

Decipherment of Substitution Ciphers with Neural Language Models

by

Nishant Kambhatla

M.Sc. (Integrated) in Software Engineering,
Vellore Institute of Technology, 2016

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

© Nishant Kambhatla 2018
SIMON FRASER UNIVERSITY
Summer 2018

Copyright in this work rests with the author. Please ensure that any reproduction
or re-use is done in accordance with the relevant national copyright legislation.

Approval

Name: Nishant Kambhatla

Degree: Master of Science (Computing Science)

Title: Decipherment of Substitution Ciphers with Neural Language Models

Examining Committee:

Chair: Keval Vora
Assistant Professor

Anoop Sarkar
Senior Supervisor
Professor

Fred Popowich
Supervisor
Professor

David Alexander Campbell
Internal Examiner
Associate Professor
Department of Statistics and Actuarial Science

Date Defended: July 26, 2018

Abstract

The decipherment of homophonic substitution ciphers using language models (LMs) is a well-studied task in Natural Language Processing (NLP). Previous work in this topic score short local spans of possible plaintext decipherments using n -gram LMs. The most widely used technique is the use of beam search with n -gram LMs proposed by Nuhn et al. (2013). We propose a new approach on decipherment using a beam search algorithm that scores the entire candidate plaintext at each step with a neural LM. We augment beam search with a novel rest cost estimation that exploits the predictive power of a neural LM. This work, to our knowledge, is the first to use a large pre-trained neural language model for decipherment. Our neural decipherment approach outperforms the state-of-the-art n -gram based methods on many different ciphers. On challenging ciphers such as the Beale cipher our system reports significantly lower error rates with much smaller beam sizes.

Keywords: Natural Language Processing; decipherment; neural decipherment; neural language models; beam search

Dedication

To Mom, Dad and Dr. Anoop Sarkar!

Acknowledgements

First and foremost, I would like to express my gratitude towards my advisor Dr. Anoop Sarkar. This thesis would not have been possible without his vision, guidance and encouragement throughout the course of the project. His brilliant insights were pivotal to the completion of this work. I owe my accomplishments as a Master's student to his understanding, patience and constant support.

I am extremely grateful to my supervisor, Dr. Fred Popowich, for his patient advice and valuable comments to improve the thesis. I am also grateful to Dr. Dave Campbell for the perceptive discussion and his valuable insights which helped shape the final version of this thesis.

I would like to thank my fellow Natural Language Lab colleagues, especially Jetic Gu, Anahita Mansouri, Andrei Vacariu, Hassan Shavarani, Ashkan Alinejad, Golnar Sheikshab and Maryam Siahbani. Discussions with them were always interesting and insightful, and made the place filled with computers, books and deadlines seem full of fun.

I wouldn't have survived my time here without the constant support of Pratik Gujjar, Akash Abdujyoti, Srikanth Muralidharan, Royal Sequeira and Aniket Mane - my friends, philosophers and guides.

I shall forever remain indebted to my family. I am truly fortunate to have been blessed with amazing parents who have always supported me with their unconditional love and have pushed me to embrace opportunities.

Table of Contents

Approval	ii
Abstract	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Definitions	2
1.1.1 1:1 Substitution Ciphers	3
1.1.2 Homophonic Substitution Ciphers	5
1.2 Motivation	5
1.3 Contribution	6
1.4 Overview	6
2 Related Work	8
3 Decipherment Model	10
3.1 Language Model	10
3.1.1 Neural Language Model	11
3.2 Beam Search	13
3.2.1 Algorithm for Decipherment	14
4 Score Estimation	16
4.1 Baseline	17
4.2 New Improved Rest Cost Estimation	17
4.2.1 Frequency Matching Heuristic	18

4.3	Summary	19
5	Experimental Evaluation	21
5.1	Experimental Setup	21
5.2	Deciphering 1:1 Substitution Ciphers	22
5.3	Deciphering Homophonic Ciphers	23
5.3.1	A Simple Cipher: Zodiac-408	24
5.3.2	A Difficult Cipher: Beale Pt 2	25
5.3.3	Reference Plaintexts	28
5.3.4	Summary	28
6	Conclusion	30
	Bibliography	31

List of Tables

Table 1.1	Substitution table for simple substitution cipher.	3
Table 5.1	Symbol Error Rates (SER) and average Run Times (RT) in seconds based on Neural Language Model and beam size (Beam) for solving 1:1 substitution ciphers of lengths 64 and 128, respectively. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with frequency matching heuristic.	22
Table 5.2	Homophonic ciphers used in our experiments.	23
Table 5.3	Symbol Error Rates (SER) based on Neural Language Model and beam size (Beam) for deciphering Zodiac-408, respectively. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with the frequency matching heuristic.	25
Table 5.4	Symbol Error Rates (SER) based on Neural Language Model and beam size (Beam) for deciphering Part 2 of the Beale Cipher. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with the frequency matching heuristic.	26

List of Figures

Figure 1.1	An example of Simple Substitution cipher.	2
Figure 1.2	A historic homophonic cipher used in our experiments [<i>Wikisource</i>]	4
Figure 1.3	An example Homophonic Cipher.	5
Figure 3.1	Character NLM in action on the word 'zodiac'.	11
Figure 5.1	Symbol error rates for decipherment of 1:1 substitution ciphers of different lengths. The beam size is 100k. Beam 6-gram model is [26].	23
Figure 5.2	Deciphering the Zodiac-408 cipher.	24
Figure 5.3	Our decipherment of Beale cipher Pt-2.	26
Figure 5.4	Beale cipher Pt-2.	27
Figure 5.5	Reference plaintexts for Zodiac-408 and Beale Pt-2 ciphers used for evaluation.	28

Chapter 1

Introduction

The term *decipherment* is used to describe the process of converting an encoded text or a signal into a comprehensible normal language text. In cryptanalysis, decipherment means decryption - the process of retrieving the original data from its encrypted form. Decipherment in archaeology and historical linguistics means producing an interpretation of ancient scripts and languages.

