil and Ibrahim Aldarrab. Just thinking that my name was always there in their prayers helped me overcome many hard times. Thank you for being there whenever I needed you, even when I thought I did not. Thank you so much for all the love, encouragement, and support you have provided throughout my entire life.

I have been blessed to have Kevin Knight as my thesis advisor, not only for his scholarship but also for his patience and great support. Over the whole year, Kevin has been a great source of information, guidance, and laughter. I am totally indebted to him for such an enjoyable experience.

I have also been fortunate to have Daniel Marcu and Jonathan May on my thesis committee. I have benefited a lot from their comments in our weekly handout meetings, and later, on my thesis drafts. Their feedback has always been invaluable in correcting me and pointing me towards interesting directions.

A big thank you to our amazing collaborator, Beáta Megyesi (Uppsala University, Sweden). This work would not have been possible without the astonishing

work and effort that she had put into this project. Thank you for the awesome team that helped us from Sweden.

My friends have probably been the major unseen collaborators of this thesis. They have always been there for me and supported me even though we are thousands of miles apart. Thank you Zeynep Betül Kuran, Fatma Alkassimi, Müge Doğan, and Samaher Kozzabah.

I have been lucky to have had the chance to work closely with the outstanding people of the natural language group at ISI. I have learned the most from our interactions and discussions. Thank you to Yonatan Bisk, Aliya Deri, Marjan Ghazvininejad, Ulf Hermjakob, Michael Pust, Nima Pourdamghani, Xing Shi, Ashish Vaswani, Barret Zoph, and to the incredible summer visitors in 2016: Angeliki Lazaridou, Xiang Li, Sebastian Mielke, and Ke Tran.

Special thanks to Flor Martinez, my wonderful academic advisor, and Peter Zamar, our amazing project assistant at ISI. You have always been a great inspiration, and you have always made my life much easier.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Throughout history, people have used many clever ways to send secret messages. Codes and ciphers have been used by military and diplomatic forces to keep confidential information from adversaries. Businesses also sent encoded data to protect trade secrets. Many other people have used ciphers to conceal information, including scientists, criminals, and secret societies.

The mysteries surrounding those ciphers have sparked much interest among amateur enthusiasts and computer scientists. In fact, major advancements in computing were inspired by the intention of code breaking during World War II (Clements, 2013). In this thesis, we discuss different methods for automatically solving historical ciphers and apply those methods to crack real historical ciphers.

## 1.1 The big picture

European libraries are filled with undeciphered historical manuscripts from the 16th-18th centuries. These documents are enciphered with classical methods, which puts their contents out of the reach of historians who are interested in the history of that era. Our big goal is to decipher the large collection of ciphers we have obtained from European archives with the help of a Swedish team from Uppsala University. Figure 1.1 shows an example of a historical cipher from our collection.

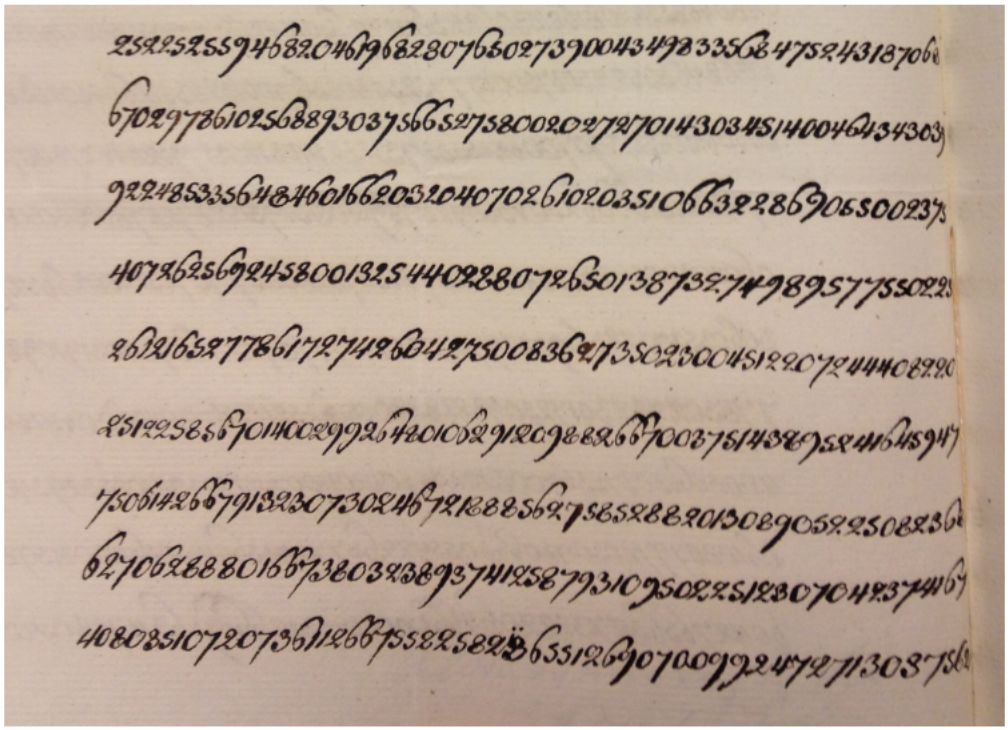


Figure 1.1: An example cipher from our historical cipher collection.

We first need to transcribe those ciphers into computer-readable format, and then we need to decipher them. We want to do this fully automatically so that someone could take a cellphone camera into an archive, point it at a new cipher, and see the plaintext decipherment appear on the screen.

## 1.2 Main decipherment tasks

Decipherment conditions vary from one cipher to another. In the most mysterious case, the cryptanalyst only has access to the ciphertext. This means that the encipherment method, the plaintext language, and the key are all unknown. This is called a *ciphertext-only attack*. This section describes the three main tasks involved in ciphertext-only attacks; cipher type detection, plaintext language identification (plaintext language ID), and finding the key.

### 1.2.1 Cipher type detection

Various encipherment methods can be used to create ciphers. Since we are dealing with pre-computer ciphers, we focus our discussion on the encipherment methods that were used in the early modern ages. In his book "A Brief History of Cryptology and Cryptographic Algorithms," Dooley (2013) describes three types of cryptograms predominantly used in that era; codes, ciphers, and nomenclatures. A code is created by substituting a numerical or alphabetic *codeword* for a complete word from the plaintext. A cipher is created by transforming smaller language elements (usually characters) into ciphertext. A nomenclature is a combination of a cipher and a small codebook. Figure 1.2 shows an example of a nomenclature key from our cipher collection.

Ciphers come in two general categories; *substitution ciphers* and *transposition ciphers*. Substitution ciphers are created by replacing each letter in a message with a cipher symbol, whereas transposition ciphers are created by rearranging the letters of a message. Substitution ciphers can use just a single cipher alphabet where each plaintext letter type is deterministically replaced with one cipher letter type; these are known as *simple substitution ciphers* (or *1:1 substitution ciphers*). A substitution cipher that provides multiple substitutions for some letters (i.e. 1:M substitutions) is called a *homophonic cipher*.

Various methods have been proposed for cipher type detection (Nuhn and Knight, 2014). We survey those methods in chapter 2. In this work, we focus on solving substitution ciphers, including simple substitution and homophonic ciphers. Thus, we do not address the problem of cipher type detection.

Figure 1.2: A nomenclature key from our historical cipher collection. Top part shows the cipher key (plaintext letters and corresponding ciphertext). Bottom part shows a small codebook for the most frequent words (usually used to encode proper names).

### 1.2.2 Plaintext language identification

Sometimes, ciphers are accompanied with some unrelated cleartext. This is usually a strong clue of the plaintext language of the ciphertext. Most ciphers, however, are completely enciphered, so it is crucial to identify the language of the plaintext before trying to find the key. We present automatic methods for plaintext language ID in section 3.4.

### 1.2.3 Finding the key

Once we have a theory about the cipher type and plaintext language, we can proceed to the next step; finding the key. The key is usually a letter substitution table or a codebook. This is a very challenging task. Throughout history, many methods have been used by cryptographers to challenge cryptanalysts. For example, many ciphers in the early modern ages include *nulls* (Dooley, 2013). Nulls are spurious characters added randomly to the cipher to confuse cryptanalysts. Those characters are not part of the enciphered message and are only used to further conceal the message. Methods for finding the key range from manual frequency analysis (800s C.E.) to modern, computer-based methods (Dooley, 2013). We survey those methods in section 2.1.

## 1.3 Contributions of this thesis

The major contributions of this thesis are as follows:

- We present a machine learning technique for fast plaintext language ID for 1:1 substitution ciphers, which achieves a top-3 accuracy of 87% on 512-character ciphers with spaces and a top-3 accuracy of 84% on 512-character ciphers without spaces.

- We use the noisy-channel framework to automatically crack two historical ciphers; the "Borg" cipher and the Oak Island Cipher.
- We implement an unsupervised end-to-end system aimed at deciphering from images and report our results on deciphering printed text images vs. handwritten historical text images.
- We present an integer linear programming (ILP) method for evaluating transcription accuracy.

# Chapter 2

# Literature Review

This chapter reviews previous decipherment and Optical Character Recognition (OCR) techniques. Section 2.1 surveys related work on decipherment, including cipher type detection, language ID, statistical decipherment approaches, and various applications for decipherment in other Natural Language Processing (NLP) tasks. This survey is by no means exhaustive and aims to give an idea of related work. Section 2.2 reviews previous literature on OCR, from the early decipherment-based approaches to the modern, supervised neural techniques.

## 2.1 Decipherment

Unsupervised learning has played a major role in many advances in natural language processing. Even though we are witnessing great interest in supervised techniques, encouraged by high computing power and the vastness of training data, we still face many problems where supervised learning is not an option. These include deciphering unknown scripts, like the Voynich manuscript, or enciphered texts such as the Zodiac killer ciphers (figure 4.1). Machine translation of low-resource human languages is another, where we do not have enough parallel text to train data-hungry supervised models.

Earlier methods for attacking ciphers were based on *frequency analysis*, a method discovered by the great polymath, Abu Yūsuf Yaʿqūb ibn ʾIsḥāq al-Kindī (801-873 C.E.). The method of frequency analysis was first described in his book

on secret messages, *A Manuscript on Deciphering Cryptographic Messages*, which is considered one of the most important books in the history of cryptology (Dooley, 2013). Frequency analysis has become a fundamental method in cryptanalysis and is usually the first step cryptanalysts take to get an idea of the cipher type and properties. Letter frequencies and automatic clustering of cipher letters based on similarity of context were among the first steps that led to deciphering the Copiale cipher by Knight et al. (2011).

Automatic methods have been suggested for cipher type detection. Nuhn and Knight (2014) took a machine learning approach to solve this problem by training a classifier that is able to predict the encipherment method given a ciphertext. Their best results were obtained by extracting 58 features and training a linear classifier on 1M training examples, which achieved an accuracy of 58.49% on a test set of 305 ciphers (from 50 different cipher types). Although we do not address the problem of cipher type detection in this work, we are inspired by this approach and take a similar approach to solve the plaintext language ID problem as shown in chapter 3.

An enormous amount of work has been done on developing automatic methods for language ID of unenciphered texts. Various language modeling and machine learning techniques have shown great success on the language ID task, even in the context of confusable languages (Malmasi and Dras, 2015). However, such methods cannot be used for language ID of enciphered text because they mainly rely on the actual character and word n-grams, which are hidden in ciphertexts. Knight et al. (2006) propose an unsupervised approach for language ID in the context of phonetic decipherment. They suggest two methods for dealing with scripts where the language behind the script is unknown. The first method is to look for universal constraints on phoneme sequences as proposed by linguists. The

second method is to build phoneme n-gram language models and decipher against each one of them to find the source language that has the best fit. We show how this method is particularly useful for ciphertext-only attacks in chapter 3.

Automatic, unsupervised methods have shown great success in attacking decipherment problems. Knight et al. (2006) suggest an unsupervised method for solving four natural language decipherment problems; letter substitution ciphers, character code conversion, phonetic decipherment, and word-based ciphers. Their suggested method follows the noisy-channel framework where we only observe the ciphertext, and they use expectation-maximization (EM) to set the free channel model parameters, guided by a pre-trained source language model. This method proves powerful in attacking many synthetic and historical substitution ciphers, as we show in chapters 3 and 4.

Several other methods have been suggested for solving 1:1 substitution ciphers. Corlett and Penn (2010) solve the problem using A* search. Ravi and Knight (2008) use low-order letter n-gram models and enforce deterministic key constraints using integer linear programming (ILP). Although this method yields optimal solutions, it is computationally expensive and very slow, which makes it unsuitable for solving long ciphers.

Deciphering historical manuscripts has been a fascinating challenge for people in the NLP community. Ravi and Knight (2011) report the first automatic decipherment of the Zodiac-408 cipher using a combination of a 3-gram language model and a word dictionary. In this work, we only use a 5-gram letter-based language model to crack the Zodiac-408. Nuhn et al. (2013) use beam search with a high order (6-gram) letter-based language model to solve 1:1 substitution and homophonic ciphers. They also report successful decipherment of the Zodiac-408 cipher. Nuhn et al. (2014) report the first automatic decipherment of the second

part of the Beale cipher. Many other attempts have been targeted towards the Zodiac-340 cipher, which is yet to be solved. Berg-Kirkpatrick and Klein (2013) used the HMM approach described in (Knight et al., 2006). They used a row-major reading order and 1 million random restarts to attack the Zodiac-340 cipher, but the decoding they got was nonsensical.

Decipherment has been used to improve performance on many other NLP tasks. One example is machine translation (MT). Since parallel corpora are expensive and not available for every language, decipherment is used to leverage the usually more abundant monolingual data to train translation models. Example applications include low-resource machine translation (Dou and Knight, 2013; Dou et al., 2014, 2015), out-of-domain machine translation (Dou and Knight, 2012), and decipherment of lost languages (Snyder et al., 2010).

## 2.2 Optical character recognition (OCR)

Handwriting recognition is traditionally divided into two types; on-line and off-line recognition (Liwicki et al., 2012). On-line handwriting recognition systems record a time ordered sequence of coordinates that represent the movement of the pen-tip. On the other hand, off-line recognition systems work only on an image of the text. In our case, we are facing an off-line recognition problem where we try to decipher historical manuscripts from available scanned images of handwritten text.

Early automatic approaches treat OCR as a cryptogram decoding problem (Huang et al., 2007). Such methods are based on unsupervised character clustering followed by mapping character clusters onto letters, using techniques for solving simple substitution ciphers (Nagy et al., 1987; Ho and Nagy, 2000; Fang and Hull,

1995). Those unsupervised methods were abandoned in favor of supervised techniques with the abundance of manually transcribed training data. Current state-of-the art handwriting recognition systems use recurrent neural networks (RNNs) or a hybrid of hidden Markov models (HMMs) and RNNs (Liwicki et al., 2012). For example, Liwicki et al. (2012) use multidimensional long short-term memory networks (MDLSTMs) for the offline Arabic handwriting recognition task, which yields an 81.06% word accuracy.

Such supervised methods are clearly not an option for us since we have no training data for our historical cipher collection. Annotating data is very expensive and especially hard for degraded documents. Moreover, ciphers usually use unique symbol sets and have a great variance in handwriting styles. This makes it hard to reuse any annotated data, which makes supervised methods an unfeasible option.

Several works on OCR have been published in the NLP community. Zhang et al. (2016) propose an end-to-end image to translation system that targets the endangered Nyushu script. They take a pipelined approach to join OCR and machined translation (MT) (figure 2.1). The process starts with character segmentation, which is very straightforward since the Nyushu characters are nicely separated and laid out on the page. Then they extract image-based statistical features to describe the segmented characters (a feature vector of 175 dimensions for each character image). This is followed by linking character images to standard Nyushu characters, which they formulate as a multi-class classification problem. Results from OCR are given to a Nyushu-to-Chinese machine translation engine as a Nyushu lattice. We are inspired by this pipelined approach to OCR, as we show in chapter 5.

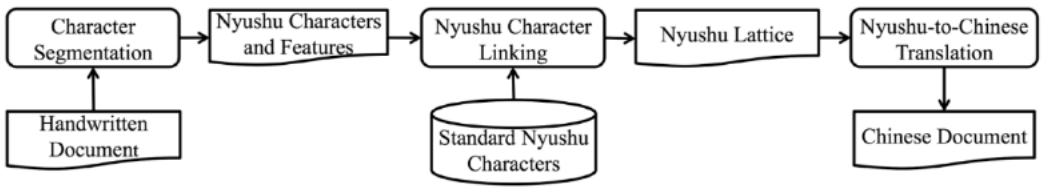



Figure 2.1: Zhang et al. (2016) propose a pipelined approach to OCR and machine translation, which targets the endangered Nyushu script.

Other works on OCR target printed documents. Berg-Kirkpatrick et al. (2013) use a generative probabilistic model to build what has become the current state-of-the art OCR system for historical printed documents, *Ocular*. They model the generation of images as the joint distribution of four models; a language model, a typesetting model, an inking model, and a noise model. They test their system on two historical datasets from the printing press era; *Old Bailey* and *Trove* and report improved performance on the test set over the commercial OCR system, *ABBYY FineReader 11* and Google's open source OCR system, *Tesseract* (Smith, 2007). However, this method has some limitations that prevents adaption to handwritten historical ciphers. One is that the system requires a set of character prototypes to start with. Then the system can learn different fonts from the observed data. This makes the system unsuitable for the highly variant ciphertext symbols. Another limitation is that the system leverages whitespaces between characters, which exist naturally in printed documents, but are very rare in handwritten documents.

All ciphers in our collection are handwritten, but the characters are not nicely separated as in the Nyushu script. So, character segmentation is much more challenging. And as mentioned previously, we do not have any labeled training data for linking cipher characters to images. So, we take an unsupervised character clustering approach instead. We report our OCR experiments in chapter 5.

# Chapter 3

# Decipherment Experiments

Before we get to real historical ciphers, we need to establish decipherment methods that work on *synthetic* ciphers (i.e. ciphers that we create and know the key for).

This chapter describes our experiments on deterministic simple substitution and homophonic synthetic ciphers. It describes our datasets, methods, and results. We also address the problem of plaintext language ID in section 3.4.

