CMPUT 499 Fall 2020 Course Project Report

Choi, Colin

January 25, 2021

Abstract

The Dorabella cipher is derived from a message that Edward Elgar sent to Dora Penny. It consists of a set of 24 symbols and it is 87 characters long. Edward Elgar is also the 19th century composer who composed Pomp and Circumstance. Thus what are these symbols? Are they an encrypted message? In this paper, we first run some experiments that look into a couple of interesting claims about the Dorabella cipher from a YouTube video by Keith Massey [4]. We then compared the accuracy of some different state of the art decipherment programs and retrieve our best decipherment of the Dorabella cipher. Finally, we experiment with a language identification program [2] in an attempt to identify the language of the Dorabella cipher.

1 Introduction

Edward Elgar was a popular 19th-century composer best known for works such as Pomp and Circumstance, and the Enigma variations. He was a known cryptography enthusiast and supposedly used cryptography as an inspiration for the Enigma variations.

The Dorabella cipher is a ciphertext that was sent from Edward Elgar to Dora Penny. The cipher contains 24 unique characters forming a text of 87 characters total. The characters are a series of 1 to 3 bumps rotated in angles of 0, 45, 90, 135, 180, 225, and 270 degrees. Is this message text or could it be a musical piece? Given that the author is dead it is unlikely we'll ever have definitive evidence of what the final decipherment actually is, but we can try to come close.

In this paper, we give our best attempt at obtaining a decipherment of the Dorabella cipher. We also explore the idea of whether the Dorabella cipher is music or text. This cipher has been around for hundreds of years and no one has yet to come up with a solution to solve this mysterious cipher. Our approach to deciphering and identifying the language of the Dorabella cipher is to use language models. Our experiments in this paper show that the Dorabella cipher is not a straightforward substitution cipher written in English.

In the following sections of this paper, we will first look at some previous contributions to the decipherment and language identification of substitution ci-

phers. Then after reviewing this previous work, we will discuss the experiments we performed where we use these decipherment and language identification techniques. Our first experiment explores a couple of interesting claims made about the validity of the Dorabella cipher. We then run tests comparing multiple state-of-the-art decipherment programs in an attempt to obtain a decipherment of the Dorabella cipher. Finally, we attempt a language identification experiment in order to to find the language of the cipher.

2 Literature Review

Although the Dorabella cipher remains unsolved, there is a lot of related research that we can draw upon to assist us in solving this elusive cipher. We want to give our best attempt at obtaining a decipherment of the Dorabella cipher and we also want to determine if the Dorabella cipher is more likely to be text or music. The following papers in this literature review revolve around research on decipherment programs and techniques, language identification, and music. We will be running our experiments using substitution ciphers solvers and a language identification program from the papers reviewed. We also ran a cipher type identification program that is explored in one of the following papers.

2.1 Natural Language Corpus Data

In chapter 14 of Beautiful Data: The Stories Behind Elegant Data Solution, Peter Norvig [5] gives a great introduction to language models and the techniques used to create a decipherment program. It first describes a trillion-word corpus taken from publicly available web pages on the internet. A corpus is a collection of text treated as a sequence of tokens. Each distinct token is called a type and the 10 most common types account for almost a third of the tokens in the trillion word corpus. Sequences of tokens can be split up into n-grams and the conditional probability of a series of n-grams can be calculated using a language model derived from a corpus. This ability to calculate probabilities using a language model is very powerful and is a key part of most decipherment programs in this paper. Language models can be used to solve problems such as word segmentation or determining which decipherment of a text is more likely to be correct. For a problem like word segmentation, if we are given an English text without spaces, we can somewhat easily identify the word boundaries and extract the meaning of the text. For certain texts, this task can be very ambiguous, but we humans are able to draw upon years of experience to figure out what makes more sense. When doing this on a computer, the language models we create can be used in place of human experience. Norvig describes an established methodology for solving problems like word segmentation and decipherment. First, we define a probabilistic model, then we enumerate through the candidates, and finally we choose the most probable candidate by applying a language model to each. If we applied this methodology to deciphering substitution ciphers, we see we would run into some difficultly because we would

have to enumerate through all 26! possible keys. Instead of trying to enumerate through all possible keys, we can strategically discover potential keys by enumerating our candidate using a hill-climbing technique with random restarts after a certain number of steps have been taken. This chapter goes step by step on how the Norvig solver was built and gives an in-depth description on how the multiple modules in the code work together in order to crack a substitution cipher.