For thousands of years, codes have been used by many, including the military, scientists, criminals and secret societies, to maintain secrecy. These encoded messages or *ciphertexts* are obfuscated forms of the original texts obtained by systematically operating on the constituent symbols by a process called *encipherment*. Decipherment subverts the sanctuary that encipherment creates by reversing the process to obtain the original *plaintext*. Philologists and archaeologists have spent lifetimes attempting to decipher several ancient scripts and lost languages. Despite computers being nonexistent at the time, many such scripts and languages have been deciphered, thanks to decades of man-hours spent on these undertakings. A crowning achievement in the area was the decipherment of Egyptian hieroglyphic writings during early 1820s [1]. Besides the discovery of Rosetta Stone in 1799, this success was a culmination of efforts of historians and literary scientists for centuries. In cryptanalysis, Alan Turing's code breaking machine *Bombe* which played a crucial role in cracking the Enigma code - an encryption system employed by the German forces for secure communication during World War II - was a significant breakthrough. Turing's invention of this deciphering-machine reduced the duration of the war by at least 2 years, thereby saving millions of lives [10], and has been pivotal in the advancement of computing.

In 1947 Warren Weaver, a mathematician and one of the pioneers of machine translation (MT), suggested that an article written in Russian can be really perceived as written in English but encoded with strange symbols, and that one would simply decode the text to learn its meaning [36, 22]. The cryptographic decipherment approach for machine translation of natural languages was one of the first envisioned applications of modern computing [17]. State-of-the-art machine translation approaches rely on copious amounts of parallel text to estimate translation models. While these methods are effective, it is difficult to find parallel

data in every domain or for every language pair. Though computational decipherment has been explored to offset this drawback [34, 25], translation accuracies of such techniques are still low.

The application of decipherment in different areas of research makes it an interesting venue to explore further. A robust decipherment system is crucial to making decipherment approaches functional in practical real-world applications.

1.1 Definitions

Substitution ciphers are encrypting algorithms which operate by replacing the constituent tokens of a plaintext with ciphertext units, typically following a mapping scheme (ϕ) or a set of defined rules.

Using general notations from MT [26], consider ciphertext $f_1^N = f_1 \dots f_i \dots f_N$ and plaintext $e_1^N = e_1 \dots e_i \dots e_N$ consisting of vocabularies $f_i \in V_f$ and $e_i \in V_e$ respectively. Mathematically, substitution cipher is formulated as a function $\phi(e, f)$ that computes probability of mapping a given ciphered symbol f to a context symbol e . We call a table with these probabilities of mapping of a plaintext token with a ciphertext token a *substitution table*.



Figure 1.1: An example of Simple Substitution cipher.

There are different types of substitution ciphers: 1:1 letter substitution, homophonic, polygraphic, mechanical and poly-alphabetic. Polygraphic ciphers substitute a plaintext token with multiple ciphertext tokens. Mechanical ciphers logically select a plaintext token to be substituted using a rotating letter disk (example: the Enigma machine). Polyalphabetic ciphers use multiple substitution alphabets.

Figure 1.1 shows a simple substitution cipher written in *Alienese*, the language of the aliens, borrowed from an episode (fair use screenshot) of the sci-fi cartoon tv-show *Futurama*TM. Even though the language is made-up, we are able to successfully decipher it into plain English (solution in subsection 1.1.1).

1.1.1 1:1 Substitution Ciphers

Also referred to as **simple substitution cipher**, in 1:1 substitution cipher each plaintext character is replaced with a unique ciphertext symbol. This type of ciphers is simple and can be generally solved using frequency analysis techniques. Consider the following decipherment key :

A	B	C	D	E	F	G	H	I	J	K	L	M
↓	≤	↯	⌘	⋈	⊞	⌘	⌘	⌘	⌘	⌘	⌘	⌘
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
⊙	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

In this key, the letters of English alphabet are plaintext tokens $e \in V_e$ and beneath them are the corresponding ciphertext symbols $e \in V_e$ which substitute them. Mathematically, the mapping function $c(e|f)$ assigns a value of 1 to the right plaintext-ciphertext pairs and 0 for the rest.

	A	B	C	D	E	..
↓	1	0	0	0	0	..
↯	0	0	1	0	0	..
≤	0	1	0	0	0	..
⌘	0	0	0	1	0	..
⋈	0	0	0	0	1	..
..

Table 1.1: Substitution table for simple substitution cipher.

Using the key, the Alienese text $\text{⌘⋈⋈⋈⋈ ⌘⋈⋈⋈⋈⋈⋈ ⌘⋈⋈⋈⋈⋈⋈ ⌘⋈⋈⋈⋈⋈⋈}$ shown in Figure 1.1 can be deciphered to **USED HUMAN PROBES**.

peace, contract alliances, establish commerce, and to do all other acts and things which independent States may of right do. And for the support of this declaration, with a firm reliance on the protection of Divine Providence, we mutually pledge to each other our lives, our fortunes, and our sacred honor.

The letter, or paper, so often alluded to, and marked "2," which is fully explained by the foregoing document, is as follows.