## 3.1 Challenge cryptanalysis problems

We start our experiments with a set of nine synthetic ciphers. Our goal is to automatically decipher them. These ciphers range in difficulty from simple substitution ciphers with spaces (what the American Cryptogram Association call *aristocrats*, as opposed to *patristocrats*, which hide word divisions) to homophonic ciphers with spaces removed. Figure 3.1 shows an example simple substitution synthetic cipher. Figure 3.2 shows an example homophonic synthetic cipher, with spaces removed.

```
kac butnqymkupqmr tckauv ql m tckauv ux rqyzjqlkqb mymrdlql
kamk ql jlcv ku lkjvd kcfkl gaqba mpc gpqkkcy qy my jyeyugy
rmyzjmzc myv ku lkjvd kac rmyzjmzc qklcrx gacpc kac jyeyugy
rmyzjmzc aml yu unhqujl up ipuhcy gcrrjyvcplkuuv brulc
pcrmkqhcl myv gacpc kacpc mpc xcg nqrqyzjmr kcfkl gaqba tqzak
ukacpgqlc amhc nccy jlcv ku acri jyvcplkmyv kac rmyzjmzc
```

Figure 3.1: An example simple substitution synthetic cipher.

47 21 11 24 19 35 27 02 11 30 51 38 22 37 02 41 40 35 39 50 01
41 34 18 14 20 44 28 40 14 31 10 06 45 34 49 47 04 44 19 13 43
09 43 52 20 13 45 23 14 27 39 29 08 14 15 02 41 34 47 44 34 42
54 03 48 09 47 07 49 34 16 04 37 27 12 29 45 47 34 29 06 42 23
46 30 38 45 40 14 01 24 22 45 19 15 25 40 31 19 47 05 22 23 44
26 52 08 39 47 38 51 02 43 19 45 11 30 44 19 25 10 44 52 13 15
02 41 25 30 20 48 09 37 48 20 42 47 07 24 32 30 51 05 19 41 21
43 44 32 31 21 25 12 44 08 47 10 49 28 20 35 48 23 42 12 27 17
23 43 31 21 42 08 36 24 21 22 18 14 07 48 32 12 14 32 16 43 01
45 03 08 36 48 19 15 02 35 51 34 47 07 17 02 36 45 21 28 40 08
44 03 04 18 13 24 48 02 38 12 44 08 17 01 49 33 16 45 29 19 11
45 22 30 19 24 07 39 37 08 42 10 55 45 47 08 50 04 18 14 13 21
48 03 44 39 37 51 31 10 15 16 34 41 09 34 44 52 23 30 06 16 25
02 44 32 42 37 23 24 02 38 36 23 47 12 42 45 09 30 54 06 44 01
09 37 48 33 42 53 33 02 22 50 13 30 39 37 38 49 04 47 06 18 46
39 46 32 47 34 15 01 17 24 04 34 16 36 23 26 03 40 35 06 14 15
33 45 11 28 06 38 31 02 49 33 16 22 36 37 02 16 22 41 46 47 01
39 31 08 47 06 49 12 29 26 21 24 11 51 20 32 48 10 27 20 29 49
02 24 29 21 42 32 29 52 32 36 44 08 50 10 35 26 34 41 54 07 48
12 42 08 05 34 14 20 34 51 09 30 11 12 25 30 39 49 38 30 23 28
09 45 38 18 14 08 53 34 23 48 13 28 09 52 38 43

Figure 3.2: An example homophonic synthetic cipher, with spaces removed.

Table 3.1 shows a summary of the lengths, types and properties of the nine synthetic ciphers. The full set of ciphers is shown in appendix A.

| Cipher No. | Type | # chars | Spaces? | Language |
|---|---|---|---|---|
| 1 | simple substitution | 353 | yes | English |
| 2 | simple substitution | 150 | yes | English |
| 3 | simple substitution | 653 | no | English |
| 4 | simple substitution | 128 | yes | Unspecified |
| 5 | simple substitution | 107 | yes | Unspecified |
| 6 | simple substitution | 331 | no | Unspecified |
| 7 | simple substitution | 168 | no | Unspecified |
| 8 | homophonic | 2376 | no | Unspecified |
| 9 | homophonic | 436 | no | Unspecified |

Table 3.1: Summary of the properties of the nine synthetic ciphers we used for our decipherment experiments. The full set of ciphers is shown in appendix A.

## 3.2 Data sets

Since we are targeting historical ciphers, we start by collecting historical text for various European languages. We use a dataset created by Barret Zoph, which includes historical text for 20 different languages scraped from Project Gutenberg. We use 13 of those languages, namely: Spanish, Latin, Esperanto, Hungarian, Icelandic, Danish, Norwegian, Dutch, Swedish, Catalan, French, Portuguese, and Finnish.

To this, we add three languages: German, English, and Italian, as these languages seem relevant to our cipher collection. For German and Italian, we scrape historical text from Project Gutenberg. We use Wikipedia dumps to get data for English. This completes our dataset that we will be using to build language models. Table 3.2 summarizes our datasets.

| language | # of words | # of characters |
|---|---|---|
| Catalan | 915,595 | 4,953,516 |
| Danish | 2,077,929 | 11,205,300 |
| Dutch | 30,350,145 | 177,835,527 |
| English | 48,041,703 | 289,170,305 |
| Esperanto | 315,423 | 2,079,649 |
| Finnish | 22,784,172 | 168,886,663 |
| French | 39,400,587 | 226,310,827 |
| German | 3,273,602 | 20,927,065 |
| Hungarian | 497,402 | 3,145,451 |
| Icelandic | 72,629 | 377,910 |
| Italian | 4,587,027 | 27,786,754 |
| Latin | 1,375,804 | 8,740,808 |
| Norwegian | 706,435 | 3,673,895 |
| Portuguese | 10,841,171 | 62,735,255 |
| Spanish | 20,165,731 | 114,663,957 |
| Swedish | 3,008,680 | 16,993,146 |

Table 3.2: Summary of data sets obtained from Project Gutenberg and English Wikipedia.

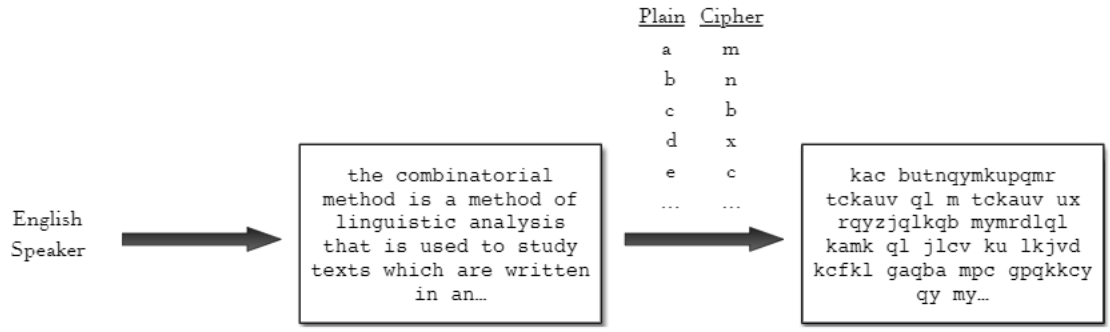
## 3.3 Decipherment methods

Let's assume for now that we know the plaintext language of the cipher. To find the actual plaintext, we follow the well-known noisy-channel framework, which can be used for attacking decipherment problems (Knight et al., 2006). Figure 3.3 depicts the noisy-channel formulation of the decipherment problem. In this

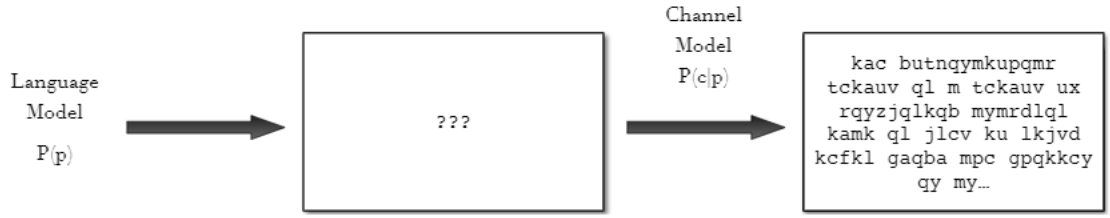
formulation, we think about a *generative story* of how the ciphertext was created. To create an English 1:1 substitution cipher, we can imagine that:

(a) Someone first came up with some English text (the plaintext).

(b) Then they enciphered each plaintext character according to a 1:1 substitution table (the key) (Figure 3.3(a)).




(a) A generative story of how the ciphertext was created. We imagine that someone first came up with some English text (the plaintext). Then they enciphered each plaintext character according to a 1:1 substitution table (the key)




(b) In a ciphertext-only attack, we only have the ciphertext, and we want to find the plaintext. We model the plaintext generation process by an English language model $P(p)$ that assigns a probability to each English string $p$, and we model the encipherment process by a channel model $P(c|p)$ that assigns a probability for each character substitution

Figure 3.3: The noisy-channel formulation of the decipherment problem.

In a ciphertext-only attack, we only have the ciphertext, and we want to find the plaintext. We model the plaintext generation process by an English language model $P(p)$ that assigns a probability to each English string $p$, and we model the

encipherment process by a channel model $P(c|p)$ that assigns a probability for each character substitution (Figure 3.3(b)). Our objective is to find the plaintext $p$ that maximizes $P(p|c)$ for a given cipher $c$. That is:

$$\arg\max_{p} P(p|c)$$

By Bayes rule:

$$\arg\max_{p} P(p|c) = \arg\max_{p} \frac{P(p) * P(c|p)}{P(c)} \tag{3.1}$$

Since $P(c)$ does not affect the choice of $p$ (i.e. $P(c)$ is constant), the equation becomes:

$$\arg\max_{p} P(p|c) = \arg\max_{p} P(p) * P(c|p) \tag{3.2}$$

So, we basically need to build two probabilistic models; a language model $P(p)$ and a channel model $P(c|p)$. Language models could be built and trained independently on any plaintext language data. The channel model explains how plaintext $p$ becomes ciphertext $c$. This could be a two-dimensional substitution table. To estimate the parameters of the channel model, we use the expectation-maximization algorithm (EM) (Dempster et al., 1977). Our objective function is to maximize $P(c)$, which (by the law of total probability) is:

$$P(c) = \sum_{p} P(p) * P(c|p) \tag{3.3}$$

Once we have the trained models, we can use the Viterbi algorithm to find the plaintext that maximizes $P(p|c)$ as given by equation 3.2.

We implement all our models as a cascade of finite-state machines (FSMs). We use the finite-state toolkit, *Carmel*, to do the EM training of the channel model and the final Viterbi decoding (Graehl, 2010).

In summary, we take the following steps to attack ciphers:

1. Build letter language models: We build letter-based language models for the 16 European Languages that we collected historical texts for: Catalan, Danish, Dutch, English, Esperanto, Finnish, French, German, Hungarian, Icelandic, Italian, Latin, Norwegian, Portuguese, Spanish, and Swedish. We experiment with different n-gram orders; 2-gram, 3-gram, 4-gram, and 5-gram. We implement these models as finite-state acceptors (FSAs). We use 80% of the data for training, 10% for development, and 10% for testing. We mainly use the development set to smooth our language models. For smoothing, we estimate context-specific backoff parameters by running EM on the development set.

2. Train the channel model to get $P(c|p)$ (EM training): We implement the channel model as a single-state, fully-connected finite-state transducer (FST). Then we use Carmel to run EM training, guided by the pre-trained language model probabilities. To get better decipherment results, we use two techniques described in previous literature; more random restarts and square-rooting language model probabilities during EM training (Ravi and Knight, 2009)

3. Decode the ciphertext to find the plaintext $p$ (Viterbi decoding): Using our trained models, we run the Viterbi algorithm to find the best path (i.e. the

plaintext that maximizes $P(p|c)$ as given by equation 3.2). As suggested by Knight and Yamada (1999), we also find that cubing channel probabilities before the final decoding results in better decipherment accuracy.

## 3.4 Plaintext language identification

Now we turn to the plaintext language ID problem. We experiment with two approaches; partial decipherment and n-way classification. The next two subsections describe these two methods.

### 3.4.1 Language identifications with partial decipherment

The idea here is to follow the same decipherment procedure described in section 3.3. Only this time, we use a low-order language model (a 3-gram language model) and fewer EM iterations (we call this *partial decipherment*). Our motivation here is to identify the plaintext language without consuming as much time or computing power as when we do full decipherment using large language models. The procedure is as follows: Given ciphertext $c$, we run partial decipherment against each of the 16 European languages, then we rank candidate languages based on some evaluation metric. Recall from equation 3.3 that EM's objective function was to maximize $P(c)$. We can use this post-training $P(c)$ to rank candidate plaintext languages.

#### Language ID results

We tested the partial decipherment method on 6 synthetic ciphers from our challenge cryptanalysis problems (ciphers 4-9, where plaintext language is unspecified as shown in table 3.1). Table 3.3 summarizes the results of our experiments. For ciphers with spaces (ciphers 4 and 5), a 3-gram language model was sufficient

to identify the plaintext language. However, for ciphers without spaces (ciphers 6-9), a 5-gram language model was necessary to correctly identify the plaintext language. Of course, a 5-gram language model takes much longer to run than a 3-gram language model, but gives more accurate results. We provide an alternative method for fast language ID for 1:1 substitution ciphers in the next section.

| Cipher No. | Type | # chars | Spaces? | Lang | LM n-gram | Time (min) |
|---|---|---|---|---|---|---|
| 4 | 1:1 substitution | 128 | yes | Spanish | 3 | 26 |
| 5 | 1:1 substitution | 107 | yes | German | 3 | 11 |
| 6 | 1:1 substitution | 331 | no | Swedish | 5 | 285 |
| 7 | 1:1 substitution | 168 | no | Portuguese | 5 | 198 |
| 8 | homophonic | 2376 | no | Latin | 5 | 229 |
| 9 | homophonic | 436 | no | Latin | 5 | 967 |

Table 3.3: Summary of language ID results on 6 synthetic ciphers using partial decipherment against 16 candidate European languages. Since the 16 partial decipherments were run in parallel, time refers to the longest time a language took to finish decipherment.

### 3.4.2 Language identification as a classification problem

As we have discussed in section 3.4.1, language ID with partial decipherment is very expensive in terms of time and computing power. Here, we take a machine learning approach to perform fast plaintext language ID.

We formulate the language identification task as an n-way classification problem (where n is the number of candidate plaintext languages). We build a database of <cipher, plaintext-language-ID> pairs by enciphering texts with random keys. Then we train a linear classifier and test our model on previously unseen ciphers. We describe our dataset, feature set, and results in the following subsections.

### Data

An interesting feature of this formulation of the problem is that we can create an infinite amount of training data. We can take any text from any set of languages and create as many ciphers as we want with random keys. To start with, we created a balanced corpus of ciphers in 16 languages. Table 3.4 summarizes our dataset properties.

| | |
|---|---|
| Train | 1M ciphers |
| Test | 100K ciphers |
| Cipher lengths | 32, 64, 128, 256, 512 chars |
| Languages | Catalan, Danish, Dutch, English, Esperanto, Finnish, French, German, Hungarian, Icelandic, Italian, Latin, Norwegian, Portuguese, Spanish, and Swedish |

Table 3.4: Summary of our synthetic simple substitution cipher dataset.

We created 2 versions of this dataset. One is the original enciphered text (simple substitution ciphers with spaces) and the other has the same set of substitution ciphers but with spaces removed. Figure 3.4 shows three ciphers from our synthetic simple substitution cipher dataset. Figure 3.5 shows sample ciphers from the second version of our dataset (ciphers with spaces removed).

| | |
|---|---|
| latin | o u c b g s h s u b _ m w b u w s g x _ b g _ b d f g o h _ b g c s _ x h y c h _ r u f d j g s u w _ x d r u h c s e w |
| hungarian | d s m n _ m _ a p e p l t v m x _ v m q v m _ f j x m x n p k p x x _ m a j q p l _ i e _ z p y _ b h e l j _ i k _ j s |
| french | f q r v h u o o h _ o h _ o l u t u h r _ g q m _ g h l c _ o h _ f q w w u g g u q m m l u r h _ g h g _ i l b c h t g |

Figure 3.4: Three ciphers from our synthetic simple substitution cipher dataset (with spaces).

| | |
|---|---|
| latin | d f u t d i r g h c d c h h y w i z u i z d d r j h y h u i n u z h d a h n z d a y f d x n x f d a r u |
| catalan | j t f y z f z q b n z b o a s b a b j j v n f m b b s c s b n v s s v s n q w v v m j f m d t s b n |
| finnish | f v n i v v d s i i v d r i b v n h d c s c v d c q f q h d k s a q i n h x v d i v v d s o b v d d r i |

Figure 3.5: Three ciphers from our synthetic simple substitution cipher dataset (spaces removed).

### Features

Inspired by human expert tricks for language ID, we extract a small set of features:

1. Index of coincidence: chance that two randomly-picked character tokens are identical. The index of coincidence was invented by the US Army cryptographer, William F. Friedman (Dooley, 2013), and is defined as:

$$Index\ of\ Coincidence\ (IC) = \frac{\sum_{i=A}^{Z} F_i(F_i - 1)}{N(N-1)} \tag{3.4}$$

where $F_i$ is the frequency of letter $i$ and $N$ is the total number of characters in the cipher.

2. Properties of unigram letter distribution:

   (a) Relative frequencies of the most frequent 5 characters

   (b) Frequency difference between the top-2 most frequent characters

3. Frequency of double letters.

4. Word length distribution: mean, median and standard deviation.

### Language ID results

We use Vowpal Wabbit (Langford et al., 2007) to train a linear classifier using stochastic gradient descent. We use a logistic loss function. Our best results use one-against-all classification, 30 passes over the training data and the default learning rate. Table 3.5 shows top-1 and top-3 accuracies for each cipher length in our dataset. Runtime is the total runtime for classifying the 100K test examples.

| | Length 32 | Length 64 | Length 128 | Length 256 | Length 512 | Runtime (s) |
|---|---|---|---|---|---|---|
| Spaced ciphers | 16% (36%) | 24% (49%) | 35% (64%) | 50% (80%) | 59% (87%) | 1.839s |
| No-space ciphers | 13% (32%) | 19% (42%) | 28% (57%) | 43% (75%) | 50% (84%) | 1.729s |

Table 3.5: Summary of language ID results on 100K synthetic ciphers (20K ciphers of each length) using n-way classification (where n = 16 candidate plaintext languages). Accuracy reported as: top-1 (top-3). Runtime is the total runtime for classifying the 100K test examples.