2.2 Solving Substitution Ciphers with Combined Language Models

Continuing on the topic of decipherment programs, Hauer [1] talks about a novel method of solving monoalphabetic ciphers. The HHK solver explores a method of scoring candidate keys with a combined language model that incorporated both character-level and word-level information. One drawback to using a word language model is that it would be very difficult to train the language model with enough data to get probabilities for all possible words. To work around this issue, a special unk character is appended at the very end of the training corpus. Since this special unk character only appears once in the text, it would be given a very low probability and any out-of-vocabulary words would be assigned this probability. Another technique the solver used was incorporating a pattern list for the key mutation process. Pattern equivalent n-grams were used to generate new keys that are likely to achieve a higher score. Combining these techniques together, the experiments in the paper showed that the HHK solver was able to outperform all the other solvers it was tested against.

2.3 Bayesian Inference for Zodiac and Other Homophonic Ciphers

Our last paper on decipherment discusses a novel probabilistic decipherment approach using Bayesian inference for deciphering complex substitution ciphers such as the Zodiac-408 cipher [8]. Most of the work on the area of solving substitution ciphers is focused on solving simple substitution ciphers where the substitution key is deterministic. This means that there is a 1 to 1 mapping between the plaintext letters and ciphertext symbols. What makes solving the Zodiac-408 cipher so special is that it was discovered to be a homophonic cipher where the key can map a plaintext letter to multiple symbols. Having plaintext letters map to more than one symbol flattens out the frequency distribution of symbols, making a frequency-based cryptanalysis attack difficult. This combined with the fact that the Zodiac-408 cipher contains no spaces, has intentional misspellings of words, and has a section of gibberish appended at the end made it a very hard cipher to crack. The techniques discussed in this paper involves combining a word dictionary and an n-gram language model to crack the Zodiac-408 cipher. The word dictionary part of the model helps guide the solver to favor high probability letter sequences that form valid words. The n-gram part helps the solver deal with unseen words or the misspelling in the

Zodiac-408 cipher. Since the Dorabella cipher only uses 24 symbols and there are 26 letters in the English alphabet, there is a chance that it could be using a homophonic cipher where the symbols could map to more than one letter. This paper gives us a method to decipher the Dorabella cipher if it is in fact a homophonic cipher.

2.4 Cipher Type Detection

There have been various encryption methods used throughout the centuries. In this paper, Nuhn and Knight [6] talk about their implementation of a state-of-the-art classifier for detecting which type of encryption is used for a cipher. For certain ciphers, there exist automated tools for deciphering them, however, these tools first require the user to know what type of encipherment method is used. Traditionally, this often required focusing human resources on analyzing and classifying cipher types which can be very time-consuming. Nuhn and Knight's goal is to automate the task of finding the encryption type. They take a machine learning approach that builds on top of the existing BION classifier program [1]. There is a list of 56 encryption methods provided by the American Cipher Association, each with detailed descriptions and examples of the different cipher types. Using this, they were able to create an infinite amount of labeled training data for their machine learning approach. They started off by reimplementing all of the features used in the BION classifier and then adding three newly developed sets of features, Repetition Feature (REP), Amsco Feature (AMSC), and Variant Feature (VAR). Adding these 3 features, they were able to achieve a cipher type detection accuracy of 58.49% which is 11% higher than the BION classifier. Identifying the type of encryption used is an important part of determining what techniques should be used to solve a cipher. In our case, we want to know what type of encryption the Dorabella cipher uses because most of the decipherment tools we explored are meant for substitution ciphers. Since the BION classifier is available on the web, we ran the Dorabella cipher through this classifier. Patristocrat, the ACA terms for a simple substitution cipher, was the detected cipher type. Although the BION classifier only has a 47% accuracy, this is still a promising sign to explore our substitution cipher decipherment programs.

2.5 Decoding Anagrammed Texts Written in an Unknown Language and Script

The Voynich manuscript is a medieval codex written in a unique script in an unknown language [2]. We learned from the cipher type identification paper that it is hopeless to start any decipherment attempts without first knowing what encryption scheme was used, the same can be said about first figuring out what language the original text was written in. The process for deciphering a