115, 73, 24, 807, 37, 52, 49, 17, 31, 62, 647, 22, 7, 15, 140, 47, 29, 107, 79, 84, 56, 239, 10, 26, 811, 5, 196, 308, 85, 52, 160, 186, 59, 211, 36, 9, 46, 316, 554, 122, 106, 95, 53, 58, 2, 42, 7, 35, 122, 53, 31, 82, 77, 250, 196, 56, 96, 118, 71, 140, 287, 23, 353, 37, 1003, 65, 147, 807, 24, 3, 8, 12, 47, 43, 59, 807, 45, 316, 101, 41, 78, 154, 1003, 122, 133, 191, 16, 77, 49, 102, 57, 72, 34, 73, 85, 35, 371, 59, 196, 81, 93, 191, 106, 273, 60, 394, 620, 270, 220, 106, 388, 287, 63, 3, 6, 191, 122, 43, 234, 400, 106, 290, 314, 47, 48, 81, 96, 26, 115, 92, 158, 191, 110, 77, 85, 197, 46, 10, 118, 140, 353, 48, 120, 106, 2, 607, 61, 420, 811, 29, 125, 14, 30, 37, 103, 28, 248, 16, 159, 7, 35, 19, 301, 125, 110, 486, 287, 98, 117, 511, 62, 51, 220, 37, 113, 140, 807, 138, 540, 8, 44, 287, 388, 117, 16, 79, 344, 34, 20, 59, 511, 543, 107, 603, 220, 7, 66, 154, 41, 20, 50, 6, 575, 122, 154, 248, 110, 61, 52, 33, 30, 5, 38, 8, 14, 84, 57, 540, 217, 115, 71, 29, 84, 63, 43, 131, 29, 138, 47, 73, 239, 540, 52, 53, 79, 118, 51, 44, 63, 196, 12, 239, 112, 3, 49, 79, 353, 105, 56, 371, 557, 211, 505, 125, 360, 133, 143, 101, 15, 284, 540, 252, 14, 205, 140, 344, 26, 811, 138, 115, 48, 73, 34, 205, 316, 607, 63, 220, 7, 52, 150, 44, 52, 10, 40, 37, 158, 807, 37, 121, 12, 93, 10, 13, 33, 12, 131, 62, 115, 102, 807, 49, 53, 135, 138, 30, 31, 62, 67, 41, 85, 63, 10, 106, 807, 138, 8, 113, 20, 32, 33, 37, 353, 287, 140, 47, 85, 50, 37, 49, 47, 64, 6, 7, 71, 33, 4, 43, 47, 63, 1, 27, 600, 208, 230, 15, 101, 246, 85, 94, 511, 2, 270, 20, 39, 7, 33, 44, 23, 40, 7, 10, 3, 811, 106, 44, 486, 230, 353, 211, 200, 31, 10, 38, 140, 297, 61, 603, 320, 302, 666, 297, 2, 44, 33, 32, 511, 548, 10, 6, 250, 557, 246, 53, 37, 52, 83, 47, 320, 38, 33, 807, 7, 44, 30, 31, 250, 10, 15, 33, 106, 160, 113, 31, 102, 406, 230, 540, 320, 29, 66, 33, 101, 807, 138, 301, 316, 353, 320, 220, 37, 52, 28, 540, 320, 33, 8, 48, 107, 50, 811, 7, 2, 113, 73, 16, 125, 11, 110, 67, 102, 807, 33, 59, 81, 158, 38, 43, 581, 138, 19, 85, 400, 38, 48, 77, 14, 27, 8, 47, 138, 63, 140, 44, 35, 22, 177, 106, 250, 314, 217, 2, 10, 7, 1003, 4, 20, 25, 44, 48, 7, 36, 46, 110, 230, 807, 191, 34, 112, 117, 41, 110, 121, 125, 96, 41, 51, 50, 140, 56, 47, 152, 540, 61, 807, 28, 42, 250, 138, 582, 98, 643, 32, 107, 140, 112, 26, 85, 138, 540, 53, 20, 125, 371, 38, 36, 10, 52, 118, 136, 102, 420, 150, 112, 71, 14, 20, 7, 24, 18, 12, 807, 37, 67, 110, 62, 33, 21, 93, 220, 511, 102, 811, 30, 83, 84, 305, 620, 15, 2, 108, 220, 106, 333, 105, 106, 60, 275, 72, 8, 50, 205, 185, 112, 125, 540, 65, 106, 807, 138, 96, 110, 16, 73, 37, 807, 150, 409, 400, 50, 154, 285, 96, 106, 316, 270, 205, 101, 811, 400, 8, 44, 37, 52, 49, 241, 34, 205, 38, 16, 46, 47, 85, 24, 44, 15, 64, 73, 138, 807, 85, 78, 110, 33, 420, 505, 53, 37, 38, 22, 31, 10, 110, 106, 101, 140, 13, 38, 3, 5, 44, 7, 98, 287, 135, 150, 96, 33, 84, 125, 807, 191, 96, 511, 118, 440, 370, 643, 466, 106, 41, 107, 603, 220, 275, 30, 150, 105, 49, 53, 287, 250, 208, 134, 7, 53, 12, 47, 83, 63, 138, 110, 21, 112, 140, 485, 486, 503, 14, 73, 84, 375, 1003, 150, 200, 16, 42, 5, 4, 25, 42, 8, 16, 811, 125, 160, 32, 205, 603, 807, 81, 96, 405, 41, 600, 136, 14, 20, 23, 26, 338, 302, 246, 8, 131, 160, 140, 84, 440, 42, 16, 811, 40, 67, 101, 102, 194, 138, 205, 51, 93, 241, 540, 122, 8, 10, 63, 140, 47, 48, 140, 288.

Figure 1.2: A historic homophonic cipher used in our experiments [Wikisource]

1.1.2 Homophonic Substitution Ciphers

A homophonic cipher is a more advanced cipher. It uses an encryption scheme which substitutes each plaintext token with one or more than one cipher symbol. Thus, a plaintext token $e \in V_e$ is mapped to multiple symbols $f \in V_f$, which often makes it difficult to crack.

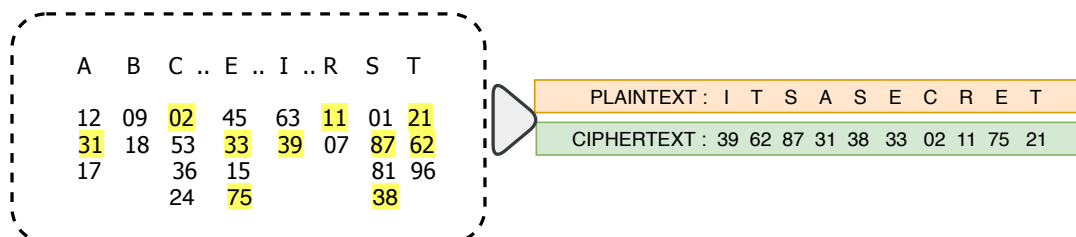


Figure 1.3: An example Homophonic Cipher.