To investigate the effect of training data size, we experiment with training our classifier on different dataset sizes: 250K, 500K, 750K, and 1M training examples. Figure 3.6 shows the learning curve for our classifier. Accuracy is the overall classification accuracy for all cipher lengths.

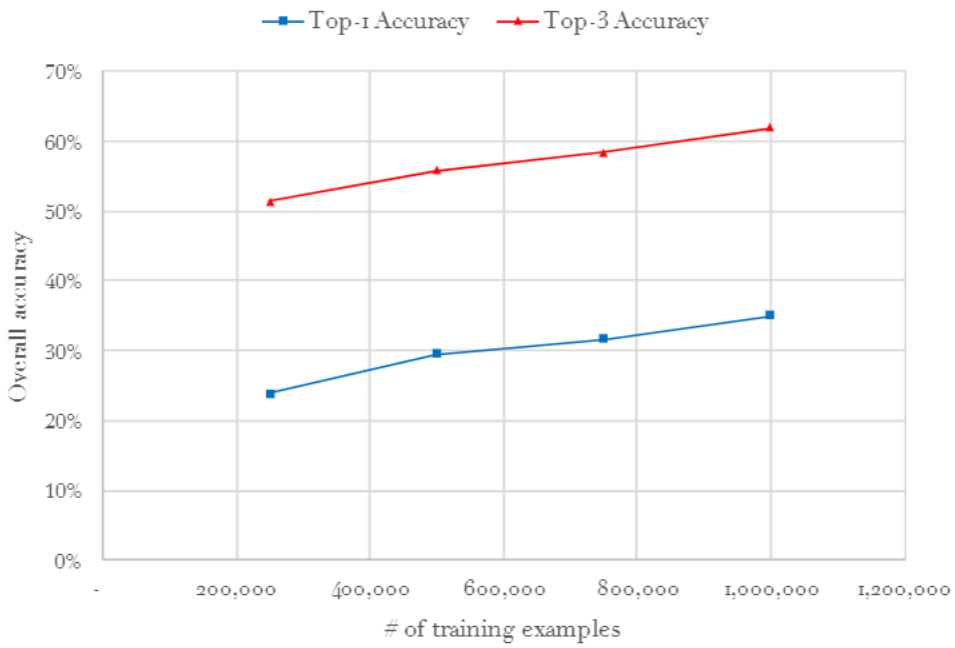



Figure 3.6: Learning curve of our language ID classifier. Accuracy is the overall classification accuracy for all cipher lengths in our 100K test set.

## 3.5 Decipherment results

We use the decipherment method described in section 3.3 to crack the nine synthetic ciphers in our test set. Table 3.6 shows a summary of our decipherment results. % Error is the percentage of character mistakes in the final decoded message compared to the gold answer.

| No. | Type | # chars | Spaces? | Language | LM n-gram | Restarts | % Error |
|---|---|---|---|---|---|---|---|
| 1 | 1:1 substitution | 353 | yes | English | 3 | 10 | 1.98% |
| 2 | 1:1 substitution | 150 | yes | English | 3 | 25 | 4.67% |
| 3 | 1:1 substitution | 653 | no | English | 4 | 20 | 1.53% |
| 4 | 1:1 substitution | 128 | yes | Spanish | 4 | 20 | 3.91% |
| 5 | 1:1 substitution | 107 | yes | German | 5 | 1 | 0.00% |
| 6 | 1:1 substitution | 331 | no | Swedish | 5 | 1 | 0.60% |
| 7 | 1:1 substitution | 168 | no | Portuguese | 5 | 1 | 1.80% |
| 8 | homophonic | 2376 | no | Latin | 5 | 1 | 2.14% |
| 9 | homophonic | 436 | no | Latin | 5 | 2 | 6.19% |

Table 3.6: Summary of decipherment results on our nine synthetic ciphers. % Error is the percentage of character mistakes in the final decoded message compared to the gold answer.

# Chapter 4

# Deciphering Historical Manuscripts

Real ciphers pose great challenges for automatic decipherment. The challenges begin with turning those ciphers into computer-readable format; a process usually referred to as *transcription*. To transcribe a historical document, a transcriber has to first recognize the cipher alphabet and decide on character boundaries, which is sometimes hard to do, especially if the cipher uses unfamiliar symbols. Then the transcriber should come up with an easy-to-type, easy-to-remember transcription scheme for faster and more accurate transcription. Moreover, the transcriber usually faces the challenges of degraded manuscripts, misshapen pages, ink blotches, and low-quality scans, which make the transcription process even more challenging.

After transcribing the document, we can proceed to decipherment. This chapter describes the application of our previously described techniques for language ID and decipherment on three real ciphers; the Zodiac-408 cipher, the"Borg" cipher, and the Oak Island cipher.

## 4.1 The Zodiac-408 cipher

The Zodiac Killer was a serial killer in northern California in the late 1960s and early 1970s. In 1969, the killer sent out three letters to the Vallejo Times Herald, the San Francisco Chronicle, and The San Francisco Examiner and demanded

they be printed on each paper's front page or he would kill a dozen people over the weekend. Each letter included a third of a 408-symbol cryptogram (shown in figure 4.1). The cipher was solved manually by Donald and Bettye Harden one week after its release. The killer also sent out another similar looking 340-symbol cipher, which has not been solved yet. The identity of the Zodiac Killer remains unknown to date, which makes his ciphers even more intriguing.

### 4.1.1 Decipherment

The Zodiac-408 cipher is one of the most famous homophonic ciphers in history. Besides the difficulty with missing word boundaries, the cipher contains spelling mistakes, which make the decipherment even harder. For example, the English word "paradise" is misspelt as "paradice" and the word "forest" is misspelt as "forrest." This could confuse word-based language models and dictionary-based attacks.

We used a 5-gram English letter language model to train a fully connected FST using EM. We used the trained channel model to get the Viterbi decoding of the ciphertext (as described in chapter 3). The decoded English script that we got reads (spaces inserted automatically):

> z like killing people because it is sq much junitia more fun than killing wild game in the for dest because manzs the most dangerque an amal of all to kill something gives my the moat thrilling expedence it is even better than getting your docks off with a girl the best part of it is that when i dies will be deborn in paradice and all they have killed will become my slaves i will not give you my name because you will try to sloz downodxt qpmy collecting of slaves for my after life

Figure 4.1: The Zodiac-408 Cipher.

This gives us an accuracy of 95.13% compared to the gold solution:

> i like killing people because it is so much fun it is more fun than killing wild game in the forrest because man is the most dangeroue animal of all to kill something gives me the most thrilling experence it is even better than getting your rocks off with a girl the best part of it is that when i die i will be reborn in paradice and all the i have killed will

become my slaves i will not give you my name because you will try to sloi down or stop my collecting of slaves for my afterlife

## 4.2 The "Borg" cipher

The "Borg" cipher is a 400-page book digitized by the Biblioteca Apostolica Vaticana. The official name of the cipher is "Borg.lat.898." We call it the "Borg" cipher for simplicity. It is believed to date back to the 1600s. The first page of the book seems to be written in Arabic. The rest of the book is completely enciphered in astrological symbols. The book also contains some Latin fragments, and a page of Italian right at the end of the book. Figure 4.2 shows two pages of the book. The Vatican does not have a key associated with this cipher nor any deciphered parts of the book.

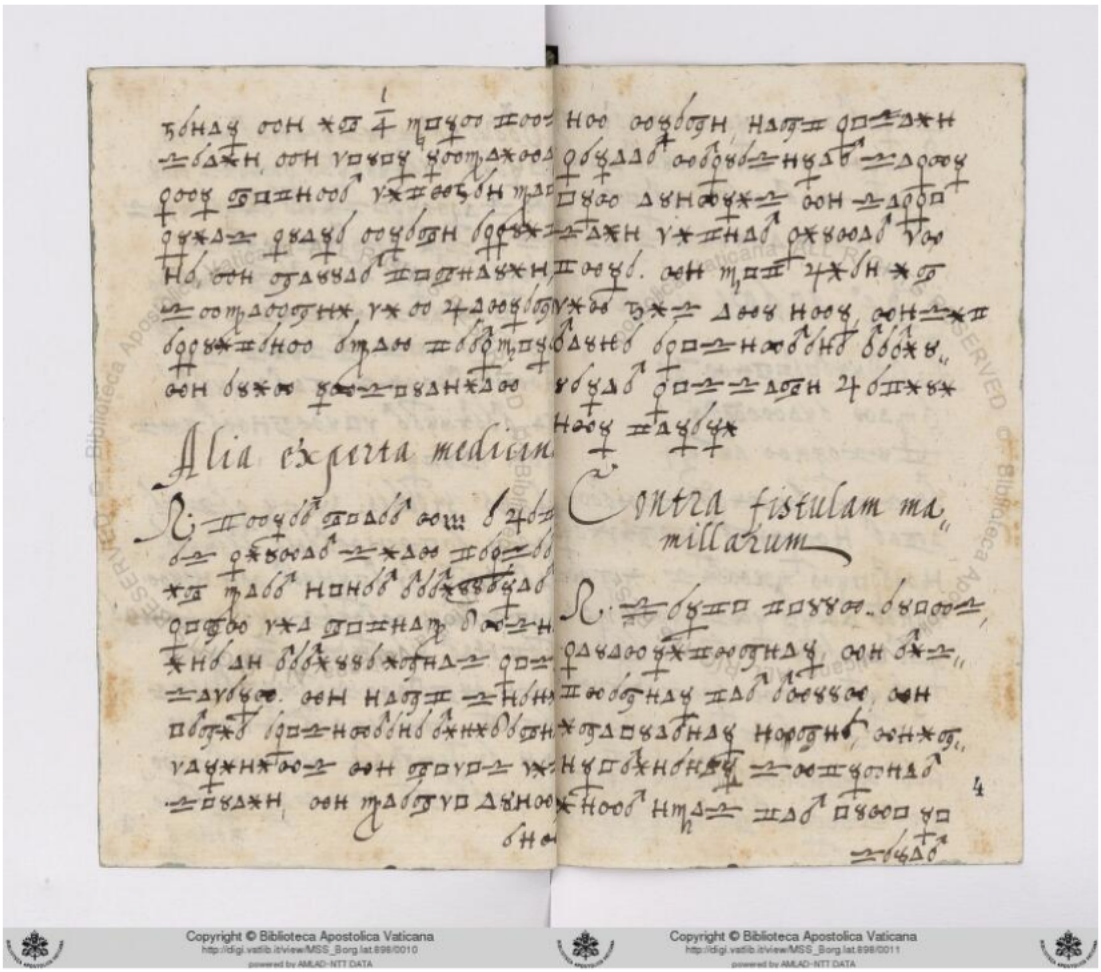



Figure 4.2: The "Borg" Cipher.[1]

### 4.2.1 Transcription

Transcribing the "Borg" cipher is more challenging than the Zodiac-408 cipher. First, the book is old, and it has obvious signs of degradation, which makes it hard to read some characters. Figure 4.3 shows a sample page with background noise and ink blotches. Another challenge is that the characters at the book binding

[1]Images retrieved from the Digital Vatican Library's official website: `http://digi.vatlib.it/view/MSS_Borg.lat.898`

area are cut off in the scan. This leaves us with a lot of incomplete words in the middle of the text. Moreover, it is not very clear whether some symbols represent one or more cipher characters. For example, the symbol shown in figure 4.4 looks like a "6" and a "9," but could also be one symbol (the Zodiac sign for Cancer). It turned out to be the latter as we will see in the following sections. Such findings can only be assured after deciphering the manuscript.

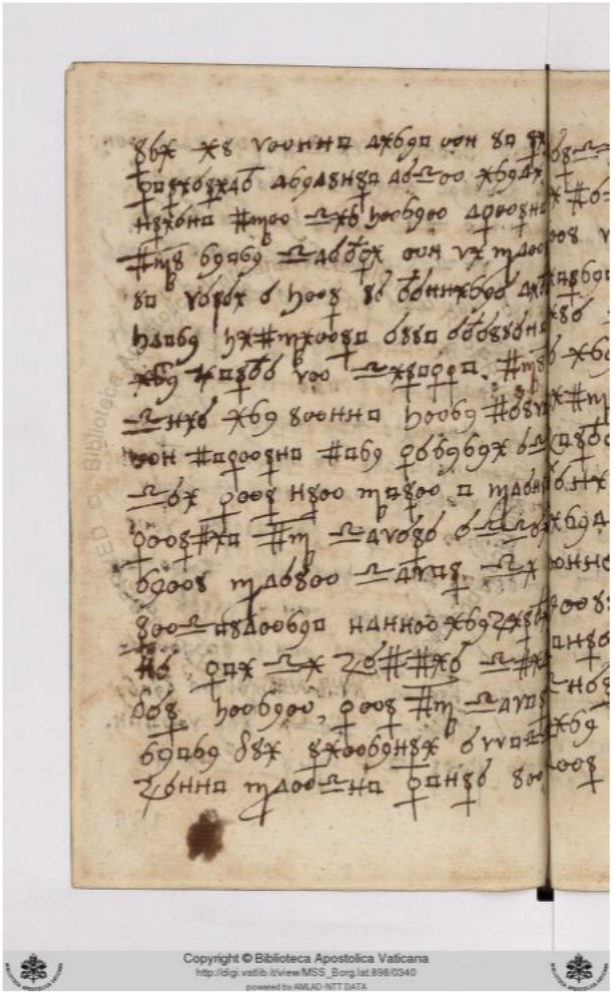



Figure 4.3: Page 0166v of the "Borg" Cipher. The image shows signs of degradation of the manuscript.

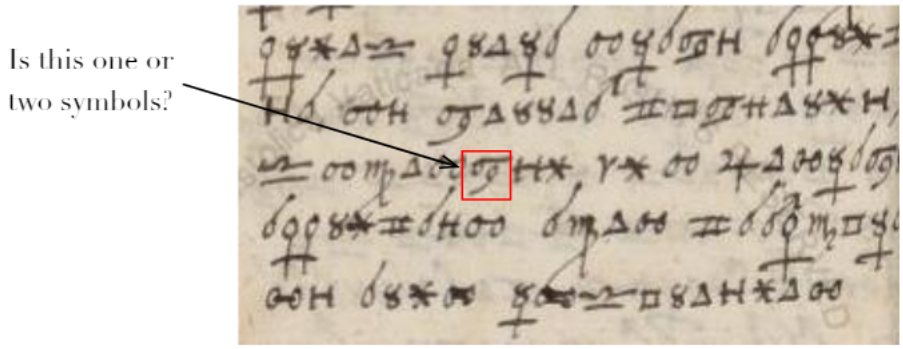
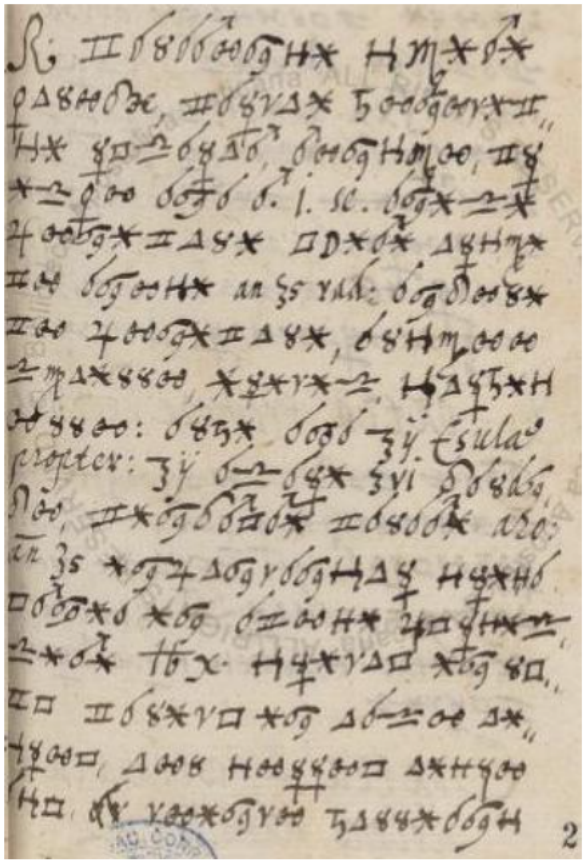



Figure 4.4: An excerpt of the "Borg" Cipher showing a confusing symbol for transcription.

We manually transcribed the first three pages of the cipher. 22 of the 26 cipher letter types appear in the first three pages of the book. Figure 4.5 shows page 0002r of the cipher, along with our transcription.

### 4.2.2 Language ID

Unlike the Zodiac-408 cipher, which we knew was English, we do not know the language of "Borg." So, we first need to identify the plaintext language. We ran EM partial decipherment against 16 European languages on the first three pages of the book. Table 4.1 shows the perplexity scores for the top five candidate plaintext languages. The results we got suggested Latin as the plaintext language of the cipher.



R i6861w9hx hmx1x
0d8wcx, i6qvdx 5w9wvxi,,
hx qon6qd1 1w9hmw iq,,
xn0w 696 16. I [se.] 69xnx
4w9xid8x obx1x dqhmx
iw 69whx [an] Z I [vad] 69cw8x,,
iw 4w9xid8x 68hmww
nmdx88w xqxvxn hdq5xh
w88w : 685x 696 Z Y [Esulam
propter :] Z Y 6n6qx Z [vi] c6869,,
cw ix961o1x i6861x [aro]
[: an] Z [5] x94d9v69hdq hqxh6
o19x6 x9 6iwhx 4oqhxn,,
nx1x hx hqxvdo x9 8o,,
io i68xvo x9 d6nw dx,,
hqwo dw8 hwqqwo dxhqw,,
6ho vwx9vw 5d88x69h

Figure 4.5: Transcription of the first page of "Borg." Square brackets are used to indicate what seems to be cleartext.

| Language | Per-Character Perplexity |
|---|---|
| Latin | 1.1323 |
| Esperanto | 1.1478 |
| English | 1.1534 |
| Hungarian | 1.1823 |
| Icelandic | 1.2170 |

Table 4.1: Top-5 languages according to perplexity scores from the partial decipherment of "Borg."