[1] http://bionsgadgets.appspot.com/gadget_forms/nnet_id_test_collection.html

text involves first identifying the language of a ciphertext. This is done by comparing it to samples of known languages. Next, each symbol of the ciphertext is mapped to a letter in the identified language. Finally, any resulting anagrams are decode into readable text. Language identification has been a crucial step in the decipherment of ancient scripts and this paper proposed 3 methods for determining the source language of a cipher, relative character frequencies, patterns of repeated symbols, and trial decipherment. Out of the three proposed language identification techniques, the first two methods are unaffected by characters reordering. Character frequency analysis involves performing frequency analysis on the ciphertext and on multiple sample languages. The idea here is that each language would have its own uniquely identifiable frequency signature and the distance between the frequency of the text and the frequency of the sample language can be calculated. The next method, Decomposition Pattern Frequency, expands on the first method by incorporating the decomposition patterns instead of just looking at character frequencies. The last method, Trial Decipherment, involves deciphering the cipher into each of the candidate languages. Bigram language models are generated using samples from the candidate languages, then the decipherment with the highest probability indicates the most likely language of that text. The trial decipherment method of language identification was able to achieve a 97% accuracy on training and test samples taken from a UDHR corpus.

2.6 Developing Fitness Functions for Pleasant Music: Zipfs Law and Interactive Evolution Systems

Zipfs law is a commonly occurring property of many areas in our everyday lives such as natural language and music. It describes the phenomena where certain types of events follow a power law, meaning some events are quite frequent while others are very rare. In English for example, the second most frequent word is used about half as often as the most frequent word, the third most frequent word is used about a third as often as the most frequent, and so on. Socially-sanctioned music displays a large number of metrics, such as melodic intervals and melodic bigrams, that follow a Zipfian distribution [3]. With this discovery, a set of 40 metrics based on Zipfs law was identified and used to train ANNs to classify pieces of music in terms of author, style, and pleasantness. Using these metrics, the author identification experiments were 95-100% accurate and the style identification experiments were 91-95% accurate. For the experiments classifying pleasantness, subjects were presented with 12 pieces of music and were asked to continuously reposition their mouse in a 2D selection space to indicate their pleasantness and activation reaction to the music. The ANN was trained on the data from 11 of the pieces of music and was able to predict with 98% accuracy their pleasantness response to the 12th piece. Its quite interesting how Zipfs law makes its way into music, and how useful these metrics turned out to be.

3 Related work

3.1 Prior work on decipherment

There have been multiple research papers on the automatic decipherment of substitution ciphers. Typically, decipherment involves enumerating through multiple keys and scoring each decipherment obtained to find the best choice. The ones we will be exploring in this paper are the HHK solver [1], the Norvig solver [5], Unravel [7] and Anirudh's solver.

Each of these solvers take a slightly different approach on decipherment. For example, the Norvig solver enumerates through potential keys using a hill-climbing technique with random restarts and scores the decipherments using a trigram language model. The HHK solver used a pattern list for the key mutation process and scores texts using both word level and character level trigram language models. We will be comparing these solvers in this paper to identify their strengths and weaknesses. We hope these tools can help assist us in deciphering the Dorabella cipher.

3.2 Prior work on language identification

The Voynich document is a enciphered document of relatively unknown origin that has plagued cryptographers for centuries. The document contains illuminated text that is not in any known language thus most assume it has been enciphered. Current analysis of Voynich document by Hauer and Kondrak [2] indicates that it is likely enciphered Hebrew. This was done by asking language models, such as n-gram models of different languages if different decipherments of texts of the Voynich manuscript were surprising to each language model. We will be using their work in our attempt to identify the Dorabella cipher's language.

4 Methods

4.1 Language models

n-gram models are models of sequences where one can calculate the probability of a sequence occurring based on past sequences. For a n-gram model, one can ask how common is p_4 given p_1, p_2 and p_3 before it. An n-gram language model will answer this question: $p(p_4|p_1p_2p_3)$. With a language model, one can determine that $p(teapot|I'malittle) > p(monolith|I'malittle)$. We can also determine the probability of a decipherment given a language model. These language models are a key part of all our decipherment and language identification programs.

The language identification and decipherment programs we used all have their own specialized programs for creating language models. Aside from how the language model is formatted, they all follow a similar process of counting the number of unigrams, bigrams, and trigrams from a text file. For any experiments

not involving the language identification or decipherment programs, we created language models using KenLM [2].

5 Experiments

5.1 Assumptions/encoding

For the following experiments, we will be working under the assumption that the Dorabella cipher is a monoalphabetic substitution cipher. Due to Elgar sending the cipher to Dora Penny without a key, our guess is that the encryption method used wasn't meant to be something complicated. We also have some support from the BION cipher type detection program [3] that this could be a substitution cipher.

5.2 Data/corpora

We used corpora from a variety of sources and languages in our experiments.