The encipherment process for this type of substitution involves mapping a key to different symbols. This makes homophonic ciphers non-deterministic. Figure 1.1.2 shows an example homophonic cipher. Note how the E is disguised as 33 and 75 at different positions in the ciphertext. This non-determinism makes homophonic ciphers much harder than 1:1 ciphers as guessing just one substitution option doesn't lead to the solution.

1.2 Motivation

Breaking substitution ciphers recovers the *plaintext* from a *ciphertext* that uses a 1:1 or homophonic cipher key. Previous work using pre-trained language models (LMs) for decipherment use n -gram LMs [33, 26]. Some methods use the Expectation-Maximization (EM) algorithm [19] while most state-of-the-art approaches for decipherment of 1:1 and homophonic substitution ciphers use beam search and rely on the clever use of n -gram LMs [27, 15]. Newly proposed neural LMs can globally score the entire candidate plaintext sequence [24]. However, using a neural LM for decipherment is not trivial because scoring the entire partially-deciphered plaintext candidate is computationally challenging. A partially decoded text has many undeciphered symbols as well. The scoring function in state-of-the-art beam search technique employs an n -gram language model to evaluate the partial outputs with *maximum available* context, or local context, based on some heuristic assumption. An n -gram language model can hold a maximum context window of n . A neural language model, on the other hand, has a richer context. The improvements on beam search algorithm in this decipherment system allow for it to use large pre-trained neural LM to score the partial decipherments with global context.

1.3 Contribution

In this thesis, we introduce a novel method for solving substitution ciphers using neural language models and beam search. Our contributions are as follows:

- We present the first automatic-decipherment system to use a large pre-trained neural language model.
- We propose an improvement over the beam search for decipherment introduced by Nuhn et al.[26] that solves both 1:1 and homophonic letter substitution ciphers effectively.
- Our new method seeks to complete a partially-deciphered text by sampling the missing plaintext tokens from a neural language model, implicitly enabling the entire sequence to be scored globally as beam search progresses. This provides a robust decipherment system that counterbalances the drawbacks faced by the previous methods using n -gram language models.
- We introduce a second variant of the scoring function in beam search by augmenting it with a frequency matching heuristic that estimates the correlation between a ciphertext symbol and a plaintext token. This technique seems to contribute positively towards decipherment accuracies.

1.4 Overview

This thesis introduces a simple and powerful method to obtain the plaintext from an unreadable text enciphered with a substitution cipher automatically.

In **Chapter 2**, we discuss previous work on decipherment, primarily the approaches for solving substitution ciphers. The aim of this chapter is to summarize the literature in computational decipherment while introducing the techniques with which we juxtapose our system.

In **Chapter 3**, we describe our decipherment model. Our algorithm solves monophonic (1:1) and homophonic substitution ciphers with a pre-trained source language model (LM). The chapter is divided into two sections: (1) neural language model to find the most probable deciphered text; (2) beam search algorithm that finds the best scoring mapping (ϕ) for the deciphered text.

In **Chapter 4**, we detail the score estimation function that is crucial to beam search algorithm. We discuss the baseline local scoring method and our novel global scoring method that works with neural language model.

In **Chapter 5**, we report the experiments conducted on various 1:1 and homophonic ciphertexts, both synthetic and historic, their evaluation and results. We apply our auto-

mated decipherment model on the Zodiac 408 and Beale Pt-2 ciphers, and record interesting results.

We conclude our work in **Chapter 6**. We summarize our contributions, and the experimental results and findings in this chapter. We show that our simple neural language model powered decipherment system is very effective and quite comparable to the current state-of-the-art.

Chapter 2

Related Work

One of the earliest known methods employed to decipher substitution ciphers was based on frequency analysis. The development of this technique is accredited to the great polymath, Al-kindī, who gave the description of this first cryptanalysis method in the 9th century [2]. Since then several methods relying on number theory have been studied to automate decryption or decipherment. Differential cryptanalysis [6] is one such technique which investigates the differences in ciphertext through a network of transformations to find any non-random behaviour of symbols in the cipher. Biryukov et al. [7] later developed a cipher-only attack which applies a series of methods to break a code, assuming no prior knowledge about the ciphertext. Bleichenbacher et al. [8] presented a technique that uses a guessing approach to replace ciphertext tokens at random on a unit or a segment level. In a broader sense, these techniques used mathematical models to solve ciphers with no prior knowledge about the plaintext.

The initial works on using linguistic information in decipherment systems were predominantly based on dictionary attacks. In 1979 Peleg and Rosenfield [30] used dictionaries to find cleartext patterns within ciphertext with a *relaxation algorithm* to solve substitution ciphers. Hart et al. [14] proposed a statistical model that used a large dictionary to mine symbol patterns and augment the language model. [18, 28] had further employed heuristic search based on word dictionaries to find the cipher key.

Knight and Yamada [20] used a statistical method based on Expectation Maximization (EM) algorithm to decipher unidentified foreign writing scripts with phonetic models of familiar languages. Their model was able to learn the interconnection of written tokens (cipher), and phonemes (plaintext). In another work, Knight et al. [19] proposed a fully unsupervised technique based on EM for breaking substitution ciphers. The method relied on a character n -gram language model (LM) to obtain the most *probable* decipherment. The EM computes plaintext-to-ciphertext symbol mapping probabilities, then the Viterbi algorithm is used as a decoder to produce the output sequence. Berg-Kirkpatrick and Klein [5] showed that a large number of random restarts improved the EM approach in the decipherment setting, yielding high accuracies.

Ravi and Knight [33] introduced a novel Bayesian approach for decipherment, and reported the first automatic decipherment of the Zodiac-408 cipher using a trigram LM and a word dictionary. Ravi and Knight [32] framed the decipherment problem as an integer linear programming (ILP) problem. Corlett and Penn [11] presented an efficient A* search algorithm to solve letter substitution ciphers. This method is capable of working with longer ciphers than the ILP approach. While A* search produces optimal solution, it is computationally complex in terms of time and space, which makes it less suitable for solving very long ciphers.