### 4.2.3 Decipherment

We used a 5-gram Latin letter language model to train a fully connected FST (channel model) using EM. We used the trained channel model to get the Viterbi decoding of the first three pages of the book. The decoded Latin script that we got reads:

> calamenti thimi pulegi cardui benedicti rosarum menthe cr ispe anam anisi feniculi obimi urthi ce aneti angeli ce feniculi althee shuille iridis turbit elle albi ana asali galange cinamomi calami infundantur trita omnia in aceti fortissimi ti triduo in loco calido in uase uitreo uel terreo uitre to deinde bulliant
>
> in uase fictili uitre ato ad casum medietatis fieri coleture adde sachari mellis despum? ti fiatserupus hui arpmatibetur cum cr cimacis cinamomi bin miberis suspe datur in saculo intus et seruetur usui nos sumus experti h? si tollatur puluis rubeorum et croci et uino bibatur subtili statim tollit tremore cordis
>
> magnum secretum indolorem mamillarum pellistalpe superposita milabilis est si permiseris talpam mori in manu tene do oculos irsius con tra radios solis si tetigeris cumilla manu mamillam dolentem messat dolor uxor passa est apostemata mamilla rum ush ad mortem et tale adposuit emplastrm tactum lacte rani huod est pingue supernatans lacti posthuam stetelit ad tempus et cum creta communi et superpone...

### 4.2.4 Translation

To get a quick translation of the Latin text, we used the online machine translation system; Google Translate. The first three pages of the book translate to:

calamenti thimi pulegi artichoke blessed roses menthe cr ispe anam anise fennel, dill angels engage in further ce ce fennel Althea shuille elle white iris turbit ana Asali galange cinnamon infused branches After all these in the three days in a warm place in a dish of vinegar has been very strong and ti glass or glass to frighten and then boil

in an earthen vessel fired a given half a chance to add to the coleture Zechariah honey despair? ti fiatserupus hui arpmatibetur with cr cimacis cinnamon bin miber suspe is kept in a purse inside and we use We experienced h? If we take away the dust of red and saffron and wine drinking immediately takes a subtle trembling heart

big secret absence of breasts pellistalpe spread milabilis if you allow it to die in the hands of a mole do hold up irsi con tra rays of the sun if you touch with Him the painful breast messar pain she suffered breast abscesses rum ush to death thus giving a plaster floating touch milk fat is a veteran Huod milk Posthumus stetelit common with chalk at a time, and on top...

This initial translation of the text seemed to suggest that this was a medical book from the early modern period. We could see some sentences that describe recipies like "all these in the three days in a warm place in a dish of vinegar."

With the help of our Swedish collaborator, Beáta Megyesi, we consulted Urban Örneholm, an expert in pharmacological and medical interpretations of Latin from the 16th-17th centuries. He assured us that this was readable Latin and gave us a refined translation of the first three pages:

Take lesser calamint, thyme, pennyroyal, St. Benedict's thistle, roses and wrinkled-leaf mint, one handful of each; aniseed, seed of fennel, basil, nettle and dill, half a drachm of each; roots of angelica, fennel,

marsh mallow, sea squill, turbith and white hellebore, two ounces of each; of green spurge <propter> two ounces, of hazelwort six drachms; galangal, cinnamon and calamus, half a drachm of each. Everything is grated, and left for three days, on a warm place, in ten pounds of strong vinegar, in a vessel of glass or glazed earthenware. Then, you boil it

down in a glazed earthenware pot, to half its volume. Strain; add sugar and despumated honey, twenty ounces of each; this should become a syrup, which is spiced with saffron, mace, cinnamon and ginger, two drachms of each, which is suspended in a small bag inside the vessel. It is saved for future needs.

For trembling of the heart

We have noticed that if you take a powder of red coral and saffron, and this is drunk in a fine wine, it will immediately stop trembling of the heart.

For boils and pains of the breasts

An important secret (in this context = panacea) for breast pains: application of mole skin is marvellous; if you have let the mole die while holding it in your hand with its eyes towards the sun, and with that same hand covered the ailing breast, the pain goes away. The wife of Hans Stoldis (?) suffered from boils in her breasts so badly, that she <almost> died, and he used a plaster made from <lac ran> - that is fat that floats on the surface of milk which has been left for some time – and common chalk; this was applied...

### 4.2.5 The Key

The key that we got from the automatic decipherment was almost perfect. There was one character that seldom appeared in the first pages of the book, so it was hard for the machine to decide on how to decipher it. Örneholm also pointed out another interesting feature of the text, which is the use of abbreviations for measurements. For example, floreni is a measurement for the gold florin. The word "ana" means "of each" in recipies. Table 4.2 shows the key that we got from automatic decipherment, along with the abbreviations interpreted by Örneholm.

| Cipher Symbol | Transcription | Decoded Latin | Cipher Symbol | Transcription | Decoded Latin |
|---|---|---|---|---|---|
|  | a | V |  | 4 | F |
|  | b | Z |  | 5 | B |
|  | c | G |  | 6 | A |
|  | d | U |  | 8 | L |
|  | h | T |  | 9 | N |
|  | i | C |  | O | K |
|  | k | Y |  | M | Q |
|  | m | H |  | H | floreni |
|  | n | S |  | T | libram |
|  | o | O |  | W | suffix: -n or -m |
|  | q | R |  | Z | unica or drachmam |
|  | v | D |  | I | 1 |
|  | w | E |  | Y | II? |
|  | x | I |  | AU | ana |
|  | y | X |  | , | , |
|  | 0 | P |  | ,, | - |
|  | 1 | M |  | ? | ? |

Table 4.2: The transcription scheme and key of the "Borg" cipher.

### 4.2.6 A page in Arabic?

The script on first page of the book (shown in figure 4.6), though poorly written, looks like Arabic. However, the text does not seem to make any sense in Arabic. We asked native speakers of Persian and Turkish, and it did not make any sense to them as well. We then thought that it might be Latin written in Arabic script. The handwriting is not very clear, so it takes some effort to be able to recognize letters. Also, there is some ambiguity in the text since vowels are not written in the Arabic language (though the writer seems to use vowel marks). At that time, we were not sure whether vowel marks were used properly in the text. We transcribed the words to the best of our ability and sent our transcriptions, along with a scan of the page to Ambjörn Sjörs, a researcher at the Department of Linguistics and Philology at Uppsala University, who studied both Arabic and Latin. Sjörs assured us that the script on the first page is indeed Latin written in Arabic script. Table 4.3 shows Sjörs's interpretation of the first three lines of the script, which appears to be the title of the book.

| **Latin Word** | **Translation** |
|---|---|
| naturalis | natural |
| observationis | observations |
| illuminati | illustrations |

Table 4.3: Latin reading and English translation of the first three lines of the Arabic script at the beginning of "Borg."

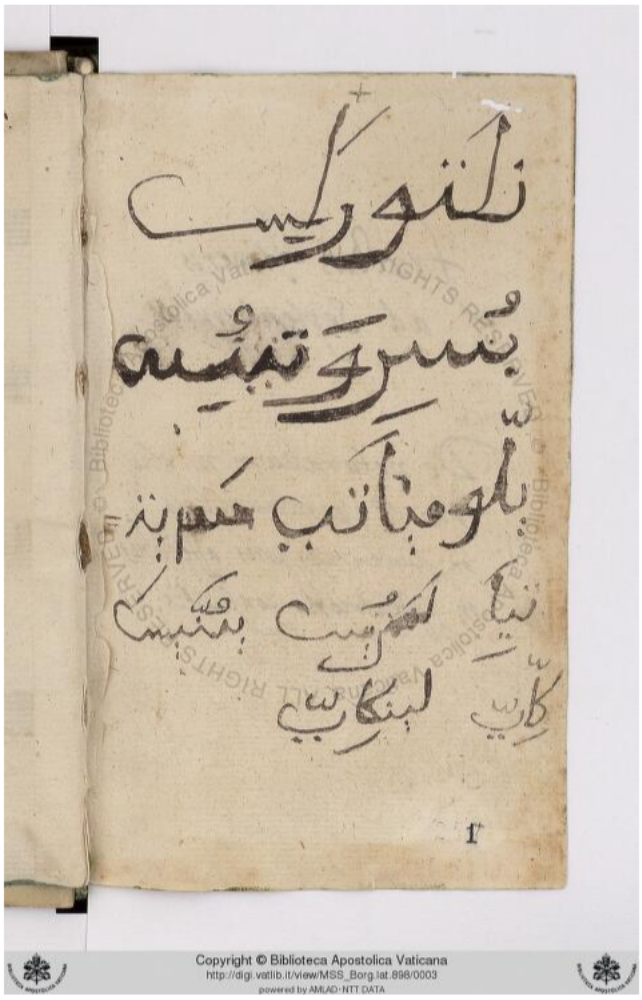



Figure 4.6: The first page of the "Borg" cipher.

### 4.2.7 What the book is about

Our Swedish collaborators manually transcribed more pages of "Borg." We now have about 100 transcribed pages of the book. Pages 0002r - 0027v discuss pharmacology, symptoms of ilness, and treatments for various diseases. Starting from page 0028r, the book discusses warefare with firebombs. Here is a translated excerpt from pages 0028r - 0029r:

Nectanebo says to Alexander: O, Alexander, may you be regarded as a virtuous king, and may you destroy your enemies with fire; I send you various kinds of fire to burn your enemies, whether on land or at sea.

The first kind of fire

Take 1 pound of the purest sandarac, or vernix, one pound <[..]rmo>, liquid, and after pestling, put it in a glazed earthenware pot, and seal with lute, then

[..] fire until liquid, [..] this liquid and an equal amount of [..] the sign of liquefaction is, that on a wooden stick, inserted through the opening, the matter should resemble butter. Then, you pour it over an equal amount of greek [..] (this sign should mean 'tar' or 'pitch') or colophonium. This may not be done indoors, because of the danger of fire. When you want to use it, however, take a bag of goat skin, which you inflate, and smear with said oil in- and outside, and tie the bag to a spear, and place on that a piece of wood. The iron should touch the bag, where the wood, when ignited, [..] the said preparation

set on fire, and falls down over the sea, and by the wind is carried towards the enemies, and burns them, and water cannot extinguish this. The second kind of fire is this: Take one scruple of balsam, one pound of the pith of ferula cane, one scruple of sulphur, one scruple of liquefied duck fat, and mix at the same time, and apply on an artfully made arrow; when this has been ignited, shoot the arrow towards the mountains; and the places where it has fallen, this concoction will set on fire, and water can not extinguish it. The third kind of fire...

We show more pages and translations of the "Borg" cipher in appendix B.

## 4.3 The Oak Island cipher

In July, 2016, we were contacted by Jason Shook, the producer of "The Curse of Oak Island," a Canadian reality TV show broadcast on the History Channel since 2014. The show details the efforts of Marty and Rick Lagina in solving the 220-year-old mystery of Oak Island, a small island on the south shore of Nova Scotia, Canada. The island is best known for theories about a possible buried treasure and many associated explorations. Shook sent us a scan of a new cipher they found and had been trying to solve. Figure 4.7 shows the cipher we received from Shook.

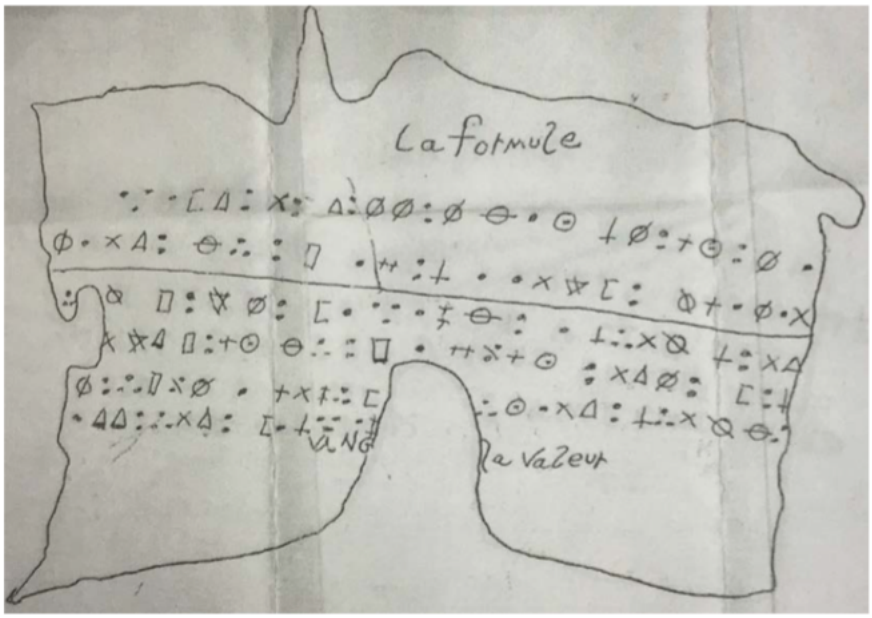
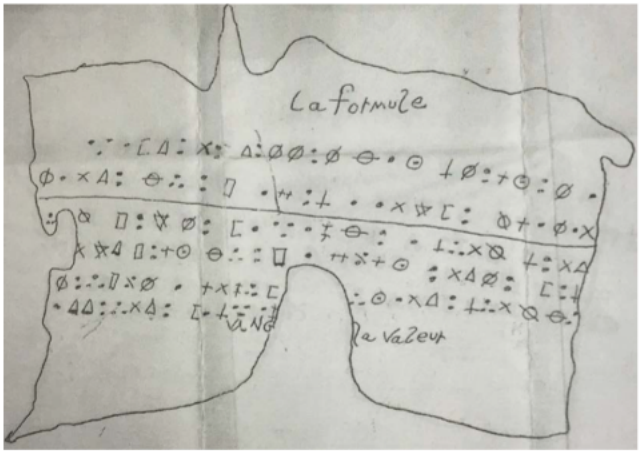



Figure 4.7: A scan of the Oak Island cipher.

### 4.3.1 Transcription

The cipher is very short. So, we manually transcribed the whole cipher. Figure 4.8 shows our transcription of the Oak Island cipher.




NBIE ADA EACCAC JBK GCAHKAC B...
...CBDEA JFAL BQAG B BDMIA PHBCBD...
...N...C LAMCA IB NBOJA B GFDP GADE...
...DME LAHK JFAL B QRHK ADECA IAG...
...CAFL RC B HD OFI...FKBDEA GFDP JF...
...BEEAFDEA IBGNBO...

Figure 4.8: Transcription of the Oak Island cipher.

### 4.3.2 Language ID

We ran EM partial decipherment against 16 European languages on the cipher. Table 4.4 shows the perplexity scores for the top five candidate plaintext languages. The results we got suggested French as the plaintext language of the cipher.

| Language | Per-Character Perplexity |
|---|---|
| French | 1.0222 |
| Esperanto | 1.0870 |
| English | 1.1006 |
| Hungarian | 1.1142 |
| Icelandic | 1.1175 |

Table 4.4: Top-5 languages according to perplexity scores from the partial decipherment of the Oak Island cipher.

### 4.3.3 Decipherment

We used a 5-gram French letter language model to train a fully connected FST using EM. We used the trained channel model to get the Viterbi decoding of the ciphertext. The decoded French script that we got reads:

halt ene terrer pas creuser a...
...rante pied avec a angle quaran...
...h...r degre la hampe a cinq cent...
...ngt deus pied a vous entre lec...
...reid or a un mil...isante cinq pi...
...atteinte lacham...

Then we realized that we had better take the shape of the paper into account, where some words might be cut off. Figure 4.9 shows Kevin Knight's drawing of the deciphered text that we got, with first guesses on incomplete words.

The word "deus" looks like a misspelt "deux." Also, the last cut off word could be "la chambre," which is French for "the room" or "the chamber."

### 4.3.4 Translation

We used Google Translate to get a quick translation of the French text. The original decoded message translates to:

ene halt terrer not dig a ...
... Appli- foot with a quarantinable angle ...
... H ... R degree the pole five hundred ...
... Ngt deus foot between you lec ...
... Reid or a thousand ... Cient five pi ...
... Lacham reached ...

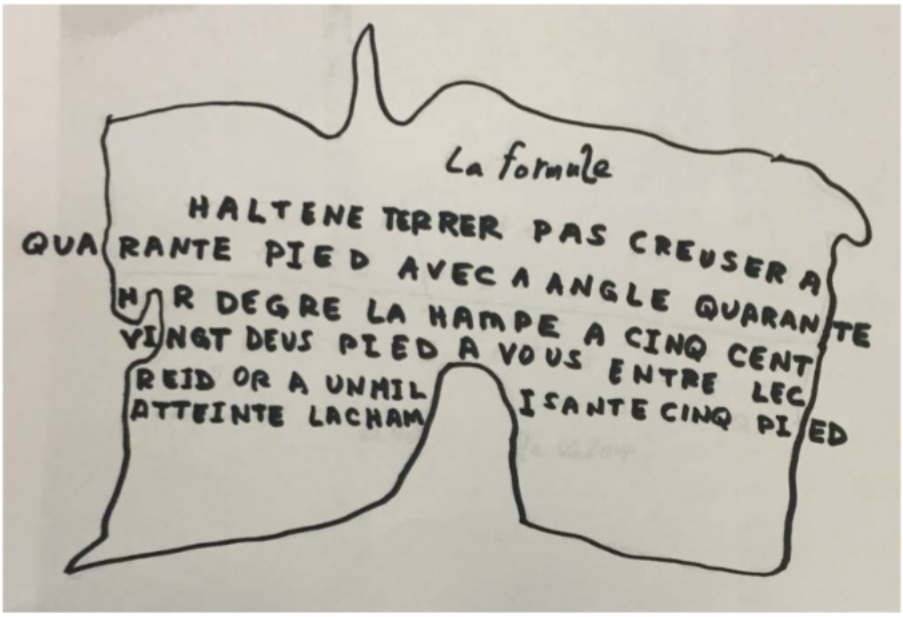



Figure 4.9: Kevin Knight's drawing of the deciphered Oak Island cipher, with attempts to figure out the incomplete words.