The Natural Language corpora consists of:

- The Adventures of Sherlock Holmes by Arthur Conan Doyle [4](Test Samples)
- The Letters of Jane Austen by Jane Austen [5](Test Samples)
- Dangerous Connections by Choderlos de Laclos [6](Training data for MASC solvers)
- New York Times Corpus [1](Training data for HHK Solver)
- Universal Declaration of Human Rights [2](Language Identification Training data)
- Corpra from Open Subtitles (Language Identification Training data)

The Music corpora consist of:

- Some of the monophonic tracts extracted from a collection of Edward Elgar MIDI files.
- Some of the monophonic tracts extracted from a collection of Bach MIDI files.

[2] https://kheafield.com/code/kenlm/
[3] http://bionsgadgets.appspot.com/gadget_forms/nnet_id_test_collection.html
[4] http://www.gutenberg.org/ebooks/1661
[5] http://www.gutenberg.org/ebooks/42078
[6] http://www.gutenberg.org/ebooks/45512

5.3 Normalization of Data

5.3.1 Text Data

To reduce the complexity of the text used in our experiments, we remove special characters, numbers, and lowercase all the natural language corpora used. Test samples used in the following experiments are all 87 length long, the same length as the Dorabella cipher. These 87 length test samples are created by first randomly selecting a number between 0 and the number of words in the corpus. Starting at the word at that index, a sample is created by appending each subsequent word one by one until the length of the text is 87 letters long. Samples that exceeded 87 letters are discarded and another random number will be picked until a sequence of words that contained 87 characters is found. The random numbers generated are tracked to avoid the creation of duplicate samples.

5.3.2 Music Data

For music samples, MIDI files are transposed to the key of C major and only a single octave of notes is used. Three different durations are used for each notes. A duration of 0.5 represents anything shorter than a quarter note, a duration of 1 represents a quarter note, and a duration of 2 represents anything longer than a quarter note. Two different encodings are used for music.

The first encoding uses 51 unique symbols representing 3 durations of the 17 notes A, A-, A#, B, B-, C, C#, D, D-, D#, E, E-, F, F#, G, G-, and G#

The second encoding uses 24 unique symbols, the same amount of unique symbols that can be made using the Dorabella cipher symbols. This encoding only uses the 8 most frequent notes A, B, C, D, E, F, F#, and G along with 3 durations. Notes not in the 8 most frequent notes are moved half a step up or down. For example, a D# would be changed to a D and A- would be changed to A.

6 YouTube Experiments

The following experiments in this section are inspired by two interesting remarks made in a YouTube video by Keith Massey [4]. The first is that there are way too many mirrored symbols in the Dorabella cipher. The second is that there are long series of symbols in the Dorabella cipher where there is no repeat of a symbol with the same number of bumps.

Keith Massey claims that these two occurrences are statistically impossible and uses this as proof that the Dorabella cipher is a nonsensical message constructed as a playful joke.

6.1 Experimental setup: Mirrored Symbols

The Dorabella cipher contains 13 pairs of mirrored symbols. A mirrored pair is when 2 consecutive symbols have the same number of bumps but are facing in opposite directions. This experiment will test how likely it is for a piece of text enciphered with Dorabella symbols to contain 13 mirrored pairs.

1. Take 100000 samples of 87 length text from a piece of English text. The Adventures of Sherlock Holmes was used in this experiment.
2. For each sample, generate a random key that maps each letter in the sample to a Dorabella symbol. Since there are 26 letters in the alphabet but only 24 Dorabella symbols, 2 letters will share a symbol with another letter.
3. Encode each of our 100000 samples with Dorabella symbols.
4. Count the number of mirrored symbols that occur in each sample.

6.2 Results/discussion on mirrored symbols

The results from this experiment showed that 87 length English text enciphered with the Dorabella cipher should only average 3.64 mirrored pairs. Out of our 100000 samples, only 123 of them contained 13 or more mirrored pairs. This shows that a text with 13 mirrored pairs, like the Dorabella cipher, should only have about a 0.1% chance of occurring. These results confirm Keith Massey's claim and leads us to suspect that the mirrored pairs in the Dorabella cipher may have some additional meaning behind them. We aren't sure yet, but the mirrored symbols could have been used by Elgar to represent repeated letters, or they could have possibly been encoded as their own distinct letters.