Nuhn et al. [26] produced better results on 1:1 and homophonic letter substitution ciphers in shorter time compared to ILP and EM-based decipherment methods by employing a higher order language model and an iterative beam search algorithm, thus becoming the state of the art for solving substitution ciphers. The beam search allowed for solving one symbol at a time, building the search space into a tree, such that exactly k symbols are deciphered at the k -th level of the tree. Using a character 6-gram LM to quantify the probabilities of partial solutions, beam search keeps only the n best scoring solutions and *prunes* the rest. Nuhn et al. [27] further presented various improvements and extensions to this beam search algorithm by introducing an improved rest cost estimation and an optimized strategy to arrange the order in which the cipher symbols are deciphered. This is the current state of the art. Our approach investigates these two methods almost exhaustively and directly builds on top of the decipherment models given by Nuhn et al. [26, 27]. With extensions to the model within the same framework, our system compares against its predecessors on deciphering substitution ciphers. For shorter cryptograms, Hauer et al. [15] proposed a novel approach for solving mono-alphabetic substitution ciphers which combined character-level and word-level language model. They formulated decipherment as a tree search problem, and used Monte Carlo Tree Search (MCTS) as an alternative to beam search. Their approach is the best for short ciphers.

In NLP, the effectiveness of neural network models for decipherment has not been thoroughly investigated. Greydanus [13] framed the decryption process as a sequence-to-sequence translation task and use a deep LSTM-based model to learn the decryption algorithms for three poly-alphabetic ciphers including the Enigma cipher. However, their approach needs supervision compared to our approach which uses a pre-trained neural LM. Gomez et al. [12] (CipherGAN) used a generative adversarial network to learn the mapping between the learned letter embedding distributions in the ciphertext and plaintext. They apply this approach to shift ciphers (including Vigenère ciphers). Their approach cannot be extended to homophonic ciphers and full message neural LMs as in our work. To our knowledge, our work is the first to apply neural networks based approach for decipherment of natural language.

Chapter 3

Decipherment Model

The aim of decipherment is to transform an unreadable *ciphertext* into the original human comprehensible *plaintext*. To describe the problem formally, consider ciphertext $f_1^N = f_1..f_i..f_N$ and plaintext $e_1^N = e_1..e_i..e_N$ which consist of vocabularies $f_i \in V_f$ and $e_i \in V_e$ respectively. The beginning tokens in the ciphertext (f_0) and plaintext (e_0) are set to “\$” denoting the beginning of a sentence. The substitutions are represented by a function $\phi : V_f \rightarrow V_e$ such that 1:1 substitutions are *bijective* while homophonic substitutions are *general*. A cipher function ϕ which does not have every $\phi(f)$ fixed is called a partial cipher function [11]. The number of f s that are fixed in ϕ is given by its cardinality. ϕ' is called an extension of ϕ , if f is fixed in ϕ' such that $\delta(\phi'(f), \phi(f))$ yields true $\forall f \in V_f$ which are already fixed in ϕ where δ is the Kronecker delta function [37].

The objective of automated **decipherment** is then to find the ϕ that maximizes the probability of the deciphered text:

$$\hat{\phi} = \arg \max_{\phi} p(\phi(f_1)...\phi(f_N)) \quad (3.1)$$

Here, $p(\cdot)$ is the language model (**LM**). In practice, argmax function is approximated using a beam search algorithm [26] which incrementally finds the most likely substitutions using the language model scores as the ranking.

3.1 Language Model

Language modelling is a very important task in natural language processing. A language model (LM) is an approach to statistically specify the model of a natural language through a lot of examples. A statistical language model is a probability distribution over words or characters. Traditionally, a statistical LM is trained using large amounts of monolingual data, to make probabilistic prediction of the subsequent element in a sequence given a set of elements that precede it. These elements could be either words or characters. The statistical n -gram language models make an estimate of the likelihood of the element being of each

certain value given the preceding $n - 1$ elements. The goal of a language model (LM) is to estimate the probability distribution of a sequence of words or characters. It can, therefore, assign a probability to an entire given sequence of characters/words.

For example, consider a sequence $x_{1:T} = x_1, x_2, x_3, \dots, x_T$. An aforementioned LM models the likelihood of the sequence of characters $p(x_{1:T})$ as:

$$\begin{aligned}
 p(x_{1:T}) &= p(x_1, x_2, x_3, \dots, x_T) \\
 p(x_{1:T}) &= p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_2, x_1) \dots p(x_T|x_{T-1} \dots x_1) \\
 p(x_{1:T}) &= \prod_{i=1}^T p(x_i | x_1, x_2, \dots, x_{i-1}) \\
 p(x_{1:T}) &= \prod_{i=1}^T p(x_i | x_{<i})
 \end{aligned} \tag{3.2}$$

Language models are pivotal to solving the problem of decipherment, since we can use them to guide the search algorithm to the possible optimal solution. Thus, in equation (3.1) the probability distribution from an LM can be treated as an objective function to find the cipher key which gives the most probable decipherment.

3.1.1 Neural Language Model

As opposed to a statistical LM, a neural LM models language sequence probability using neural network representations. Bengio et al. [3] proposed a feed-forward neural network language model which operates on a fixed number of elements at each step. The major drawback in this approach is that a feed forward network has to use fixed-length context that needs to be specified ad hoc before training. A solution to this issue is using Recurrent Neural Networks (RNNs). RNNs can incorporate any number of elements as possible thus allowing an unlimited context. By using recurrent connections, information persists inside these networks for arbitrarily long time [9]. Mikolov et al. [23] proposed the simplest form of optimized RNN language model. The main advantage of an RNN LM, or a neural LM in general, over a statistical n -gram LM is that it can compute the log-likelihood of the entire candidate plaintext for a partial hypothesis during decipherment. In this work, we use a state-of-the-art byte (character) level neural LM using a multiplicative LSTM [31].

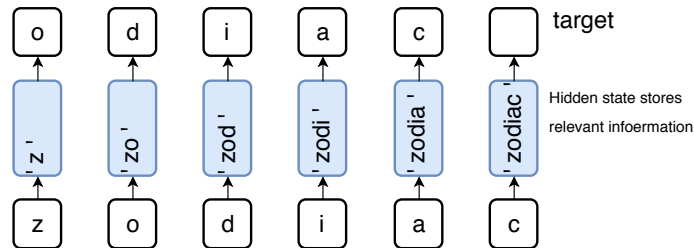


Figure 3.1: Character NLM in action on the word 'zodiac'.