After adding guesses about incomplete words, we could get a refined translation from Google Translate:

> ene halt terrer not dig a ...
>
> ... Forty forty foot with angle ...
>
> ... H ... R degree the pole five hundred ...
>
> ... Twenty two foot between you lec ...
>
> ... Reid or a thousand ... Sufficient five foot ...
>
> ... The room reached ...

Shook sent our decipherment to Luckas Cardona, who gave us a refined translation of the French message (some words were not translated and kept as deciphered):

> HALT/DO NOT HOLE UP DIG AT

FORTY FEET WITH A FORTY N R DEGREE
ANGLE THE SHAFT IS FIVE HUNDRED
TWENTY-TWO FEET TO YOU ENTER THE/C...
...REID OR A UNMIL...ISANTE FIVE FEET
REACH THE FIELD/ROOM/CHAMBER

### 4.3.5 The Key

The key that we got from the automatic decipherment was almost perfect. Table 4.5 shows the key that we got from automatic decipherment, along with our transcription scheme.

| Cipher Symbol | Transcription | Decoded French |
|---|---|---|
|  | B | A |
|  | G | C |
|  | L | D |
|  | A | E |
|  | M | G |
|  | N | H |
|  | F | I |
|  | I | L |
|  | O | M |
|  | D | N |
|  | R | O |
|  | J | P |
|  | P | Q |
|  | C | R |
|  | K | S |
|  | E | T |
|  | H | U |
|  | Q | V |

Table 4.5: The transcription scheme and key of the Oak Island cipher.

# Chapter 5

# Deciphering from Images

So far, we have been focusing on deciphering manually transcribed historical manuscripts. In this chapter, we discuss decipherment from images. We describe our models, experiments, and results on different ciphers.

## 5.1 OCR challenges

As we have discussed in chapter 4, the transcription process is very challenging for humans, and it is even more challenging for computers. One challenge we face with ciphers is that they are usually written in an unknown alphabet or in symbols, as opposed to known alphabet like documents written in English or Arabic, for example. It is thus unclear what the end result of a transcription should be. Secondly, we are targeting handwritten documents, not typed historical documents. There are many more irregularities and variance in handwriting styles in handwritten documents than in typed ones. In addition, characters are usually not perfectly aligned as they are in typed text. Characters also have variable sizes throughout handwritten documents, which poses a big challenge for automatic character segmentation. Figure 5.1 shows two pages from the "Borg" cipher. As the images show, many characters touch and have variable sizes and shapes. Also, there are signs of document degradation and ink blotches, which make image processing even harder.

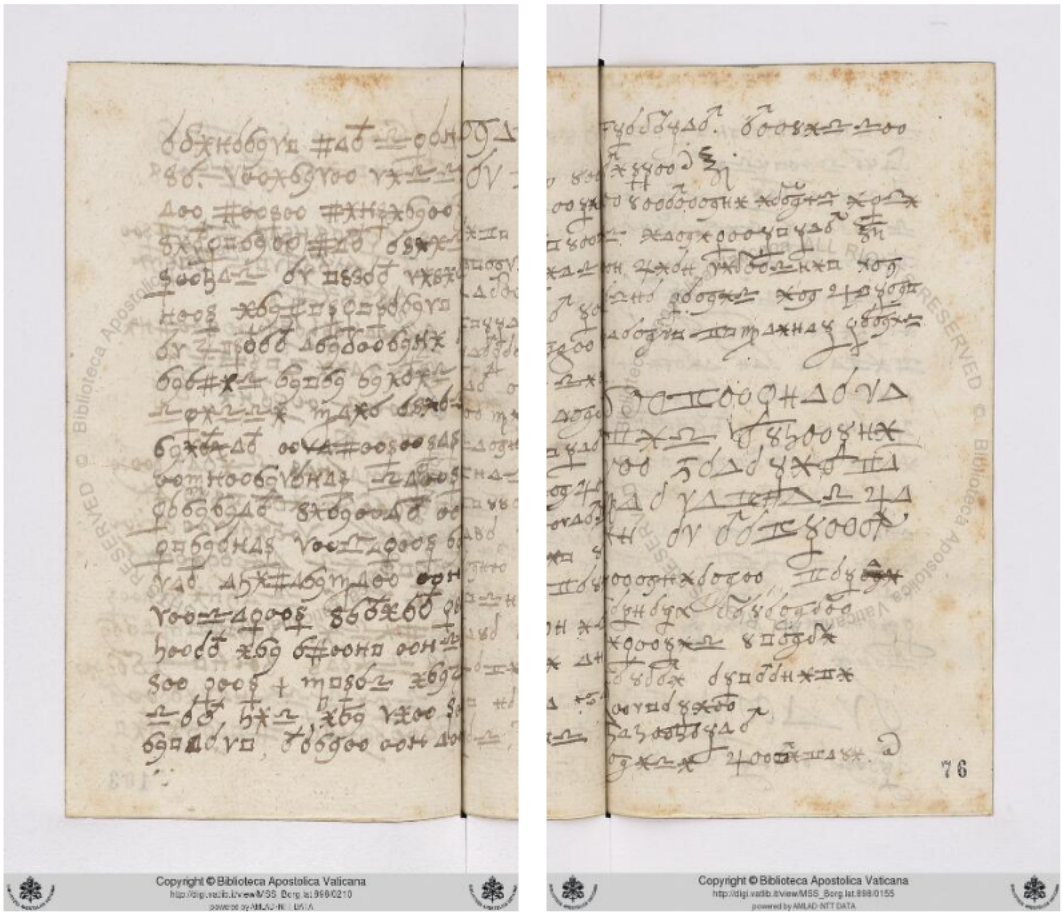



Figure 5.1: Two pages from the "Borg" cipher. Images show many challenges for OCR, like different handwriting styles, scratches, variable character sizes, and background noise.

## 5.2 OCR model

We experiment with an unsupervised model for deciphering from images. To decipher a cipher image, we take the following steps:

1. Character segmentation: clip the original image into smaller images of single characters.

2. Character clustering: cluster the segmented character images based on shape similarity. Then, substitute each character image with its cluster ID.

3. Decipherment: Decipher the sequence of cluster IDs using the decipherment methods described in chapter 3.

The next two sections describe steps 1 and 2. We will use the first pages of the "Borg" cipher to illustrate the results of each step.

## 5.3 Character segmentation

The goal here is to clip the original image into smaller images of single characters. To find individual characters, we create a generative story for how the image was generated. We represent the image by the number of black pixels in each row/column. We create two stories; one for generating rows and the other for generating characters in each row.

For row separation, our goal is to find *separator rows* that cut through the smallest number of black pixels. We represent an image by a sequence of integers, one for each row, representing how many black pixels are on that row. Now, we are trying to explain that sequence. So, we create this generative story:

**Parameters:** mean, stdev, stdev2, p.

1. Pick the number of rows $n$ according to a normal distribution $N(mean, stdev)$.

2. for $i = 1$ to $n$:

(a) Pick a height $h_i$ for row $i$ according to another normal distribution $H(mean2, stdev2)$. Note that:

$$mean2 = \frac{total\ number\ of\ pixel\ rows}{number\ of\ rows}$$

(b) for $j = 1$ to $h_i$:

Output an integer according to a uniform distribution.

(c) Output an integer according to a geometric distribution $G(p)$.

The integer output in $2(c)$ is the separator row. No matter how $p$ is set, step $2(c)$ will prefer to output a small number than a large number (i.e. minimize the number of black pixels in each separator row).

We create a similar story for segmenting characters in each row. For character separation, our goal is to find *separator columns* that cut through the smallest number of black pixels. We represent a row image by a sequence of integers, one for each column, representing how many black pixels are on that column. Now, we are trying to explain that sequence. So, we create this generative story:

**Parameters:** mean, stdev, stdev2, p.

1. Pick the number of characters $n$ according to a normal distribution $N(mean, stdev)$.

2. for $i = 1$ to $n$:

   (a) Pick a width $w_i$ for character $i$ according to another normal distribution $H(mean2, stdev2)$. Note that:

$$mean2 = \frac{total\ number\ of\ columns}{number\ of\ characters}$$

(b) for $j = 1$ to $w_i$:

Output an integer according to a uniform distribution.

(c) Output an integer according to a geometric distribution $G(p)$.

The integer output in $2(c)$ is the separator column. No matter how $p$ is set, step $2(c)$ will prefer to output a small number than a large number (i.e. minimize the number of black pixels in each separator column).

We manually set the values for these parameters. We implement our generative story as a composition of a finite-state-acceptor (FSA) and a finite-state-transducer (FST). Figures 5.2 and 5.3 show our FSA and FST for the character generation story. We use the finite-state toolkit, *Carmel*, to determine the Viterbi state sequence of maximum probability (Graehl, 2010). From that Viterbi state sequence, we can see which rows are separator rows. Figure 5.4 shows character segmentation results for the first page of "Borg."

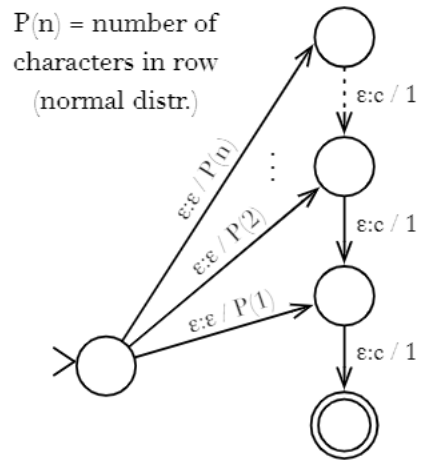



Figure 5.2: An FSA for generating characters in a row. We generate n characters with probability $P(n)$ (normal distr.).

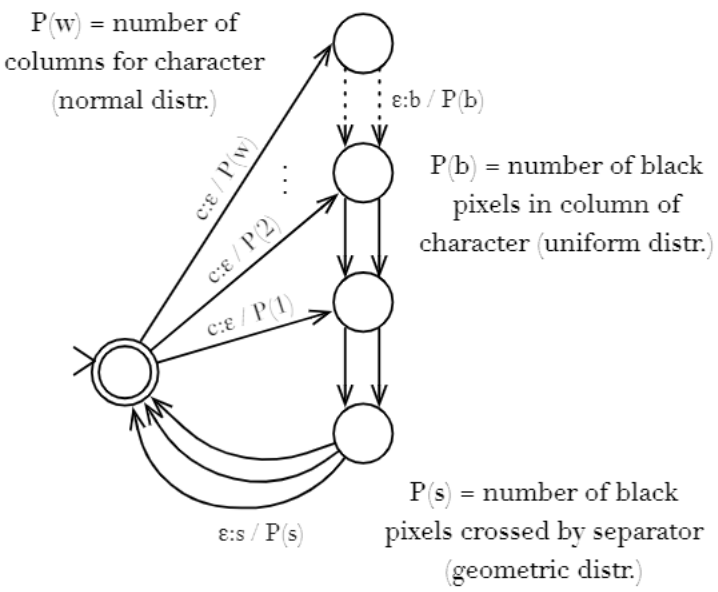



Figure 5.3: An FST for generating the number of black pixels in each column. For each character, we generate character columns (normal distr.), followed by a separator column (geometric distr.).

## 5.4 Character clustering

To cluster the segmented character images, we take the following steps:

1. Compute pairwise similarity among character images: We compute the pairwise similarity matrix using the signal tool, *correlate2d* from *Scipy* (Jones et al., 2001).

2. Run K-means clustering on the similarity matrix: We use the *KMeans* implementation from *Scikit* (Pedregosa et al., 2011). The package provides the *fit-predict()* method, which computes cluster centers and predicts the closest cluster index for each sample.

Table 5.1 shows seven randomly selected clusters we get from clustering the first three pages of "Borg" (with K=26). Note that we get some clean clusters

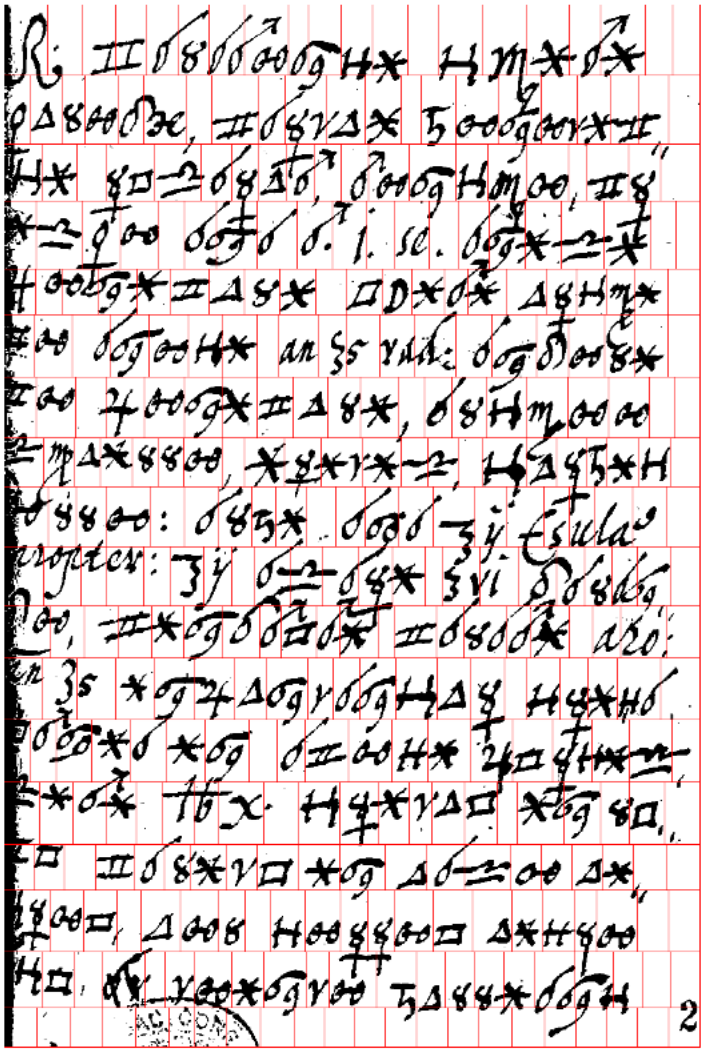


Figure 5.4: Character segmentation results for the first page of "Borg."

(like clusters 14 and 19), but we also got noisy clusters (like clusters 2 and 7). Many of these clustering errors are due to clipping errors from the challenging character segmentation step. In addition, some characters are very rare, and thus, get assigned to one of the larger character clusters. Other factors like size and inking level also seem to affect cluster assignments for characters with the same shape.

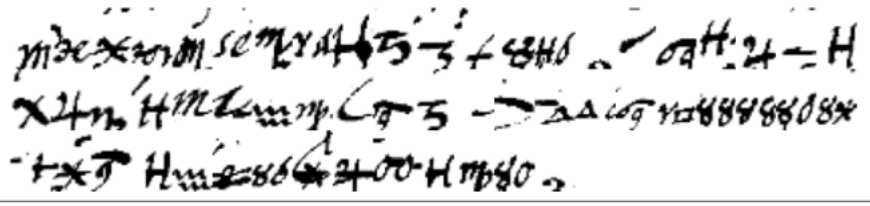
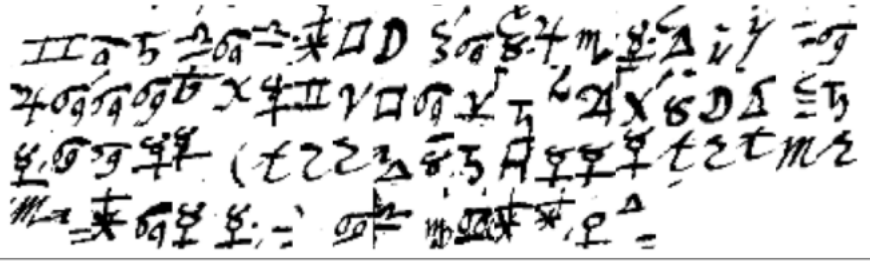
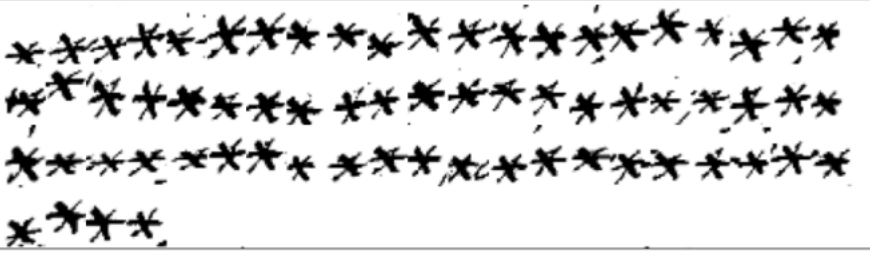
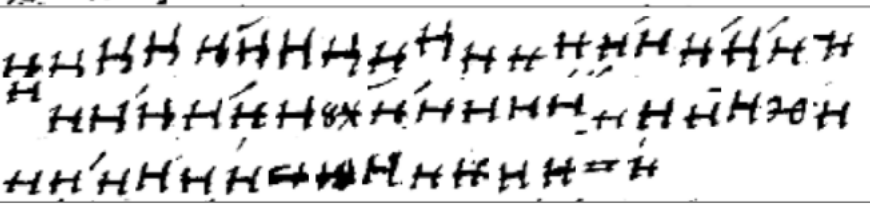
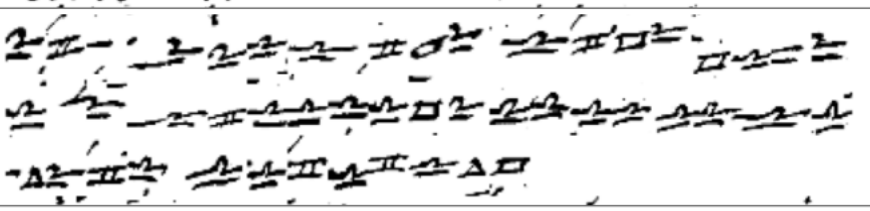
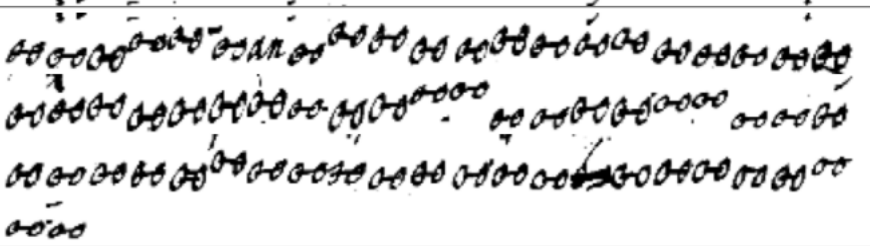
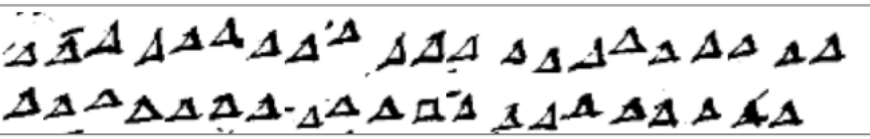
| Cluster ID | Character Images |
|---|---|
| 2 |  |
| 7 |  |
| 14 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |

Table 5.1: Seven randomly selected clusters we get from clustering the first three pages of "Borg" (with K=26).

