Can Sequence-to-Sequence Models Crack Substitution Ciphers?

Nada Aldarrab and **Jonathan May**
University of Southern California
Information Sciences Institute
{aldarrab,jonmay}@isi.edu

Abstract

Decipherment of historical ciphers is a challenging problem. The language of the target plaintext might be unknown, and ciphertext can have a lot of noise. State-of-the-art decipherment methods use beam search and a neural language model to score candidate plaintext hypotheses for a given cipher, assuming plaintext language is known. We propose an end-to-end multilingual model for solving simple substitution ciphers. We test our model on synthetic and real historical ciphers and show that our proposed method can decipher text without explicit language identification and can still be robust to noise.

1 Introduction

Libraries and archives have many enciphered documents from the early modern period. Previous work has been done to make historical cipher collections available for researchers (Megyesi et al., 2020; Pettersson and Megyesi, 2019). Decipherment of classical ciphers is an essential step to reveal the contents of those historical documents.

In this work, we focus on solving 1:1 substitution ciphers. Current state-of-the-art methods use beam search and a neural language model to score candidate plaintext hypotheses for a given cipher (Kambhatla et al., 2018). However, this approach assumes that the target plaintext language is known. Hauer and Kondrak (2016) propose a method to identify the plaintext language of simple substitution ciphers. However, it has not been evaluated on an end-to-end decipherment experiment, and it is not clear how a two-step pipeline will perform when the plaintext language prediction is incorrect. The question is: Can we build an end-to-end model that deciphers directly without relying on a separate language identification (language ID) step?

The contributions of our work are:

- We propose an end-to-end multilingual decipherment model that can solve 1:1 substitution ciphers without explicit plaintext language identification, which we demonstrate on ciphers of 14 different languages.
- We conduct extensive testing of the proposed method in different decipherment conditions; different cipher lengths, no-space ciphers, and ciphers with noise.
- We apply our model on synthetic ciphers as well as a real historical cipher (the Borg cipher). We show that our multilingual model is robust and can crack the Borg cipher using the first 256 characters of the cipher.

2 The Decipherment Problem

Decipherment conditions vary from one cipher to another. In a **ciphertext-only attack**, the cryptanalyst only has access to the ciphertext. This means that the encipherment method, the plaintext language, and the key are all unknown.

In this paper, we focus on solving 1:1 substitution ciphers. We follow (Nuhn et al., 2013; Kambhatla et al., 2018) and use machine translation notation to formulate our problem. We denote the ciphertext with $f_1^N = f_1 \dots f_j \dots f_N$ and the plaintext with $e_1^M = e_1 \dots e_i \dots e_M$.[1]

In a **1:1 substitution cipher**, plaintext is encrypted into a ciphertext by replacing each plaintext character with a unique substitute according to a substitution table called the **key**. For example: the plaintext word "doors" would be enciphered to "KFFML" using the substitution table:

Cipher	Plain
K	d
F	o
M	r
L	s

[1] Unless there is noise or space restoration, $N = M$; see Sections 4.4 and 4.2.

The decipherment goal is to recover the plaintext given the ciphertext.

3 Decipherment Model

Inspired by character-level Neural Machine Translation (NMT), we view decipherment as a sequence-to-sequence translation task. The motivation behind using a sequence-to-sequence model is:

- It can be trained on multilingual data (Gao et al., 2020), making it potentially possible to have an end-to-end multilingual decipherment without relying on a separate language ID step.

- Due to transcription challenges of historical ciphers (section 4.4), ciphertext could be noisy. We would like the model to have the ability to recover from that noise by inserting, deleting, or substituting characters while generating plaintext. Generative models seem to be a good candidate for this task.

3.1 Decipherment as a Sequence-to-Sequence Translation Problem

In a ciphertext-only attack, all we have is ciphertext. We do not have any information about the key or target plaintext. To cast it as a translation task, we need pairs of $<f_1^N, e_1^M>$ to train on (which we do not have in this case). There are $26! \approx 4 \times 10^{26}$ possible keys for English ciphers, and our target is to solve any 1:1 substitution cipher at test time.