Multiplicative RNNs or mRNNs allow flexible input-dependent transitions. They perform better than traditional RNNs on language modelling tasks [35]. Although generative RNNs measure the log-likelihoods of variable length sequences accurately, the vanishing gradient problem renders RNNs hard to train [4]. The long short-term memory (LSTM) architecture [16] was proposed to address this issue. LSTM adds additional units to model long term and short term memory separately. So during backpropagation, the gradient is less likely to vanish. Both LSTM and mRNN present multiplicative units. But the architecture of an LSTM allows it to regulate the information flow through the network, whereas in an mRNN the transition functions can vary across inputs. In other words, LSTM and mRNN have complementary architecture. The multiplicative LSTM (mLSTM) [21] is a hybrid architecture that amalgamates the factorized hidden-to-hidden transitions of mRNNs with the gating structure of LSTMs.

The demonstrated success of mLSTMs on character level language modelling [21, 31] motivated us to use mLSTM network to train our language model. For mLSTM LM, mRNN and LSTM are combined by adding connections to the intermediate state m_t to the gates of LSTM. With inputs from the input layer x_t and the previous hidden state h_t , intermediate state m_t in mLSTM is calculated using this equation:

$$m_t = (W_{mx}x_t) \odot (W_{mh}h_{t-1}) \quad (3.3)$$

Using the state m_t , the hidden state, and the sigmoid input, output and forget gates of the mLSTM are formulated as:

$$\begin{aligned} \hat{h}_t &= W_{hx}x_t + W_{hm}m_t \\ i_t &= \sigma(W_{ix}x_t + W_{im}m_t) \\ o_t &= \sigma(W_{ox}x_t + W_{om}m_t) \\ f_t &= \sigma(W_{fx}x_t + W_{fm}m_t) \end{aligned} \quad (3.4)$$

At each time step, the internal cell state is updated using the input and forget gates.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(\hat{h}_t) \quad (3.5)$$

The final output of the hidden state at step t is then given as:

$$h_t = \tanh(c_t) \odot o_t \quad (3.6)$$

The gated units of LSTM control the complex transitions in mRNN that result from the factorized hidden weight matrix. The sigmoid input and forget gates in the LSTM units allow more flexible input-dependent transition functions than in regular mRNNs. Assume we have a sequence $x_{1:T} = x_1, x_2, x_3, \dots, x_T$. To measure the likelihood of the sequence, we

calculate the probability of $x_{1:T}$ using equation 3.2 as:

$$p(x_1, x_2, \dots, x_T) = \prod_{i=1}^{i=T} p(x_i \mid x_{<i})$$

With x_t as the current input, the updated final output of the current hidden state (h_t) from equation 3.6 is used to predict the probability distribution over the next subsequent element in the sequence as a function of \mathbf{g} :

$$p(x_{t+1}) = \text{softmax}(\mathbf{g}(h_t)) \quad (3.7)$$

The function \mathbf{g} applies a matrix multiplication and then softmax normalization is performed on h_t to convert it into a valid probability distribution.

Let $\text{SCORE}()$ denote the scoring function based on the neural language model. Given a sequence $x_{1:T} = x_1 \dots x_T$, $\text{SCORE}(x_{1:T})$ determines its score as the negative log likelihood of the sequence:

$$\text{SCORE}(x_{1:T}) = -\log(p(x_{1:T}))$$

Simplifying this further from equation 3.2:

$$\begin{aligned} \text{SCORE}(x_{1:T}) &= -\log \left(\prod_{i=1}^T p(x_i \mid x_{<i}) \right) \\ &= -\sum_{i=1}^T \log(p(x_i \mid x_{<i})) \end{aligned} \quad (3.8)$$

where the probability $p(.)$ is computed using Equation 3.7. The above equation 3.8 shows the language model score ($\text{SCORE}()$) obtained on any given sequence. Since this is a neural language model, it considers the entire stream of characters to produce the score.

3.2 Beam Search

Beam search is a heuristic search algorithm that combines breadth-first search (BFS) and child pruning techniques to build its search space into a tree. At each level it produces the subsequent state of all the current states and ranks them using the scoring (or cost) function. It stores only k best options at each depth and *prunes* the rest, thus reducing the search space as it expands. Beam search is most oftenly used to maintain tractability in large systems with insufficient amount of resources to store the entire search tree. In a decipherment model, beam search finds a sequence of plain text letters by using the ciphertext based on the plaintext language model.

Consider the ciphered sequence ‘<×~≈ ∩×≈<×’ of length n . Deciphering this to English would mean the plaintext vocabulary size $|V_e|$ is 26. This would require searching across a space of 26 candidate maps for each cipher symbol $f \in V_f$. Performing a search through whole space of 26 letters would result in a high space complexity of $\mathcal{O}(26^n)$. Using beam search can help prune the candidate plaintext tokens based on their LM score. If we choose a beam size of top k candidate tokens scored by LM, the resulting search would only need a space of $\mathcal{O}(k^n)$ - a significant improvement in terms of complexity. If the language model is powerful enough, with any luck, the above code can be deciphered to ‘LOST WORLD’.

3.2.1 Algorithm for Decipherment

Algorithm 1 is the general framework of the beam search algorithm introduced by Nuhn et al. [26] for solving substitution ciphers. As an overview, it tracks all partial hypotheses by maintaining two lists: H_s and H_t , based on their quality. As the search progresses, the partial hypotheses are extended, scored with **SCORE** and appended to H_t . **SCORE(.)** scores a hypothesized partial decipherment using probabilities from the plaintext language model. The number of fixed ciphertext symbols is given by **CARDINALITY**. **EXT_LIMITS** determines which extensions should be allowed and **EXT_ORDER** picks the next cipher symbol for extension. **HISTOGRAM_PRUNE** then returns the best scoring subset with k_{best} hypotheses from the new partial hypotheses by sorting them according to their scores determined by **SCORE(.)**. The search continues after pruning: $H_s \leftarrow \text{HISTOGRAM_PRUNE}(H_t)$.