6.3 Experimental setup: Longest Sequence

Each symbol in the Dorabella cipher is constructed using either 1, 2, or 3 bumps. In each of the three lines of the Dorabella cipher, there are long sequences of symbols without there being two symbols in a row containing the same number of bumps. The Dorabella cipher contains a sequence of 12 symbols in a row without two symbols in a row containing a repeated number of bumps. This experiment was ran to test how likely this is to happen with regular text.

1. Take 100000 samples of 87 length text from a piece of English text. The same 100000 samples from The Adventures of Sherlock Holmes used in the Mirrored Symbols experiment are used in this experiment.
2. For each sample, generate a random key that maps each letter in the sample to a Dorabella symbol. Since there are 26 letters in the alphabet but only 24 Dorabella symbols, 2 letters will share a symbol with another letter.

3. Encode each sample with Dorabella symbols.

4. Count the longest sequence of symbols without repeated bumps that occur in each sample.

6.4 Results/discussion on longest sequence

Our results from this experiment showed that 87 length English text enciphered with the Dorabella cipher has on average a sequence of 10.23 symbols in a row without two back to back symbols containing repeated number bumps. On top of that, 27472 out of the 100000 samples contained sequences of 12 or more symbols where there were no repeated bumps. Since English text has a 27.5% chance of containing sequences of 12 or more symbol, the Dorabella cipher having a sequence of 12 symbols in a row where there are no repeated bumps is not as improbable as Massey makes it out to be.

7 Decipherment Experiments

In our next set of experiments, we will be comparing the effectiveness of various substitution cipher solvers. We will compare the accuracy of the solvers and use the most accurate ones to obtain our best natural language and a music decipherment of the Dorabella cipher.

7.1 Experimental setup: Text Decipherment Programs

Four different decipherment programs were tested to determine the accuracy of each solver on 87 length texts without spaces. The HHK solver [1], Norvig solver [5], Unravel [7], and Anirudh's solver were tested.

We tested each of these solvers with 300 samples of 87 length text without spaces taken from Letters of Jane Austen. Each solver used a language model trained on Dangerous Connections which is a different corpus of letters that doesn't overlap with the training data. Since the Dorabella cipher was a message being sent from Elgar to Dora Penny, we assume that the Dorabella cipher might be a letter which is why we decided to use letters for our training and testing data. We believe that we will get the best decipherment of the Dorabella cipher if the train the solvers with corpus in the same domain as the cipher. Due to time constraints, we only tested 300 samples for each solver because some of the solvers would take hours to run.

For the decipherment tests, the samples used were all randomly enciphered with a substitution cipher.

7.2 Results/discussion on text decipherment programs

In these decipherment experiments we looked at two metrics for determining how good a solver performed, key accuracy and decipherment accuracy. Key accuracy is the proportion of the different cipher character types in the alphabet

Table 1: Solver Accuracy Dangerous Connections LM

Solver	**Key Acc**	**Dec Acc**	**Language Model**
HHK	43.1%	44.9%	English Letters
Norvig	75.8%	84.5%	English Letters
Norvig CLM	78.3%	88.1%	English Letters
Anirudh	69.0%	79.1%	English Letters
Det Unravel	42.8%	47.8%	English Letters

Table 2: Solver Accuracy New York Times LM

Solver	**Key Acc**	**Dec Acc**	**Language Model**
HHK	85.4%	88.7%	NYT
Det Unravel	27.7%	28.0%	NYT

which are mapped to their correct plaintext characters. Decipherment accuracy is the proportion of the total cipher characters in the cipher which are mapped to their correct plaintext characters.

Looking at our first solver, the HHK solver had a 44.9% decipherment accuracy when using Dangerous Connections to train the language model. This low accuracy is likely due to the small size of this corpus not being sufficient enough to train the WLM it uses. The HHK solver performed very well when trained on the larger NYT corpus.

The Norvig solver was fast and had a 84.5% accuracy when solving ciphers with no spaces. It was able to perform quite well without needing a large corpus for its training data. We also ran another test where we made a small modification to the Norvig solver to only use the character language model. The word language model was only used by the Norvig solver to determine where word boundaries go for ciphers without spaces. Turning off the WLM increased the accuracy of the Norvig solver to 88.1%. Like the HHK solver, this may be due to the solver's inability to create a strong WLM because of the small corpus size.

Anirudh's solver performed quite decently with a decipherment accuracy of 79.1%, however this was still a bit lower than the HHK and Norvig solvers' accuracy of around 88%.