We create an artificial dataset of ciphers $<f_1^N, e_1^M>$ using randomly generated substitution keys (Figure 1a). We now have supervised training data, so we can train a sequence-to-sequence decipherment model. However, if we do so using the Transformer model described in Section 3.3, we get abysmal results (see Section 4.1 for scoring details). In retrospect it should be obvious why this is so. We do not expect the decipherment key at test time to be seen at training time and even if it were, the majority of other examples do not use the same key, so the model cannot learn a consistent behavior. In fact, since each training example uses a different key, we cannot assume that a character type has any particular meaning. The fundamental assumption behind embeddings is therefore broken. In the next section, we describe one way to overcome these challenges.

3.2 Frequency Analysis

To address the aforementioned challenges, we employ a commonly used trick in cryptanalysis: **frequency analysis**. Frequency analysis is attributed to the great polymath, Al-Kindi (801-873 C.E.) (Dooley, 2013). It is based on the fact that in a given text, letters and letter combinations (n-grams) appear in varying frequencies, and that the character frequency distribution is roughly preserved in any sample drawn from a given language. So, in pairs of $<f_1^N, e_1^M>$, we expect the frequency distribution of characters to be similar.

To encode that information, we rank ciphertext characters based on their frequencies (in descending order). Then, we map each cipher character to its frequency rank (Figure 1b). This way, we convert any ciphertext to a "frequency-encoded" cipher. The intuition is that by frequency encoding, we are reducing the number of possible substitution keys (assuming frequency rank is roughly preserved across all ciphers from a given language). This is only an approximation, but it helps restore the embedding assumption (that there is a coherent connection between a one-hot representation and its type embedding). For example, if the letters "e" and "i" are the most frequent characters in English, then in any 1:1 substitution cipher, they will be encoded as "0" or "1" instead of a randomly chosen character.

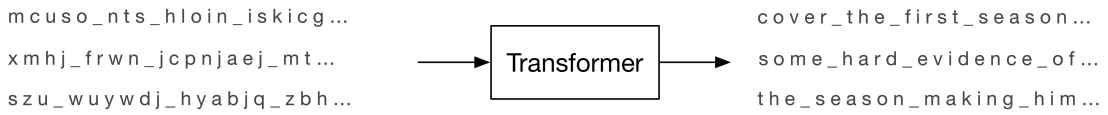
3.3 The Transformer

We follow the character-based NMT approach in Gao et al. (2020) and use the Transformer model (Vaswani et al., 2017) for our decipherment problem. The Transformer is an attention-based encoder-decoder model that has been widely used in the NLP community to achieve state-of-the-art performance on many sequence modeling tasks. We use the standard Transformer architecture, which consists of six encoder layers and six decoder layers as described in Gao et al. (2020).

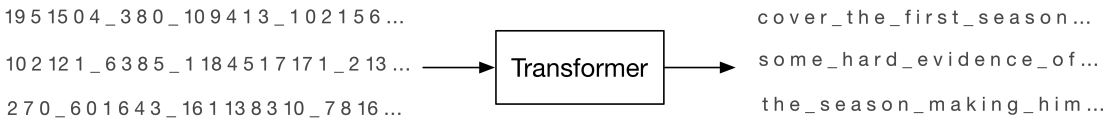
4 Experimental Evaluation

For training, we create 1:1 substitution ciphers for 14 languages using random keys. For English, we use English Gigaword (Parker et al., 2011). We scrape historical text from Project Gutenberg for the 13 other languages.[2] Appendix A summarizes

[2]Our data is available at https://github.com/NadaAldarrab/s2s-decipherment



(a) Input: example ciphers encoded in random keys. Output: plaintext in target language.



(b) Input: example ciphers encoded according to frequency ranks. Output: plaintext in target language.

Figure 1: Decipherment as a sequence-to-sequence translation problem. Figure 1a shows the original ciphers being fed to the model. Figure 1b shows the same ciphers after frequency encoding.

our dataset. We lowercase all characters and remove all non-alphabetic and non-space symbols. We strip accents for languages other than English.

To make our experiments comparable to previous literature (Kambhatla et al., 2018; Nuhn et al., 2013; Ravi and Knight, 2008), we create test ciphers from the English Wikipedia article about History.[3] We use this text to create ciphers of length 16, 32, 64, 128, and 256 characters. We generate 50 ciphers for each length. We follow the same pre-processing steps for training data.