Algorithm 1 Beam Search for Decipherment by Nuhn et al. [26]

```

1: function (BEAM_SEARCH (EXT_ORDER, EXT_LIMITS))
2:   initialize sets  $H_s, H_t$ 
3:   CARDINALITY = 0
4:    $H_s.\text{ADD}((\emptyset, 0))$ 
5:   while CARDINALITY <  $|V_f|$  do
6:      $f = \text{EXT\_ORDER}[\text{CARDINALITY}]$ 
7:     for each  $\phi \in H_s$  do
8:       for each  $e \in V_e$  do
9:          $\phi' := \phi \cup \{(e, f)\}$ 
10:        if EXT_LIMITS( $\phi'$ ) then
11:           $H_t.\text{ADD}(\phi', \text{SCORE}(\phi'))$ 
12:        HISTOGRAM_PRUNE( $H_t$ )
13:        CARDINALITY = CARDINALITY + 1
14:         $H_s = H_t$ 
15:         $H_t.\text{CLEAR}()$ 
16:   return WINNER( $H_s$ )

```

Note that `EXT_LIMITS()` chooses which partial to leave out. This is based on the constraints of the cipher function ϕ .

$$\text{MAP_LIMIT} = \begin{cases} 1 & \text{for 1:1} \\ \max_e \sum_f \phi(f|e) & \text{for Homophonic} \end{cases}$$

where `MAP_LIMIT` is the constraint on maximum number of $f \in V_f$ that can map to plaintext token e . `EXT_ORDER` is the order in which the cipher symbols will be fixed. This is a list of the symbols sorted according to their unigram-frequencies, meaning the most frequent symbol will be first operated on. This helps pruning out the deceptive hypotheses early from the search span. `SCORE` is the most central function of the algorithm that determines if a partial hypothesis should be kept or not by determining its quality based on a plaintext LM.

We augment this algorithm by updating the `SCORE` function with a neural LM. The next chapter describes the improved score function that is capable of working with a neural language model.

Chapter 4

Score Estimation

Score estimation evaluates the quality of the partial hypotheses ϕ at each step as the beam search progresses. In this chapter, we discuss the baseline scoring function and introduce a novel scoring function that works with neural language model for the beam search algorithm used for decipherment.

Consider $V_e = \{a, b, c, d\}$ for plaintext vocabulary and $V_f = \{A, B, C, D\}$ for ciphertext vocabulary. We assume an extension order of (B, A, C, D) . We now take a ciphertext

ABDDCABCDADCABDC

Let $\phi = \{(a, A), (b, B)\}$ be the partial hypothesis, meaning at this point only A and B are converted to plaintext tokens. Then $\text{SCORE}(\phi)$ scores this hypothesized partial decipherment using pre-trained language model (section 3.1.1) in the hypothesized plaintext language.

For better insights, let's look at the following demonstration of the scoring process. Consider this example [27] with vocabularies $V_e = \{a, b, c, d\}$ and $V_f = \{A, B, C, D\}$, extension order (B, A, C, D) , and ciphered text

$$f_1^N = \$ \text{ ABDD CABCD ADCABDC } \$$$

We also have $\phi = \{(a, A), (b, B)\}$ as the partial hypothesis which returns the partial deciphered text

$$\phi(f_1^N) = \$ \text{ ab...ab...a...ab... } \$$$

With only these two locations deciphered, the score estimation (SCORE) can calculate the hypothesis score only based on this partially interpreted sequence.

4.1 Baseline

The initial rest cost estimator introduced in [26] computes the score of hypotheses only based on partially deciphered text that builds a shard of n adjacent solved symbols. As a heuristic, n -grams which still consist of unsolved cipher-symbols are assigned a trivial estimate of probability 1. An improved version of rest cost estimation [27] consults lower order n -grams to score each position. This would produce a score of:

$$p(a \mid \$) \cdot p(b \mid a) \cdot 1^3 \cdot p(a) \cdot p(b \mid a) \cdot 1^2 \cdot p(a) \cdot 1^2 \cdot p(a) \cdot p(b \mid a) \cdot 1^2$$

Meaning, with n -grams as low as unigrams chipping in to score, each symbol is scored with most context available.

Though this method of scoring is computationally efficient and demonstrates good results, the scoring method greatly relies on local context, *i.e.* the estimation is strictly based on partial character sequences. The unigram-probabilities here are static and do not depend on the context at all. When estimating the parameters of an n -gram model, only the context of $(n - 1)$ words is considered. More specifically, the true conditional probability under Markov assumption is not modelled and, therefore, context dependency beyond the window is ignored. Thus, attempting to utilize a higher amount of context can lower the probability of some tokens. During decipherment, this drawback prevents the model from being able to estimate the contribution of a single character towards the structure of the entire sequence.

4.2 New Improved Rest Cost Estimation

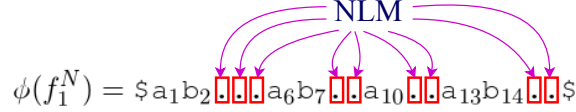
The baseline scoring method greatly relies on local context, *i.e.* the estimation is strictly based on partial character sequences. Since this depends solely on the n -gram LM, the true conditional probability under Markov assumption is not modeled and, therefore, context dependency beyond the window of $(n - 1)$ is ignored. Thus, attempting to utilize a higher amount of context can lower the probability of some tokens resulting in poor scores.