## 5.5 Decipherment Results

After getting the cluster ID sequence, we can proceed to decipherment. We experiment with three ciphers: cipher#3 from our synthetic cipher collection (a 300dpi scanned image of the cipher printed in Courier font), the first three pages of the "Borg" cipher, and the Zodiac-408 cipher. Table 5.2 shows decipherment results from automatic transcription, compared to decipherment results from manual transcription (as we have previously shown in chapters 3 and 4). In this table, decipherment error is the edit distance between the gold string and the system output. To further investigate the effect of the character segmentation step, we also experiment with manually segmenting characters of the first three pages of "Borg." We run automatic clustering and decipherment on the manually segmented character images. The result is shown in the last row in table 5.2. The results indicate that automatic segmentation and clustering are very successful if the input is clean, but they are very challenging when the input is a noisy handwritten document. Character segmentation seems to be the major bottleneck of the system since we were able to decipher "Borg" with 80% accuracy from perfectly segmented character images. Note that deciphering from images hides word divisions and might also turn 1:1 substitution ciphers into homophonic ones as a result of over-clustering. This leads us to the discussion of how we can evaluate OCR output, which we discuss in the next section.

| | Decipherment Error from Automatic Transcription | Decipherment Error from Manual Transcription |
|---|---|---|
| Cipher#3 (typed) | 13 (1.99%) | 10 (1.53%) |
| Borg (3 pages) | 863 (72.83%) | 36 (4.14%) |
| Zodiac-408 | 313 (80.26%) | 19 (4.87%) |
| Borg (manual char seg.) | 227 (20.43%) | 36 (4.14%) |

Table 5.2: Summary of decipherment results from automatic vs. manual transcription. Decipherment error is the edit distance between the gold string and the system output. The last row shows the result of deciphering the first three pages of "Borg" from manually segmented character images.

## 5.6 Transcription error

So far, we have been using decipherment accuracy to evaluate our system. However, this measure only evaluates end-to-end decipherment results and does not give feedback on the performance of each part of the system (i.e. how much of the error we get is from OCR and how much is from decipherment). Thus, we need a method to evaluate automatic transcription accuracy. Our OCR system reads an image and outputs a sequence of cluster IDs, each representing one character. We want to evaluate our system output (automatic transcription) against the gold transcription (manual transcription). For example:

| | | |
|---|---|---|
| Gold: | t i m i | (manual transcription) |
| System output: | 4 2 2 | (cluster ID sequence) |

Since our system output does not use the same alphabet as the gold transcription, we cannot directly compute string edit distance. We first need to find a substitution scheme between our system output and the gold alphabet. We are looking for a special substitution scheme; the substitution that *minimizes* the edit distance between the two strings. In our previous example, we can substitute 4

with t and 2 with i. Making these substitutions gives an edit distance of 1. We call this substitution scheme an *optimal assignment* because it is the assignment that gives the minimum edit distance between the two strings (which is 1).

Inspired by the approach described by Spiegler and Monson (2010), we use ILP to compute transcription accuracy. Spiegler and Monson (2010) find the optimal assignment by global counting over <system output, gold> pairs (each representing the morphological analysis of a single word from a test set). In our case, we only have one pair of strings instead of a set of <system output, gold> pairs. So, we formulate our integer program as follows:

Given 2 strings:

| | | |
|---|---|---|
| Gold string: | $g_j : g_1\ g_2\ \ldots\ g_m$ | (manual transcription) |
| Cluster ID sequence: | $c_i : c_1\ c_2\ \ldots\ c_n$ | (system output) |

Find the edit distance between the two strings under the optimal assignment.[1]

**Variables (binary):**

| | |
|---|---|
| $ins_{i,j}$ | insert character $g_j$ after character $c_i$ |
| $del_{i,j}$ | delete character $c_i$ |
| $match_{i,j}$ | match characters $c_i$ and $g_j$ |
| $link_{c,g}$ | link characters $c$ and $g$ |

**maximize:**

$$\sum_i \sum_j match_{i,j}$$

[1]We will refer to $c_i$ and $g_j$ as string characters for simplicity, with the understanding that they could be two-digit cluster IDs, for example (actually, they could be drawn from any set of symbols that we come up with).

**subject to:**

$$\forall c : \sum_{g} link_{c,g} \leq 1 \tag{5.1}$$

$$\forall g : \sum_{c} link_{c,g} \leq 1 \tag{5.2}$$

$$\forall match_{i,j} : match_{i,j} \leq link_{c_i,g_i} \tag{5.3}$$

$$\forall (c_i, g_j) : ins_{i,j} + del_{i,j} + match_{i,j} = ins_{i,j+1} + del_{i+1,j} + match_{i+1,j+1} \tag{5.4}$$

$$match_{0,0} = 1 \tag{5.5}$$

Constraint 5.1 ensures that every cluster ID type is assigned to a maximum of one gold character type. Constraint 5.2 ensures that every gold character type is linked to a maximum of one cluster ID type. Constraint 5.3 ensures that we can only match two characters if there is a link between them (i.e. they can be substituted for each other under the optimal assignment). If we think of our search space as a grid of possible string edits (Figure 5.5), then we can think of computing edit distance as a network flow problem. We start with an initial flow value of 1 (enforced by constraint 5.5). Then we maintain a single sequence of string edits with constraint 5.4 (conservation of flow, i.e. the total flow entering a node $(c_i, g_j)$ must equal the total flow leaving $(c_i, g_j)$ for all nodes $(c_i, g_j)$).

Figure 5.5 shows how edit distance could be computed for our example strings using this integer program. Dotted lines represent the search space for our integer program. The solid path shows the sequence of string edits that we need to perform under the optimal assignments.

With this ILP formulation, we enforce the optimal assignment to be a 1:1 mapping between system output and gold characters. However, we can relax the constraints to allow for a M:1 mapping between system output and gold characters.

This is equivalent to turning a simple 1:1 substitution cipher to a homophonic cipher. Since our decipherment method targets both kinds of ciphers, we decide to allow M:1 mappings by removing the constraint given by 5.2. We use the Gurobi Optimizer to solve this integer program (Gurobi Optimization, 2016).

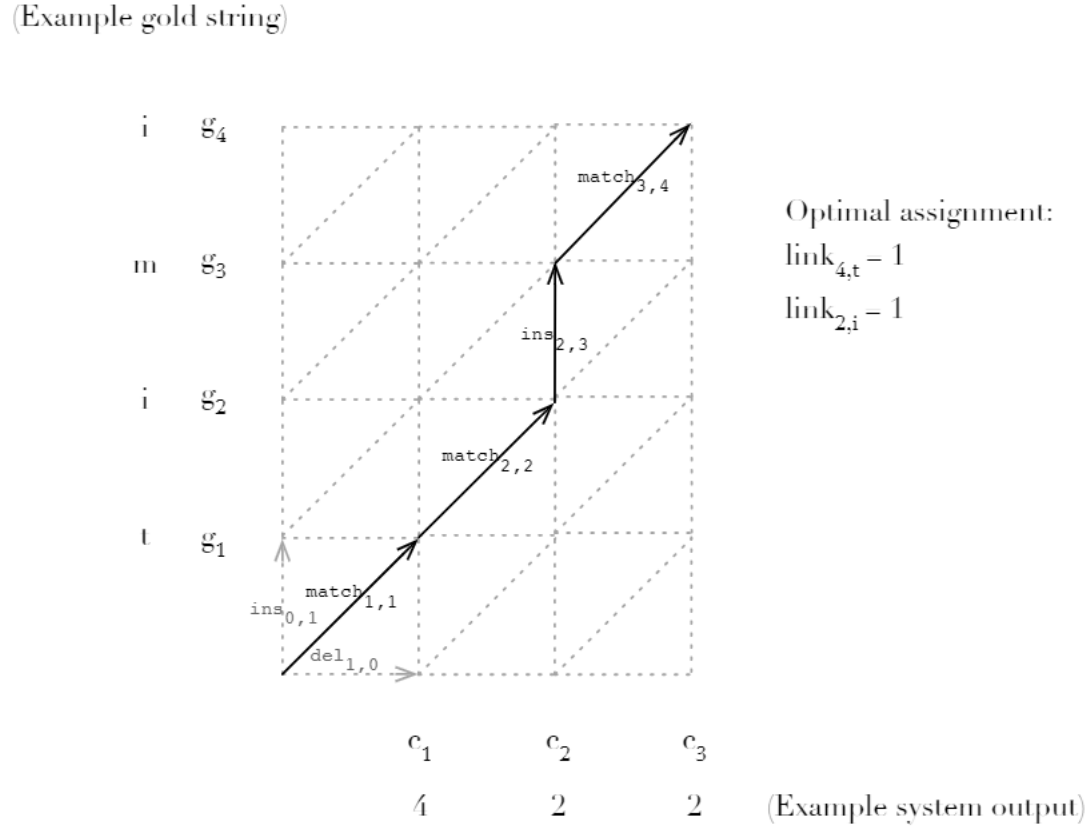



Figure 5.5: An example of how edit distance could be computed using our integer program. Dotted lines represent the search space for our integer program. The solid path shows the sequence of string edits that we need to perform under the optimal assignment.

One challenge we face with ILP is that it is very slow. Comparing two 80-character strings takes more than 24 hours on a 2.2 GHz Intel Core i7 with 16 GB RAM. Computing longer strings might not be possible with this method. To handle this, we make a slight modification to our integer program. Instead of

comparing the two strings all at once, we take the output of the OCR system and compare it to the gold transcription line-by-line. This reduces the number of variables and the search space (Figure 5.6). We still need to have a large number of variables to ensure consistent matching throughout the whole cipher, but that is a much smaller number of variables than what we need to compare the whole strings all at once.

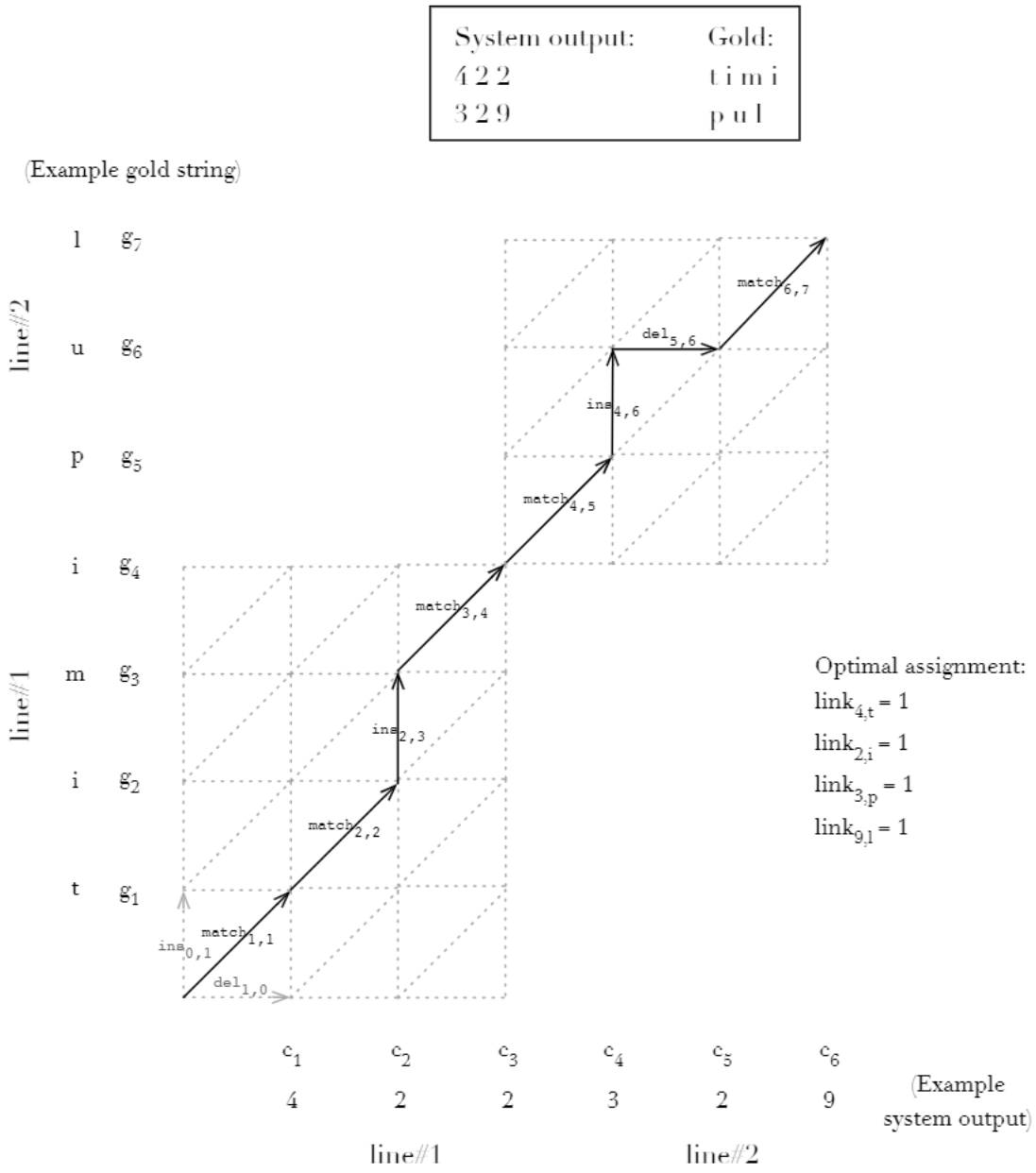



Figure 5.6: An example of computing edit distance line-by-line. Dotted lines represent the search space for our integer program. The solid path shows the sequence of string edits that we need to perform under the optimal assignment.

Table 5.3 shows decipherment results, with transcription error computation. Table 5.4 details the properties of the ciphers that we get from OCR, compared to the gold ciphers. Our integer program gives a quantitative measure if OCR

has introduced some homophonicity to the cipher and/or did not represent some cipher characters (according to the optimal M:1 mapping).

| | Transcription Error | Decipherment Error from Automatic Transcription | Decipherment Error from Manual Transcription |
|---|---|---|---|
| Cipher#3 (typed) | 37 (5.67%) | 13 (1.99%) | 10 (1.53%) |
| Borg (3 pages) | 57 (56.44%)[2] | 863 (72.83%) | 36 (4.14%) |
| Zodiac-408 | 71 (18.21%) | 313 (80.26%) | 19 (4.87%) |
| Borg (manual seg.) | 347 (28.61%) | 227 (20.43%) | 36 (4.14%) |

Table 5.3: Summary of decipherment results from automatic vs. manual transcription. Transcription error is the edit distance of the optimal assignment between system output and gold transcription. Decipherment error is the edit distance between the gold string and the system output. The last row shows the result of deciphering the first three pages of "Borg" from manually segmented character images.

| | Transcription Error | Introduced Homophonic Characters | Unrepresented Cipher Characters |
|---|---|---|---|
| Cipher#3 (typed) | 37 (5.67%) | 4 | 1 |
| Borg (3 pages) | 57 (56.44%)[2] | 6 | 6 |
| Zodiac-408 | 71 (18.21%) | 4 | 4 |
| Borg (manual seg.) | 347 (28.61%) | 4 | 5 |

Table 5.4: Properties of the ciphers that we get from OCR, compared to the gold ciphers. Our integer program gives a quantitative measure if OCR has introduced some homophonicity to the cipher and/or did not represent some cipher characters.

[2]Transcription accuracy could only be computed for the first 5 lines (first 101 characters), which gave an edit distance of 57 (56.44%). The computation for the whole three pages exceeded 300 hours on a 2.2 GHz Intel Core i7 with 16 GB RAM.

# Chapter 6

# Conclusions and Future Directions

Decipering historical manuscripts is an intriguing challenge. Every cipher is usually a unique story, with a unique combination of language, system, and key. Building general-purpose automatic solvers remains a big goal we strive to achieve. It is our hope that this work has contributed towards this goal by addressing some decipherment problems and applying those methods on real historical ciphers.

## 6.1 Conclusions

In this thesis, we have presented decipherment methods and experiments on 1:1 substitution and homophonic ciphers. We have worked on a real historical cipher collection. Applying our decipherment methods resulted in automatic cracking of two historical ciphers; the "Borg" cipher and the Oak Island Cipher. Despite human transcription errors and missing characters from the original ciphers, our automatic decipherment was very robust and could yield almost perfect keys. We release data, cipher keys, cipher transcriptions, and answers on this website:

`http://bit.ly/2jdBdai`

We have also presented a machine learning technique for fast plaintext language ID for 1:1 substitution ciphers, with and without spaces. Our big motivation is to build a mobile phone app that automatically deciphers historical manuscripts. By

developing fast, lightweight methods for decipherment tasks, we hope that we are getting closer to that goal.