We carry out 4 sets of experiments to study the effect of cipher length, space encipherment/removal, unknown plaintext language, and transcription noise. Finally, we test our models on a real historical cipher, whose plaintext language was not known until recently.

As an evaluation metric, we follow previous literature and use Symbol Error Rate (SER). SER is the fraction of symbols in the deciphered text that are not correct. For space restoration experiments (section 4.2), we use Translation Edit Rate (TER) (Snover et al., 2006), but on the character level. Thus, we define character-level TER as:

$$\text{TER} = \frac{\text{\# of edits}}{\text{\# of reference characters}} \quad (1)$$

where possible edits include the insertion, deletion, and substitution of single characters. When the ciphertext and plaintext have equal lengths, SER is equal to TER.

We use Fairseq to train our models (Ott et al., 2019). We use the same hyperparameters used in (Gao et al., 2020) for character NMT. The only difference is that we set the maximum batch size to 10K tokens and use half precision floating point computation for faster training. We train for 20 epochs. Unless otherwise stated, we use 2M example ciphers to train, 3K ciphers for tuning, and 50 ciphers for testing in all experiments. We report the average SER on the 50 test ciphers of each experiment.

[3] https://en.wikipedia.org/wiki/History

4.1 Cipher Length

We first ran an experiment on ciphers of length 256 using the approach described in section 3.1 (i.e. we train a Transformer model on pairs of $<f_1^N, e_1^M>$ without frequency encoding). As expected, the model was not able to crack the 50 test ciphers, resulting in an SER of 71.75%. For the rest of the experiments in this paper, we use the frequency encoding method described in section 3.2.

Short ciphers are more challenging than longer ones. Following previous literature, we report results on different cipher lengths using our method. Table 1 shows decipherment results on ciphers of length 16, 32, 64, 128, and 256. Kambhatla et al. (2018) use beam search and a neural language model to score candidate plaintext hypotheses for each cipher. They also add a frequency matching heuristic to the scoring function, which gives the results in Table 1. Our model achieves comparable results to their method and leads to perfect decipherment for ciphers of length 256. In addition, our method provides the ability to train on multilingual data, which we use to attack ciphers with an unknown plaintext language as described in section 4.3.

4.2 No-Space Ciphers

Spaces make decipherment easier because word boundaries can give a strong clue to the cryptanalyst. In many historical ciphers, however, spaces are hidden. For example, in the Copiale cipher, spaces are enciphered with special symbols just like

	Cipher Length				
	16	**32**	**64**	**128**	**256**
Beam NLM (Kambhatla et al., 2018)	26.80	5.80	0.07	0.01	0.00
Beam (NLM + FreqMatch) (Kambhatla et al., 2018)	31.00	2.90	0.07	0.02	0.00
Transformer + Freq (this work)	20.00	5.56	1.66	0.34	0.00

Table 1: SER (%) for solving 1:1 substitution ciphers of lengths 16, 32, 64, 128, and 256 using our decipherment method.

other alphabetic characters (Knight et al., 2011). In other ciphers, spaces might be omitted, like the Zodiac-408 cipher (Nuhn et al., 2013). We test our method in four scenarios:

1. Ciphers with spaces (comparable to Kambhatla et al. (2018)).

2. Ciphers with enciphered spaces. In this case, we treat space like other cipher characters during frequency encoding as described in section 3.2.

3. No-space ciphers. We omit spaces in both (source and target) sides.

4. No-space ciphers with space recovery. We omit spaces from source but keep them on the target side. The goal here is to train the model to restore spaces along with the decipherment.

Table 2 shows results for each of the four scenarios on ciphers of length 256. During decoding, we force the model to generate tokens to match source length. We notice that the method is robust to both enciphered and omitted spaces. In the last scenario, where the model is expected to generate spaces and thus the output length differs from the input length, we limit the model to output exactly 256 characters, but we allow the model freedom to insert spaces where it sees fit. The model generates spaces in accurate positions overall, leading to a TER of 1.88%. We leave other length control methods for future work.