Our final solver Unravel didn't perform very well on short ciphers without spaces with a decipherment accuracy of only 47.8%. Like the HHK solver, we also tried training the Unravel solver on the larger NYT corpus to see if this would make any difference. This lead to a even lower decipherment accuracy.

The following was the best decipherment we got using the Norvig CLM solver:

```
ychswamsopledieveeacceirprultmemarsofsheehaudmeleantdiroorltht
anthingutheandtuscutasirs
```

Assuming the Dorabella cipher is a substitution cipher written in English, the results from these experiments show that our best solvers should be able to accurately decipher more than 80% of the Dorabella cipher. However when the Norvig solver was run on Dorabella we are still left with something unreadable. The other solvers didn't produce anything much better. The fact that it can't solve the cipher could mean that the Dorabella cipher is not a simple substitution cipher or it might be written in a different language.

7.3 Experimental setup: Music Decipherment Programs

Two different decipherment programs were tested to determine the accuracy of our solvers on music. The Norvig solver and Anirudh's solver were chosen for in this test because the text decipherment experiments showed that these two solvers performed the best on short ciphers without spaces and without the need of a large training corpus.

One of the language models used for this experiment was trained with various pieces of music from Bach. The other language model was trained with various pieces of music from Elgar. The training corpus for the Bach LM was around 3 millions notes long and the training corpus for the Elgar LM was around 1 millions notes long. We used the Elgar LM to solve enciphered samples of Elgar music and the Bach LM to solve enciphered samples of Bach music.

In an attempt to provide a best case scenario and get a higher decipherment accuracy, the samples used for testing the solvers were around 20000 notes long. For the decipherment tests, the samples used were all randomly enciphered with a substitution cipher.

7.4 Results/discussion on music decipherment programs

Table 3: Solver Accuracy

Solver	**Key Acc**	**Dec Acc**	**Language Model**
norvig	7.0%	12.0%	Elgar
norvig	21.6%	30.4%	Bach
norvig	26.5%	32.0%	Bach 24
anirudh	4.8%	6.4%	Elgar
anirudh	15.6%	23.5%	Bach
anirudh	26.6%	32.7%	Bach 24

At first we tried replicating the same steps used in the text decipherment experiments. We test these two solvers with 87 length samples but the accuracy for both solvers were very low. We tried slowly increasing the length of the test samples to get a better decipherment accuracy, but eventually stopped at 20000 length samples as increasing the sample length gave us diminishing returns. Despite the samples being thousands of notes long, the solvers were only able to

get at best a 32.7% decipherment accuracy. The experiment shows that there may be something fundamentally different between our representations of music and natural language. This was an unexpected results and we may need to find a different way of representing music for our future experiments.

8 Language Identification Experiments

An important step in our attempt to decipher the Dorabella cipher is to first identify what language it is written in. We will be testing two of Hauer's [2] language identification programs, one that uses unigram counts and one that uses a trial decipherment method. The trial decipherment method was tested to have a 97% accuracy when using a training and testing set derived from the UDHR so it will be interesting to see what we get when testing it on the Dorabella cipher.

Running the unigram language identification program using the UDHR in multiple languages as training data, we get in descending order mam, cym, nyn, lug, and ndo as the top 5 languages. Running the trial decipherment program, we get lat, ace, eng, tpi, and sco as the top 5 languages.

Table 4: Top 5 Identified Languages

Rank	**Unigram Count**	**Trial Decipherment**
1	mam	lat
2	cym	ace
3	nyn	eng
4	lug	tpi
5	ndo	sco

8.1 Experimental setup: Language Identification

We will be running additional test to check the validity of our results. In this language identification experiment, we will start off by testing if the language identification programs can tell the difference between text and music. We want to see whether it thinks the Dorabella cipher is more likely to be English or music and we also want to test if the language identification program can tell the difference between Bach and Elgar music.

In this experiment, we ran the trial decipherment language identification on the Dorabella cipher, English, Bach, and Elgar. For English, Bach, and Elgar, 100 samples of 87 length text were tested and the average ranking of the 100 samples was recorded.

The Universal Declaration of Human Rights in multiple languages was used as training data for natural language and pieces of Bach music and Elgar music were used as training data for music. The English samples were taken from Letters of Jane Austen while the Music samples were taken from various Bach

and Elgar MIDIs. The testing and training data used in this experiment was all lowercase text with no spaces.