We have taken initial steps towards deciphering from images by implementing an unsupervised end-to-end decipherment system. We have experimented with deciphering printed text images and handwritten historical text images. We have also presented an integer linear programming method for evaluating transcription accuracy. While the end-to-end decipherment of "Borg" was not successful, our experiments have shown that decipherment might still be possible with better character segmentation. This opens up many directions for future work, which we discuss in the next section.

## 6.2 Future directions

As we have discussed previously, historical ciphers are a great source for cryptologic investigation. Our historical cipher collection is full of ciphers with different types. While this work focuses on substitution ciphers, future work might target codes or nomenclatures, among many other cipher types. A probably more challenging and intriguing path to take is trying to solve the famous unsolved ciphers, like the Voynich Manuscript or the Zodiac-340.

Deciphering from images is still an open problem. There are many ways to further improve our end-to-end system. One is building better models for character segmentation. Since our experiments have shown that character segmentation is a major bottleneck for the system, it should be a great target for improvements. Another possible direction is trying different algorithms for unsupervised character clustering. Of course, our goal is to minimize automatic transcription error, and ultimately, get more accurate decipherments.

It would also be interesting to try lattice decipherment instead of string deciphement. The idea here is that the OCR system gives the deciphement system a lattice of possible cluster IDs and lets decipherment chooses the best sequence, guided by a language model. This might help with edge cases where the OCR part does not have enough information to decide on cluster assignments.

People around the world will probably continue to discover many undeciphered manuscripts. Or we might get invaded by Aliens and need to decipher their language. Who knows? At the very least, we still have hundreds of undeciphered historical documents in many European libraries and archives. The contents of those documents will remain mysterious until someone unveils them. It is our dream that one day, we will be able to provide historians with a handy mobile phone app that allows them to read those documents and uncover the secrets of the past.

# Appendix A

# Challenge Cryptanalysis Problems

In this appendix, we present the full set of synthetic ciphers that we used for the decipherment experiments presented in chapter 3. Data files can be found here:

`http://bit.ly/2jdBdai`

### Cipher#1 (353 characters, from English)

kac butnqymkupqmr tckauv ql m tckauv ux rqyzjqlkqb mymrdlql kamk ql jlcv ku lkjvd kcfkl gaqba mpc gpqkkcy qy my jyeyugy rmyzjmzc myv ku lkjvd kac rmyzjmzc qklcrx gacpc kac jyeyugy rmyzjmzc aml yu unhqujl up ipuhcy gcrrjyvc-plkuuv brulc pcrmkqhcl myv gacpc kacpc mpc xcg nqrqyzjmr kcfkl gaqba tqzak ukacpgqlc amhc nccy jlcv ku acri jyvcplkmyv kac rmyzjmzc

### Cipher#2 (150 characters, from English)

ldsg obmy ybbujsy zyblu qj jds aysqf bw xqlg rbbf bmj obmy lcgxbl qgx crr as ebgs obmys jds ysqubg ct qjyqksrcge bg amj xbgj jdcgf jlczs cju qrr ycedj

### Cipher#3 (653 characters, from English, spaces removed)

egjtvsoztsaqhresozoeavigfsgfitaomtvozingxsfthqhrlttfngxstntavortziteiqhetvghzegj tquqohqhrrghzaftqlzggagghygszitvittkaazokkohafohqhrzitstahgztkkohuvigziqzoza hqjohuygszitkgatshgvvokkwtkqztszgvohygszitzojtazitnqstqeiqhuohuegjtathqzgsae

ghustaajthfktqatittrziteqkkrghzazqhrohzitrggsvqnrghzwkgelxfzitiqkkygsitziqzutza
ixszvokkwtitvigiqaazqkktrzitstaqwqzzktgxzaortqhrozoasquohuozkkagghaiqltngxs
vohrgvaqhrsqzzktngxsvqkkaygszitzojtazitnqstqeiqhuohuegjtjgzitsaqhryqzitsazisgx
uigxzzitkqhrqhrrghzesozoeomtviqzngxeqhzxhrtsazqhrngxsaghaqhrngxsrqxuiztsaqs
twtnghrngxsegjjqhrngxsgkrsgqroasqforknquohufktqatutzgxzgyzithtvghtoyngxeqh
zkthrngxsiqhrygszitzojtazitnqstqeiqhuohu

### Cipher#4 (128 characters, unspecified European language)

tjacxlsi tklsjp filip di sitrp ixfcjgr j yrzjy wki xrp eiygrxip erywki pjactrp gi zkiyyj
o oj di ixfcjtrp j gjy xkipdyr gipljyzr

### Cipher#5 (107 characters, unspecified European language)

xv krs xzc rpuxs deczb vxzc txsy krs vgtkxs vxzc trnmu krs bsrn hxs rslx rpux
deczb xs crtl xzcx ancbx isrn

### Cipher#6 (331 characters, no spaces, unspecified European language)

lrlpjnvoptucivnsxuhhpjnnvgsvronstcvuxgsnxegstxgstcgirvinhhnvgsoucngjsugourm
ftroknxtphnvbrcgsfnvgskvpseuoucngjsugoutgkgspllxndhulnanjnusrvvbrhhgspsegv
hnjghrmfbrvansnxhnjghogsnvgstcvrsntsnvnvgnsgstanjxburkvniulrlpjnvoptucivns
xuhhpjnxnvsugoutirvthnxpdgsvronsrmffnstthrvnkgsrobvrhhbrcgstnjegtuthrvnpllj
nkrvrmfbgjrsnegtogenpkpthlvutghnv

### Cipher#7 (168 characters, no spaces, unspecified European language)

epdrigxhdpxlxrkdrgxrhxtrdwvvxkdrgxhrcxhpiuxpfivvltrdkxvcrpvtxercgvgxvcprw
vphxrbxcxdlkvthvpxlmvgxedhxhvgxhvmrpwxvcmrpwxxlvhvhvxuixhrevfivaxgriex
vcgxwxlhvjrgnxhvjgxthpvl

**Cipher#8 (2376 characters, homophonic, no spaces, unspecified European language)**

47 21 11 24 19 35 27 02 11 30 51 38 22 37 02 41 40 35 39 50 01 41 34 18 14 20 44
28 40 14 31 10 06 45 34 49 47 04 44 19 13 43 09 43 52 20 13 45 23 14 27 39 29 08
14 15 02 41 34 47 44 34 42 54 03 48 09 47 07 49 34 16 04 37 27 12 29 45 47 34 29
06 42 23 46 30 38 45 40 14 01 24 22 45 19 15 25 40 31 19 47 05 22 23 44 26 52 08
39 47 38 51 02 43 19 45 11 30 44 19 25 10 44 52 13 15 02 41 25 30 20 48 09 37 48
20 42 47 07 24 32 30 51 05 19 41 21 43 44 32 31 21 25 12 44 08 47 10 49 28 20 35
48 23 42 12 27 17 23 43 31 21 42 08 36 24 21 22 18 14 07 48 32 12 14 32 16 43 01
45 03 08 36 48 19 15 02 35 51 34 47 07 17 02 36 45 21 28 40 08 44 03 04 18 13 24
48 02 38 12 44 08 17 01 49 33 16 45 29 19 11 45 22 30 19 24 07 39 37 08 42 10 55
45 47 08 50 04 18 14 13 21 48 03 44 39 37 51 31 10 15 16 34 41 09 34 44 52 23 30
06 16 25 02 44 32 42 37 23 24 02 38 36 23 47 12 42 45 09 30 54 06 44 01 09 37 48
33 42 53 33 02 22 50 13 30 39 37 38 49 04 47 06 18 46 39 46 32 47 34 15 01 17 24
04 34 16 36 23 26 03 40 35 06 14 15 33 45 11 28 06 38 31 02 49 33 16 22 36 37 02
16 22 41 46 47 01 39 31 08 47 06 49 12 29 26 21 24 11 51 20 32 48 10 27 20 29 49
02 24 29 21 42 32 29 52 32 36 44 08 50 10 35 26 34 41 54 07 48 12 42 08 05 34 14
20 34 51 09 30 11 12 25 30 39 49 38 30 23 28 09 45 38 18 14 08 53 34 23 48 13 28
09 52 38 43 16 22 47 34 25 21 31 08 05 37 44 12 46 26 33 41 35 39 14 47 21 34 48
19 29 08 44 34 46 05 42 32 18 10 27 43 40 41 46 28 38 49 23 28 09 53 32 39 19 32
27 10 42 43 20 24 52 40 47 07 44 22 52 19 11 43 44 38 42 41 32 36 06 26 20 16 32
36 05 50 09 30 18 19 47 05 44 34 42 12 44 43 47 10 15 01 22 08 27 49 11 22 27 02
46 10 27 28 07 15 33 43 44 12 15 16 39 30 11 33 37 06 27 19 25 03 17 50 06 37 44
33 47 32 42 41 22 51 02 15 46 10 31 14 13 35 27 04 19 26 37 38 18 26 31 21 42 53
33 10 11 55 46 08 29 01 22 24 34 30 16 40 47 47 03 43 05 26 32 48 02 35 09 41 11
26 44 23 29 04 27 22 25 08 42 47 04 49 13 35 25 10 31 45 13 26 07 24 02 39 37 05

44 40 35 43 13 48 05 41 32 17 53 32 20 41 35 39 49 02 16 22 25 26 34 47 09 30 32

36 25 50 09 37 11 30 22 49 07 47 02 17 54 30 20 24 11 30 37 38 16 26 11 28 48 38

42 05 35 07 44 32 18 14 31 12 44 23 31 48 04 36 23 47 01 16 48 33 56 20 39 45 23

30 44 52 06 10 46 02 16 43 39 45 32 16 42 08 25 30 38 45 40 47 03 24 22 45 31 20

51 10 35 14 06 16 09 24 46 30 21 08 18 10 15 43 40 35 54 10 29 12 37 24 05 21 11

12 46 03 16 43 38 45 40 28 33 41 35 12 44 35 39 37 50 12 37 08 29 20 49 04 47 06

15 54 29 23 27 08 29 07 38 19 37 35 09 18 50 39 56 22 47 10 46 07 17 43 39 45 32

17 31 08 51 03 35 23 13 51 31 22 48 12 44 08 36 18 02 15 04 26 09 41 53 33 13 48

09 47 11 51 23 26 33 16 46 30 40 43 39 37 51 28 13 16 17 03 49 33 16 08 33 43 52

21 29 06 14 24 02 43 04 14 36 10 42 45 13 32 27 45 23 26 31 08 42 45 32 28 46 32

31 19 25 33 42 01 47 47 32 43 42 07 27 24 22 41 34 15 53 34 02 36 40 16 27 28 19

41 51 08 29 19 27 45 13 30 14 40 35 22 42 19 48 26 11 23 34 26 09 41 42 12 37 24

09 41 35 39 16 26 31 19 42 50 13 28 25 07 42 33 18 50 09 37 11 30 19 49 04 47 06

14 54 30 22 24 11 30 37 38 18 26 11 28 53 32 01 40 47 04 42 45 39 37 27 03 54 05

44 07 38 43 40 35 17 06 47 01 40 43 39 28 32 36 34 41 46 30 21 12 54 32 02 16 25

05 53 32 01 53 32 12 05 21 42 41 32 47 27 38 18 45 11 28 22 35 36 40 16 48 02 51

36 34 42 12 25 54 03 48 11 47 13 41 25 10 46 28 21 08 49 10 15 21 46 38 14 26 38

47 04 43 20 10 17 12 48 04 16 41 07 36 02 16 27 28 32 41 01 35 50 19 31 06 07 18

42 20 51 36 06 45 39 37 48 12 29 06 16 40 54 54 04 43 06 07 14 53 34 22 42 40 49

30 13 41 54 05 48 12 37 26 09 28 25 12 37 38 16 26 31 33 42 41 01 18 14 33 42 08

55 26 31 21 40 30 48 07 35 22 31 04 33 41 44 39 37 15 27 05 25 33 01 41 33 18 19

48 46 30 02 41 32 41 44 38 36 51 31 12 18 14 33 41 11 33 44 52 21 30 01 14 25 04

44 34 42 36 21 24 04 40 35 21 47 08 42 28 11 06 46 32 49 47 04 43 19 08 44 10 44

52 20 08 48 01 11 49 34 17 55 50 55 50 03 04 46 30 38 55 02 41 04 41 11 36 14 02

16 40 44 24 40 49 31 03 14 05 37 32 29 49 11 49 29 32 36 10 35 17 04 43 09 47 11

49 29 20 36 48 32 41 52 21 44 48 19 24 21 38 44 43 20 16 07 38 36 10 32 36 23 43

33 41 17 23 44 28 07 17 46 19 15 26 39 28 01 49 34 17 39 41 35 01 49 32 14 53 32

12 44 52 29 04 17 24 07 20 36 03 18 54 13 28 50 02 48 20 16 48 09 40 29 12 48 48

09 15 51 28 23 26 05 21 17 53 34 38 48 50 38 43 13 08 33 43 52 20 31 01 18 26 04

21 17 07 51 37 03 54 05 43 20 26 32 28 46 30 39 17 32 49 47 04 42 06 48 39 37 38

44 40 29 45 40 28 03 15 11 24 18 19 36 51 32 05 37 07 16 43 52 28 05 16 27 01 11

39 16 48 09 41 53 33 13 04 36 50 07 26 23 49 04 16 22 48 48 02 50 04 35 19 42 49

09 37 11 48 06 43 27 02 38 36 13 41 07 41 46 47 40 30 20 37 48 23 42 45 09 30 46

28 40 43 09 18 14 07 40 35 09 41 11 33 44 52 19 28 04 17 27 07 44 21 42 41 13 35

16 37 38 14 27 28 22 10 14 24 34 26 39 41 35 12 17 52 33 05 32 17 37 19 27 04 38

36 07 15 46 22 18 24 40 30 09 18 44 52 31 06 18 26 02 54 04 48 09 47 11 17 44 38

52 40 31 24 13 35 27 34 29 33 27 08 33 43 52 20 29 01 18 24 04 43 34 41 14 20 44

28 05 54 07 43 02 33 42 54 29 10 53 33 13 35 27 10 29 20 44 48 09 50 38 24 13 45

19 28 25 06 44 01 46 10 35 24 29 08 44 40 35 43 05 47 07 23 27 03 40 35 06 14 15

21 44 28 19 41 11 36 25 38 20 49 47 33 19 35 25 34 31 14 20 43 31 40 04 49 06 48

09 41 31 12 54 06 44 01 21 36 27 32 29 43 38 36 50 04 50 06 38 53 34 38 17 50 11

28 34 30 36 52 38 30 27 23 42 34 31 33 25 14 20 43 11 30 48 38 25 10 18 14 34 38

17 45 47 33 31 04 42 06 11 55 02 18 27 05 42 12 37 24 53 34 05 23 48 38 42 03 35

39 13 47 09 43 27 04 16 32 36 24 11 26 44 38 36 15 27 02 25 32 27 06 11 33 44 52

21 30 06 14 25 02 22 10 42 06 37 01 18 26 31 05 22 26 53 34 11 48 13 06 54 01 47

03 34 15 53 34 38 26 03 48 05 10 48 12 16 43 09 37 48 07 27 48 09 15 05 36 34 45

20 24 29 05 14 15 33 45 11 28 20 47 27 23 29 12 04 35 42 19 35 02 49 32 14 15 21

43 13 28 48 40 27 07 14 15 54 28 23 36 44 01 17 43 32 15 21 17 18 02 15 04 13 35

14 01 16 22 48 41 39 37 01 26 03 38 36 10 18 47 13 51 22 24 06 39 37 08 42 26 32

21 42 26 39 24 32 41 53 32 09 43 38 36 51 30 09 17 18 33 42 08 36 07 55 12 44 38

42 41 07 26 27 03 42 34 16 50 23 47 03 48 22 12 07 35 25 10 31 43 13 16 17 02 38

36 05 49 09 19 27 03 17 18 06 41 21 12 50 04 29 51 07 36 02 15 42 22 29 07 20 08

39 41 35 06 34 41 53 34 11 16 22 35 44 26 38 28 34 41 52 34 02 33 14 35 21 26 01
39 37 04 17 49 11 36 13 48 04 43 27 07 40 35 09 41 48 12 42 32 42 21 46 39 14 26
39 47 03 43 20 41 44 23 11 47 13 17 27 07 15 51 31 23 25 06 21 12 37 32 36 24 06
23 42 08 25 45 31 39 46 08 36 18 19 12 37 38 18 26 31 21 13 50 39 47 32 36 27 19
25 07 17 24 13 15 27 10 42 53 34 21 42 26 06 49 01 02 42 45 29 06 42 01 14 47 04
49 13 35 27 10 29 04 42 45 09 30 27 03 42 32 17 50 38 47 34 41 33 14 35 39 42 03
36 10 35 39 18 24 29 40 44 33 37 43 27 03 18 45 20 16 25 40 31 04 49 32 18 28 11
47 04 51 06 40 16 02 18 50 05 28 07 18 13 24 42 34 47 07 13 30 03 49 32 15 44 52
29 03 15 24 06 54 01 48 11 47 05 49 33 16 47 23 04 43 06 17 44 02 50 04 47 06 49
33 16 22 33 43 25 39 31 02 25 20 27 06 49 32 15 10 25 39 42 35 01 49 32 14 44 38
36 51 29 12 16 17 33 14 11 33 44 52 19 28 07 17 27 03 44 05 46 23 29 26 03 43 05
45 20 35 25 03 49 33 16 45 09 28 20 41 22 35 25 10 31 47 23 30 51 07 22 30 03 15