4.3 Unknown Plaintext Language

For some ciphers, plaintext language might be unknown. To solve that problem, we train a single multilingual model on 14 different languages; Catalan, Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Latin, Norwegian, Portuguese, Spanish, and Swedish. Training data and pre-processing is explained in Appendix A. We train on a total of 2.1M random ciphers of length 256. We report results as the number of training languages increase while keeping the total number of 2.1M training examples fixed (Table 3). Increasing the number of languages negatively affects performance, as we expected. However, our experiments show that our 14-language model is still able to decipher 700 total test ciphers with an average SER of 0.68%. Since we are testing on 256-character ciphers, this translates to no more than two errors per cipher (on average).

Cipher Type	**TER(%)**
Ciphers with spaces	0.00
Ciphers with enciphered spaces	0.00
No-space ciphers	0.77
No-space ciphers + generate spaces	1.88

Table 2: TER (%) for solving 1:1 substitution ciphers of length 256 with different spacing conditions.

4.4 Transcription Noise

Real historical ciphers can have a lot of noise. This noise can come from the natural degradation of historical documents, human mistakes during a manual transcription process, or misspelled words by the author, as in the Zodiac-408 cipher. Noise can also come from automatically transcribing historical ciphers using Optical Character Recognition (OCR) techniques (Yin et al., 2019). It is thus crucial to have a robust decipherment model that can still crack ciphers despite the noise.

Hauer et al. (2014) test their proposed method on noisy ciphers created by randomly corrupting $log_2(N)$ of the ciphertext characters. However, automatic transcription of historical documents is very challenging and can introduce more types of noise, including the addition and deletion of some characters during character segmentation (Yin et al.,

# lang	ca	da	nl	en	fi	fr	de	hu	it	la	no	pt	es	sv	avg
14	0.34	1.29	0.79	0.25	0.20	0.20	0.41	0.64	1.52	1.43	0.41	0.69	0.72	0.70	0.68
7	-	-	-	0.08	-	0.34	0.30	-	1.23	1.38	-	0.48	0.40	-	0.60
3	-	-	-	0.04	-	0.23	-	-	-	-	-	-	0.39	-	0.29

Table 3: SER (%) for solving 1:1 substitution ciphers using a multilingual model trained on a different number of languages. Each language is evaluated on 50 test ciphers generated with random keys.

	Noise Type	
% Noise	sub	sub, ins, del
5	1.10	2.87
10	2.40	5.87
15	5.28	10.58
20	11.48	16.17
25	17.63	27.43

Table 4: TER (%) for solving 1:1 substitution ciphers with random insertion, deletion, and substitution noise. These models have been trained with 10% noise.

2019). Thus, we test our model on three types of random noise: insertion, deletion, and substitution. We experiment with different noise percentages for ciphers of length 256 (Table 4). We report the results of training (and testing) on ciphers with only substitution noise and ciphers that have all three types of noise (divided equally). We experimentally find that training the models with 10% noise gives the best overall accuracy, and we use those models to get the results in Table 4. Our method is able to decipher with up to 84% accuracy on ciphers with 20% of random insertion, deletion, and substitution noise.

4.5 The Borg Cipher

The Borg cipher is a 400-page book digitized by the Biblioteca Apostolica Vaticana.[4] It was first automatically cracked by Aldarrab (2017) using the noisy-channel framework described in Knight et al. (2006).

We trained a Latin model on 1M ciphers and used the first 256 characters of the Borg cipher to test our model. Our model was able to decipher the text with an SER of 3.91%. We also tried our 14-language multilingual model on this cipher, which was able to decipher it with an SER of 5.47%.

5 Related Work

Deciphering substitution ciphers is a well-studied problem in the natural language processing community, e.g., (Ravi and Knight, 2008; Corlett and Penn, 2010; Nuhn et al., 2013, 2014; Hauer et al., 2014; Aldarrab, 2017). Many of the proposed methods search for the substitution table (i.e. cipher key) that leads to a target plaintext that scores high according to a character n-gram language model. The current state-of-the-art method uses a neural language model to score candidate hypotheses from the search space (Kambhatla et al., 2018). This method assumes prior knowledge of the target plaintext language. Our method, by contrast, can solve substitution ciphers from different languages without explicit language identification.