8.2 Results/discussion on language ID

Table 5: Trial Decipherment LangID Average Rank

Language	**Dora**	**Eng**	**Bach**	**Elgar**
Bach.tra	246	243.43	60.07	90.53
Elgar.tra	328	256.73	85.00	108.63
English.tra	3	57.00	61.27	67.63

Table 6: Unigram LangID Average Rank

Language	**Dora**	**Eng**	**Bach**	**Elgar**
Bach.tra	330	288.55	70.81	182.99
Elgar.tra	317	274.21	76.27	171.98
English.tra	61	82.84	160.45	163.35

In tables 5 and 6, the columns represent the testing data used as input and the rows represent the rank of the training data. The results in the Dorabella column are from a single sample while the English, Bach, and Elgar columns are the average rank of 100 samples. The results in this experiment show that English ranked quite high for the Dorabella cipher compared to music. One concerning point about these results however is that the language identification program thinks that English is only the 57th most likely language for our English test samples. We will take the results in these tables with a grain of salt for now while we look into why the program isn't properly identifying English.

8.3 Experimental setup: Language Identification 2

We know that the trial decipherment language identification program should have a 97% accuracy, so it is concerning to see how low the language identification accuracy for English was. As a follow up to the previous language identification experiment from section 8.1, we decided to perform some furthers tests on the language identification program using a smaller group of languages to verify the numbers we were getting.

Using a training and testing set derived from the Universal Declaration of Human Rights, we ran the following language identification experiments on test samples of these 5 languages English, French, Polish, Latin, and Italian as well as the English letters that scored poorly in the experiment above:

- Tested language identification with spaces in training and testing data

- Tested language identification with no spaces in training and testing data
- Tested language identification with 87 length testing data with spaces in training and testing data
- Tested language identification with 87 length text with no spaces in training and testing data

8.4 Results/discussion on language ID 2

Table 7: Top 5 Ranking Languages (Spaces in data)

Rank	**Eng**	**Fra**	**Pol**	**Lat**	**Ita**	**Letters**
1	eng	fra	pol	lat	ita	eng
2	sco	lnc	slk	lat_1	cos	sco
3	pcm	cat	tuk_latn	mxi	vec	pcm
4	swe	roh	hsb	ace	ido	mlt
5	afr	ast	azj_cyrl	cat	fur	tzm
Correct Rank	1	1	1	1	1	1

Table 8: Top 5 Ranking Languages (No Spaces in data)

Rank	**Eng**	**Fra**	**Pol**	**Lat**	**Ita**	**Letters**
1	eng	fra	pol	lat	ita	eng
2	sco	prv	ces	lat_1	cos	sco
3	pcm	lnc	azj_cyrl	tet	ido	pcm
4	ron_1993	sco	tuk_latn	ita	fur	als
5	als	eml	uig_arab	cha	vec	slv
Correct Rank	1	1	1	1	1	1

Table 9: Top 5 Ranking Languages (87 length Test sample with spaces in data)

Rank	**Eng**	**Fra**	**Pol**	**Lat**	**Ita**	**Letters**
1	eng	ron_1953	rus	lat	ita	lnc
2	dan	ron_2006	ssw	010	est	gle
3	cat	ron_1993	hrv	pon	swe	lat
4	lat	eng	lav	ina	mxi	aii
5	ron_2006	nob	bos_latn	tet	bug	lat_1
Correct Rank	1	27	6	1	1	73

In this experiment, we recorded the top 5 ranking languages that the language identification program returned. We also recorded the rank of the correct

Table 10: Top 5 Ranking Languages (87 length Test sample with no spaces in data)

Rank	**Eng**	**Fra**	**Pol**	**Lat**	**Ita**	**Letters**
1	sco	sot	hat_popu	eus	fin	sco
2	src	007	hrv	guu	ita	mkd
3	mah	umb	mlt	btb	est	nld
4	fra	fin	hat_krey	ceb	vmw	aka_asan
5	cym	ido	slv	src	krl	cjk_AO
Correct Rank	19	263	91	8	2	62

language for each test sample. This way we could see what rank the program assigned to the correct language if it wasn't in the top 5.

Looking at tables 7 and 8, our results showed that the language identification program was able to easily identify the correct language as rank 1 when sufficiently long test samples were used. Even after the spaces were removed, the program was still able to identify all the correct languages as rank 1.

It wasn't until we cut the length of the test samples down to 87 characters where we saw the program struggle a bit. Once we removed space from the 87 length samples, the program failed to identify any of the correct languages as rank 1. This is quite problematic as it would mean that none of our language identification results on 87 length text with no spaces are reliable.