**Cipher#9 (436 characters, homophonic, no spaces, unspecified European language)**

47 21 11 24 19 35 27 02 11 30 51 38 22 37 02 41 40 35 39 50 01 41 34 18 14 20 44
28 40 14 31 10 06 45 34 49 47 04 44 19 13 43 09 43 52 20 13 45 23 14 27 39 29 08
14 15 02 41 34 47 44 34 42 54 03 48 09 47 07 49 34 16 04 37 27 12 29 45 47 34 29
06 42 23 46 30 38 45 40 14 01 24 22 45 19 15 25 40 31 19 47 05 22 23 44 26 52 08
39 47 38 51 02 43 19 45 11 30 44 19 25 10 44 52 13 15 02 41 25 30 20 48 09 37 48
20 42 47 07 24 32 30 51 05 19 41 21 43 44 32 31 21 25 12 44 08 47 10 49 28 20 35
48 23 42 12 27 17 23 43 31 21 42 08 36 24 21 22 18 14 07 48 32 12 14 32 16 43 01
45 03 08 36 48 19 15 02 35 51 34 47 07 17 02 36 45 21 28 40 08 44 03 04 18 13 24
48 02 38 12 44 08 17 01 49 33 16 45 29 19 11 45 22 30 19 24 07 39 37 08 42 10 55
45 47 08 50 04 18 14 13 21 48 03 44 39 37 51 31 10 15 16 34 41 09 34 44 52 23 30
06 16 25 02 44 32 42 37 23 24 02 38 36 23 47 12 42 45 09 30 54 06 44 01 09 37 48

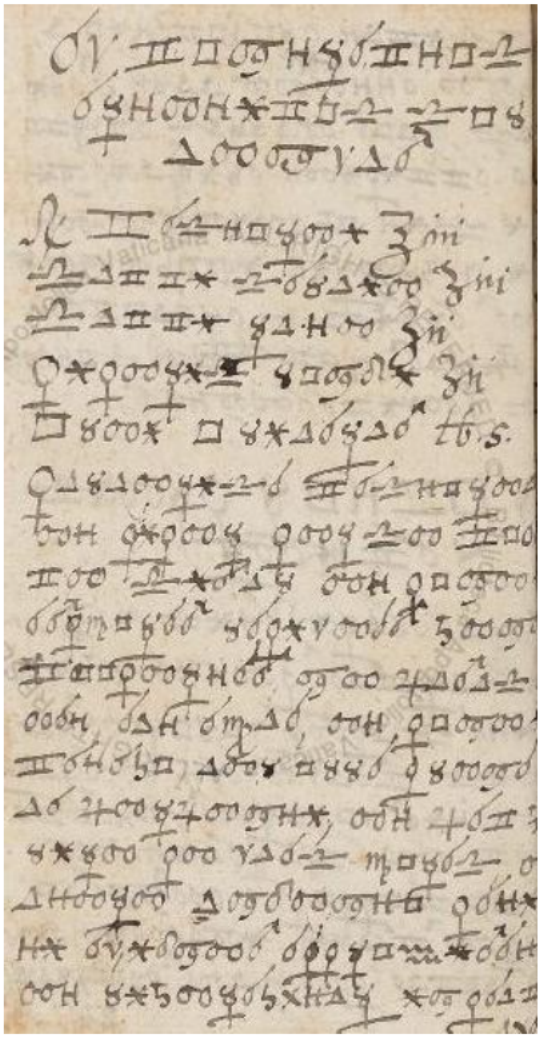
33 42 53 33 02 22 50 13 30 39 37 38 49 04 47 06 18 46 39 46 32 47 34 15 01 17 24
04 34 16 36 23 26 03 40 35 06 14 15 33 45 11 28 06 38 31 02 49 33 16 22 36 37 02
16 22 41 46 47 01 39 31 08 47 06 49 12 29 26 21 24 11 51 20 32 48 10 27 20 29 49
02 24 29 21 42 32 29 52 32 36 44 08 50 10 35 26 34 41 54 07 48 12 42 08 05 34 14
20 34 51 09 30 11 12 25 30 39 49 38 30 23 28 09 45 38 18 14 08 53 34 23 48 13 28
09 52 38 43

# Appendix B

# The "Borg" Cipher

In this appendix, we present more pages of the "Borg" cipher, with decoded Latin and English translation (translation by Urban Örneholm). We release the full transcription of the book, deciphered Latin text, and English translation on this website: `http://stp.lingfil.uu.se/~bea/borg/`

**Page 0024v**

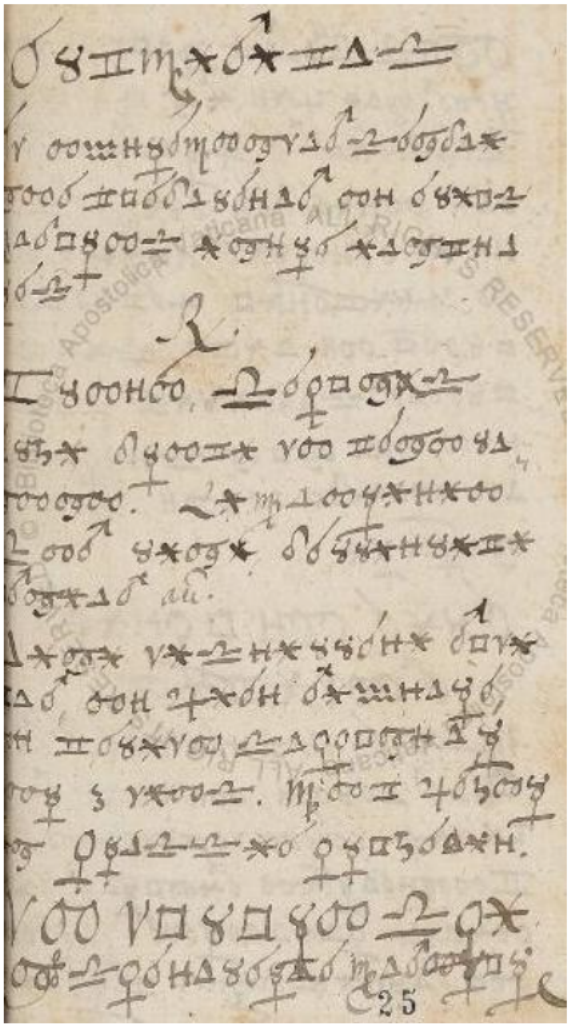


ad contractos
arteticos sol-
uendum
R. castorei (uncia-drachmam) [iii]
succi saluie (uncia-drachmam) [iii]
succi rute (uncia-drachmam) [ii]
piperis longi (uncia-drachmam) [ii]
olei oliuarum [libram semis]
puluerisa castoreum
et piper per se coer-
ce simul et pone in
amp(q-h)oram lapideam bene
coopertam, ne fumus ab-
eat aut a(q-h)ua, et pone in
catcabo, uel olla plena a(q-h)-
ua ferfenti, et fac bul-
lire per duas (q-h)oras e
utere ungento patien-
ti ad, ignem approximato
et liberabitur in pauc

For loosening contracted joints

Take three ounces of castoreum, three drachms of juice of sage, two ounces of juice of rue, two drachms of long pepper, and half a pound of olive oil. Pulverize the castoreum and the pepper separately, mix them together and put them in a carefully closed stone jar, so neither fumes might escape nor water, and place it in a clay pot, or jar, full of simmering water, and make it boil for two hours, and use as an ointment with the patient close to the fire, and he will be liberated in a few [..]

**Page 0025r**

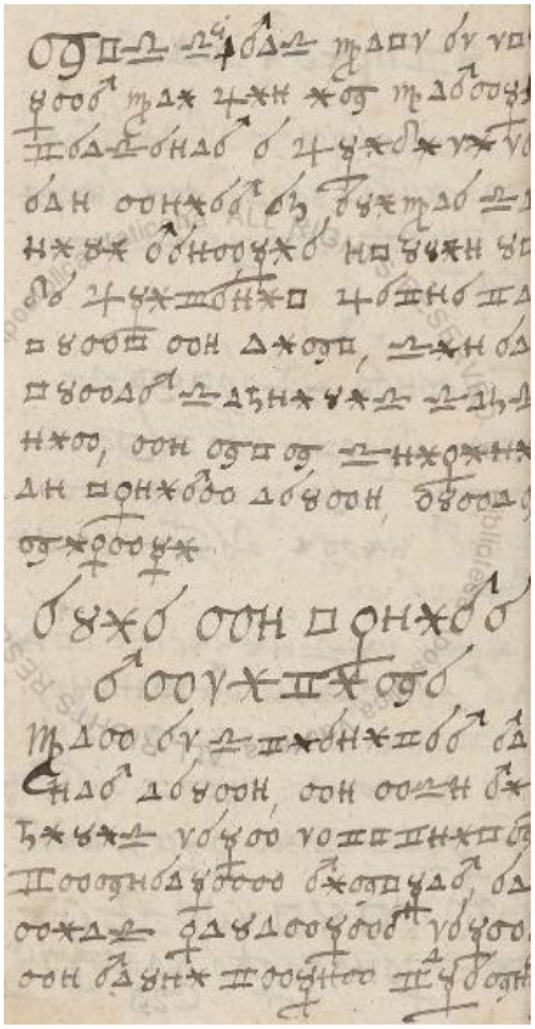


alc(q-h)imicus
ad extra(q-h)endum sangui-
neam coagulatum, et alios
(q-h)umores intra iunctu-
ras
[R.]
clr?ete, saponis
[A]lbi greci de cane lu-
?nene. [L]i(q-h)ueritie
sem lini, gallitlici
omnium [aú].
uini distillati modi-
cum, et fiat mixtura-
et calide suppontur
per [3] dies. (q-h)ec faber
in prussia probauit.
de dolore spi-
ne spatularuam (q-h)umerorum

an alchemist on extracting coagulated blood and other fluids inside the joints Take chalk(?), soap, album graecum from a [..] dog, licorice, linseed, and seed of vervain, equal parts of all; a modest amount of brandy, and it should be a mixture, and it is applied hot for three days. This, a craftsman in Prussia has proved.

On back pain of the shoulder blades

**Page 0025v**

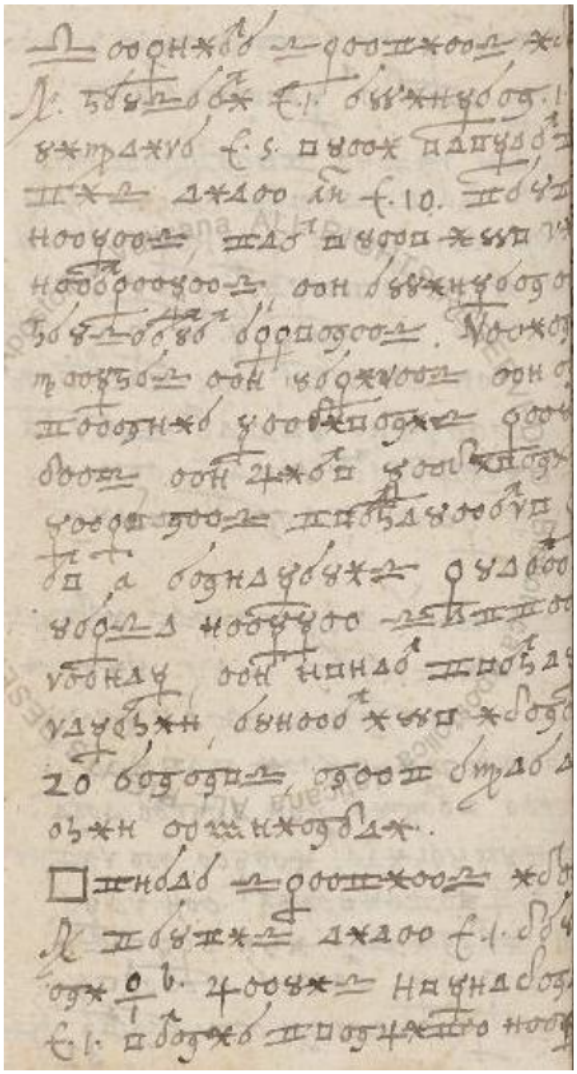


nos sumus (q-h)uod ad dolo-
rem (q-h)ui fit in (q-h)umeri
causatum a frigidida
aut etiam ab ali(q-h)ua sub-
tili materia tollit lon-
ga fricatio facta cum
oleo et uino, sit autem
oleum subtilis substan-
tie, et non stipitis
ut optime ualet, oleum
niperi
alia et optima
medicina
(q-h)ue ad sciaticam mul-
tum ualet, et est mira-
bilis dare decoctionem
centauree minorum, aut
eius puluerem dare (uncia-
drachmam)
et multi certe curantur

We know that, for pain in the shoulders from cold reasons, or even from some subtle matter, a long rub with oil or wine cures this; the oil should, however, be of fine quality, and not made from twigs to be most efficient, juniper oil.

Another, and very good, remedy which is very efficient and wonderful in sciatica, is to give a decoction of common centaury, or [..] ounces of a powder thereof, and many are certainly cured.

**Page 0031v**



septima species ig
R.? balsami [E.I.] allitran 1.
li(q-h)uida [E.S.] olei ouorumc
cis uiue [an] [f. 10.] calc
teres, cum oleoillo di
temperes et allitrane
balsamlm appones. N?ein
(q-h)erbas et lapides et n
centia regionis pel
ges et fimo legionX
repones combulemdo
mo A anturalis plume
lapsu terre succe
detur, et totum combur
durabit, altem illo igne?
[20] annos, nec a(q-h)uau
pbit extingui..
octaua species ige?
R. calcis uiue [E.I.] gal
ni p? ?. felis tortugn
[E.I.] omnia confice tep?

The seventh kind of fire

Take one pound of balm, half a pound of alkitran or liquid pitch, ten pounds each of oil of eggs, and quicklime. Grind the lime, then you dilute it with the oil, and add alkitran and balm. Then you pour it on herbs and stones, and anything that grows in the region, to burn it. And from the first natural rain the earth will be set on fire, and it will all burn. This fire will however last for twenty years, and will not be extinguished by water.

The eighth kind of fire

Take one pound of quicklime, six ounces of galbanum, one pound of tortoise bile. Mix everything, grinding it

# Bibliography


Taylor Berg-Kirkpatrick and Dan Klein. Decipherment with a million random restarts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 874–878, 2013.

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. Unsupervised transcription of historical documents. In *ACL (1)*, pages 207–217. The Association for Computer Linguistics, 2013. ISBN 978-1-937284-50-3.

Alan Clements. *Computer Organization and Architecture: Themes and Variations*. Cengage Learning, 2013. ISBN 978-1111987046.

Eric Corlett and Gerald Penn. An exact A* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1040–1047, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

John F. Dooley. *A Brief History of Cryptology and Cryptographic Algorithms*. Springer International Publishing, 2013. ISBN 978-3-319-01627-6.

Qing Dou and Kevin Knight. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 266–275, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Qing Dou and Kevin Knight. Dependency-based decipherment for resource-limited machine translation. In *EMNLP*, pages 1668–1676. ACL, 2013. ISBN 978-1-937284-97-8.

Qing Dou, Ashish Vaswani, and Kevin Knight. Beyond parallel data: Joint word alignment and decipherment improves machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 557–565, Doha, Qatar, October 2014. Association for Computational Linguistics.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. Unifying Bayesian inference and vector space models for improved decipherment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 836–845, Beijing, China, July 2015. Association for Computational Linguistics.

Chi Fang and Jonathan J. Hull. Modified character-level deciphering algorithm for OCR in degraded documents. volume 2422, pages 76–83, 1995. doi: 10.1117/12.205843.

Jonathan Graehl. Carmel finite-state toolkit, 2010. URL `http://www.isi.edu/licensed-sw/carmel`.

Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016. URL `http://www.gurobi.com`.

Tin Kam Ho and G. Nagy. OCR with no shape training. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, pages 27–30 vol.4, 2000. doi: 10.1109/ICPR.2000.902858.

Gary B. Huang, Erik G. Learned-Miller, and Andrew McCallum. Cryptogram decoding for OCR using numerization strings. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil*, pages 208–212, 2007. doi: 10.1109/ICDAR.2007.93.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`.

Kevin Knight and Kenji Yamada. A computational approach to deciphering unknown scripts. In *in: Proceedings of the ACL Workshop on Unsupervised Learning in Natural Language Processing*, 1999.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 499–506, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Kevin Knight, Beáta Megyesi, and Christiane Schaefer. The Copiale cipher. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, BUCC '11, pages 2–9, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-015.

John Langford, Lihong Li, and Alex Strehl. Vowpal Wabbit, 2007. URL `https://github.com/JohnLangford/vowpal_wabbit/wiki`.

Marcus Liwicki, Alex Graves, and Horst Bunke. *Neural Networks for Handwriting Recognition*, pages 5–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-24049-2. doi: 10.1007/978-3-642-24049-2_2.

Shervin Malmasi and Mark Dras. Language identification using classifier ensembles. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 35–43, Hissar, Bulgaria, September 2015. Association for Computational Linguistics.

G. Nagy, S. Seth, and K. Einspahr. Decoding substitution ciphers by means of word matching with application to OCR. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):710–715, May 1987. ISSN 0162-8828. doi: 10.1109/TPAMI.1987.4767969.

Malte Nuhn and Kevin Knight. Cipher type detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1769–1773, Doha, Qatar, October 2014. Association for Computational Linguistics.

Malte Nuhn, Julian Schamper, and Hermann Ney. Beam search for solving substitution ciphers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1568–1576, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Malte Nuhn, Julian Schamper, and Hermann Ney. Improved decipherment of homophonic ciphers. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1764–1768, Doha, Qatar, October 2014. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sujith Ravi and Kevin Knight. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the Conference on Empirical Methods*

*in Natural Language Processing*, EMNLP '08, pages 812–819, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. Probabilistic methods for a Japanese syllable cipher. In *Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy, 22nd International Conference, ICCPOL 2009, Hong Kong, March 26-27, 2009. Proceedings*, pages 270–281, 2009. doi: 10.1007/978-3-642-00831-3_25.

Sujith Ravi and Kevin Knight. Bayesian inference for Zodiac and other homophonic ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 239–247, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9.

Ray Smith. An overview of the Tesseract OCR engine. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 629–633, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2822-8.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1048–1057, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Sebastian Spiegler and Christian Monson. EMMA: a novel evaluation metric for morphological analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1029–1037. Association for Computational Linguistics, 2010.

Tongtao Zhang, Aritra Chowdhury, Nimit Dhulekar, Jinjing Xia, Kevin Knight, Heng Ji, Bülent Yener, and Liming Zhao. From image to translation: Processing the endangered Nyushu script. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 15(4):23:1–23:16, May 2016. ISSN 2375-4699. doi: 10.1145/2857052.