Recent research has looked at applying other neural models to different decipherment problems. Greydanus (2017) find an LSTM model can learn the decryption function of polyalphabetic substitution ciphers when trained on a concatenation of <key + ciphertext> as input and plaintext as output. Our work looks at a different problem. We target a ciphertext-only-attack for short 1:1 substitution ciphers. Gomez et al. (2018) propose CipherGAN, which uses a Generative Adversarial Network to find a mapping between the character embedding distributions of plaintext and ciphertext. This method assumes the availability of plenty of ciphertext. Our method, by contrast, does not require a large amount of ciphertext. In fact, all of our experiments were evaluated on ciphers of 256 characters or shorter.

6 Conclusion and Future Work

In this work, we present an end-to-end decipherment model that is capable of solving simple substitution ciphers without the need for explicit language identification. We use frequency analysis to make it possible to train a multilingual Transformer model for decipherment. Our method is able to decipher 700 ciphers from 14 different languages with less than 1% SER. We apply our method on the Borg cipher and achieve 5.47% SER using the multilingual model and 3.91% SER using a monolingual Latin model. To the best of our knowledge, this is the first application of sequence-to-sequence

[4] http://digi.vatlib.it/view/MSS_Borg.lat.898.

neural models for decipherment.

We hope that this work drives more research in the application of contextual neural models to the decipherment problem. It would be interesting to develop other techniques for solving more complex ciphers, e.g. homophonic and polyalphabetic ciphers.

References

Nada Aldarrab. 2017. Decipherment of historical manuscripts. Master's thesis, University of Southern California.

Eric Corlett and Gerald Penn. 2010. 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. Association for Computational Linguistics.

John F. Dooley. 2013. *A Brief History of Cryptology and Cryptographic Algorithms*. Springer International Publishing.

Yingqiang Gao, Nikola I. Nikolov, Yuhuang Hu, and Richard H.R. Hahnloser. 2020. Character-level translation with self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1591–1604, Online. Association for Computational Linguistics.

Aidan N. Gomez, Sicong Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and Lukasz Kaiser. 2018. Unsupervised cipher cracking using discrete gans.

Sam Greydanus. 2017. Learning the enigma with recurrent neural networks. *CoRR*, abs/1708.07576.

Bradley Hauer, Ryan Hayward, and Grzegorz Kondrak. 2014. 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. Dublin City University and Association for Computational Linguistics.

Bradley Hauer and Grzegorz Kondrak. 2016. Decoding anagrammed texts written in an unknown language and script. *TACL*, 4:75–86.

Nishant Kambhatla, Anahita Mansouri Bigvand, and Anoop Sarkar. 2018. Decipherment of substitution ciphers with neural language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 869–874, Brussels, Belgium. Association for Computational Linguistics.

Kevin Knight, Beáta Megyesi, and Christiane Schaefer. 2011. 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. Association for Computational Linguistics.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. 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. Association for Computational Linguistics.

Beáta Megyesi, Bernhard Esslinger, Alicia Fornés, Nils Kopal, Benedek Láng, George Lasry, Karl de Leeuw, Eva Pettersson, Arno Wacker, and Michelle Waldispühl. 2020. Decryption of historical manuscripts: the decrypt project. *Cryptologia*, 44(6):545–559.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. 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. Association for Computational Linguistics.

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

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. Gigaword fifth edition LDC2011T07.

Eva Pettersson and Beata Megyesi. 2019. Matching keys and encrypted manuscripts. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 253–261, Turku, Finland. Linköping University Electronic Press.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order N-gram models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 812–819, Honolulu, Hawaii. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Xusen Yin, Nada Aldarrab, Beata Megyesi, and Kevin Knight. 2019. Decipherment of historical manuscript images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 78–85.

A Multilingual Data

We create a dataset of historical text for 13 different languages scraped from Project Gutenberg, namely: Spanish, Latin, Hungarian, Danish, Norwegian, Dutch, Swedish, Catalan, French, Portuguese, German, Italian and Finnish. We use English Gigaword (Parker et al., 2011) for English. Table 5 summarizes our datasets. We lowercase all characters and remove all non-alphabetic characters. We strip accents for languages other than English.

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
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
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 5: Summary of data sets obtained from Project Gutenberg.