8.5 Experimental setup: Language Identification 3

One of our guesses on why the language identification program performed so poorly aside from the short lengths of the testing data was due to the small size of the training data used. To test this theory, we ran another experiment using larger corpora for English, French, Polish, German, and Italian taken from Open Subtitles. In this experiment, instead of using 385 languages as training data, we only use five. This would make the task of language identification easier for the program since it would now only need to choose from 5 languages. Latin was dropped from the previous experiment in favor of German due to the difficulty in locating Latin text. These texts were split into a training set and a testing set for each language. A training and testing set of Bach music was also thrown into the mix to see what results we would get.

In this experiment, we ran this language identification experiment on 1000 test samples of each language and calculated the Average rank out of 5, the MRR, and the Top 1 accuracy for each language.

8.6 Results/discussion on language ID 3

From these results, we see that on average, the language identification program identifies the correct language as rank 2. We also see that it is very good at identifying music. It still isn't reaching the 97% top 1 percent levels as seen

Table 11: 87 length test data with no spaces (1000 test samples)

	En	**Fr**	**Pl**	**De**	**It**	**Letters**	**Bach**
Avg Lang Rank	2.01	2.02	1.91	1.99	1.35	1.92	1.29
MRR	0.68	0.68	0.72	0.69	0.87	0.70	0.93
Top 1 Accuracy	0.49	0.48	0.55	0.50	0.78	0.51	0.89

in the paper. This limitation of the language identification program prevents us from confidently using it to identify the correct language of the Dorabella cipher.

9 Results

Although we weren't able to solve the Dorabella cipher in this paper, we did to make some interesting discoveries. First, the mirrored symbols in the Dorabella cipher may have extra meaning. We believe this is something worth looking into and doing some follow up experiments on.

From our decipherment experiments, we've identified a couple of very reliable solvers for solving short ciphers without spaces. Our decipherment experiments concluded that the Dorabella cipher is not an English monoalphabetic substitution cipher. We are still leaning towards the idea that it is a substitution but maybe not in English.

Finally, our initial language identification experiments showed that the program thinks the Dorabella cipher looks more similar to English than it does to Bach or Elgar music. Out of the over 300 languages provided to the language identification program to choose from, it ranked English as the third most likely language while it ranked Bach and Elgar music as the 246th and 328th most likely.

Table 12: Trial Decipherment LangID Rank

Language	**Dorabella**
Bach.tra	246
Elgar.tra	328
English.tra	3

10 Conclusion

The best decipherment programs we tested were able to achieve higher than a 80% decipherment accuracy on English substitution ciphers. Despite this, we couldn't get a legible decipherment when the Dorabella cipher we fed into the

decipherment programs. Because of this, our conclusion is that the Dorabella cipher is either not a substitution cipher, not written in English, or both.

One of the big issues of deciphering the Dorabella cipher is its short length and lack of word boundaries. We tried to identify the language that the Dorabella cipher is written in however we discovered that our language identification program struggles on short texts without spaces. English was among the top results but we can't confirm how accurate that because of this limitation of the language identification program, we can't say how accurate this results is. When using the language identification program to compare music and natural language, the program thinks that the Dorabella cipher is more likely to be English than Bach or Elgar music.

References

[1] Bradley Hauer, Ryan Hayward, and Grzegorz Kondrak. Solving substitution ciphers with combined language models. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2314–2325, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[2] Bradley Hauer and Grzegorz Kondrak. Decoding anagrammed texts written in an unknown language and script. *Transactions of the Association for Computational Linguistics*, 4:75–86, 2016.

[3] Bill Manaris, Penousal Machado, Clayton McCauley, Juan Romero, and Dwight Krehbiel. Developing fitness functions for pleasant music: Zipf's law and interactive evolution systems. In *Proceedings of the 3rd European Conference on Applications of Evolutionary Computing*, EC'05, page 498507, Berlin, Heidelberg, 2005. Springer-Verlag.

[4] Keith Massey. The Dorabella cipher: Proven to be a friendly joke, Youtube, https://www.youtube.com/watch?v=13we3bi_npm, May 2017.

[5] Peter Norvig. Natural language corpus datazz. In Toby Segaran and Jeff Hammerbacher, editors, *Beautiful Data: The Stories Behind Elegant Data Solution*, pages 219–242. O'Reilly Media, July 2009.

[6] 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.

[7] Malte Nuhn, Julian Schamper, and Hermann Ney. UNRAVEL—A decipherment toolkit. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 549–553, Beijing, China, July 2015. Association for Computational Linguistics.

[8] 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*, pages 239–247, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.