We address this drawback with a new improved version of the rest cost estimator by supplementing the partial decipherment $\phi(f_1^N)$ with predicted plaintext text symbols using our neural language model (NLM). Using the same example as above with ciphertext \$ ABDDCABCDAD CABDC and $\phi = \{(a, A), (b, B)\}$. The partial hypothesis constructed by [26, 27] and given a SCORE would be:

$$\phi(f_1^N) = \$a_1b_2\dots a_6b_7\dots a_{10}\dots a_{13}b_{14}\dots \$$$

The subscripts indicate the position of token. For example, a_6 implies that token is at the 6th position in the sequence. We introduce a scoring function that is able to score the

entire plaintext including the missing plaintext symbols. First, we sample the plaintext symbols from the NLM at all locations depending on the deciphered tokens from the partial hypothesis ϕ such that these tokens maintain their respective positions in the sequence, and at the same time are sampled from the neural LM to fit (probabilistically) in this context. The char-level sampling is done incrementally from left to right to generate a sequence that contains the deciphered tokens from ϕ at the exact locations they occur in the above $\phi(f_1^N)$. If the LM prediction contradicts the hypothesized decipherment we stop sampling and start from the next character.



Next, we determine the probability of the entire sequence including the scores of sampled plaintext as our rest cost estimate. In our running example, this would yield a score estimation of:

$$\phi(f_1^N) = \$ a_1 b_2 \textcolor{red}{d}_3 \textcolor{red}{c}_4 \textcolor{red}{c}_5 a_6 b_7 \textcolor{red}{c}_8 \textcolor{red}{d}_9 a_{10} \textcolor{red}{d}_{11} \textcolor{red}{d}_{12} a_{13} b_{14} \textcolor{red}{d}_{15} \textcolor{red}{c}_{16} \$$$

Thus, the neural LM is used to predict the score of the full sequence (Eqn. 3.8). Thus, in any iteration, given a ϕ with the $\text{SCORE}(\phi)$, the extension ϕ' (Algo. 1) is scored as:

$$\text{SCORE}(\phi') = \text{SCORE}(\phi) + \text{NEW}(\phi') \quad (4.1)$$

where **NEW** is the score for symbols that have been newly fixed in ϕ' while extending ϕ to ϕ' .

Since the NLM is just a distribution of probabilities, sampling characters from the NLM is stochastic. Sampling is done to reconstruct the whole candidate decipherment from a partial hypothesis every time the algorithm finds an extension ϕ' to the hypothesis ϕ . These missing characters would need to be resampled at each update. Note that it isn't a random cipher but a random sample of letters to fill the spaces. It need not be consistent with the actual ciphertext.

This method of global scoring evaluates each candidate partial decipherment by scoring the entire message, augmented by the sampled plaintext symbols from the NLM. Since more terms participate in the rest cost estimation with global context, we use the plaintext LM to provide us with a better rest cost in the beam search. In this approach, even with a few symbols decoded, we need a much smaller beam size.

4.2.1 Frequency Matching Heuristic

Homophonic ciphers are non-deterministic in nature as they allow multiple cipher symbols to be mapped to one plaintext token. The vocabulary of cipher symbols is much larger

than the number of plaintext tokens. For example, Zodiac-408 and Beale Pt-2 ciphers each have vocabulary size of 54 and 180 compared to that of 26 of the English alphabet. Plain frequency analysis approaches on homophonic ciphers have been shown to be generally ineffective. Therefore, it is interesting to explore if the non-determinism in homophones can be formulated probabilistically.

In the field of morphological analyses, more specifically, morphological alignment, Yarowsky et al. [38] addressed the problem of determining the correct past tense of a verb (sing/sang vs sing/singed) with a corpus based frequency similarity approach. This problem statement is similar to the one discussed above. This is because it is inappropriate to determine the association between an inflection and its candidate root form by relative frequencies alone as many inflections are comparatively rare and appear considerably fewer times than its root. Lemma alignment by frequency similarity assumes two forms belong to the same lemma by quantifying how well the relative frequency in the corpus fits the expected distribution.

Motivated by their approach, we introduce a frequency matching heuristic that quantitatively determines if a ciphertext symbol and a plaintext token could be actually mapped to each other with an absolute numerical value. This estimator can be described as:

$$\text{FMH}(\phi') = \left| \log \left(\frac{\nu(f)}{\nu(e)} \right) \right| \quad f \in V_f, \quad e \in V_e \quad (4.2)$$

$\nu(f)$ is the percentage relative frequency of the ciphertext symbol f , while $\nu(e)$ is the percentage relative frequency of the plaintext token e in the plaintext language model. The closer this value to 0, the more likely it is that f is mapped to e .

We augment our original scoring function by simply adding this to the score. Thus, in any iteration, given a ϕ with the $\text{SCORE}(\phi)$, we now compute the score of extension ϕ' (Algo. 1) with the augmented function:

$$\text{SCORE}(\phi') = \text{SCORE}(\phi) + \text{NEW}(\phi') - \text{FMH}(\phi') \quad (4.3)$$

where NEW is the score for symbols that have been newly fixed in ϕ' while extending ϕ to ϕ' .

This method of global scoring estimates each candidate partial decipherment by scoring the entire message, with sampled plaintext symbols from the NLM, substantiated with the relative-frequency matching heuristic. We use this as an additional method of scoring to explore if the statistical correlation between a ciphertext symbol and a plaintext token can be helpful in decipherment.

4.3 Summary

The $\text{SCORE}()$ function is crucial to determining the quality of a partial hypothesis ϕ as the beam search advances. We introduce two novel methods of scoring. The first method is **global rest cost estimation** which scores the entire partially-deciphered message by

sampling missing plaintext tokens from the NLM. The second method augments the first one with a **frequency matching heuristic** (FMH) that quantifies the correlation between a ciphertext and a plaintext symbol.

Experiments show these methods contribute positively towards the accuracy on different decipherment tasks.

Chapter 5

Experimental Evaluation

In this chapter, we describe the experiments we performed to evaluate our decipherment method. We carry out two sets of experiments: one on letter based 1:1 ciphers, and another on homophonic substitution ciphers.

We report **Symbol Error Rate** (SER) to determine the quality of the deciphered text given a reference mapping ϕ_{ref} and a candidate mapping ϕ .

Given a set of characters V_{eval} , we calculate SER as:

$$SER = 1 - \sum_{v \in V_{eval}} \frac{N(v)}{N_{eval}} \cdot \delta(\phi(v), \phi_{ref}(v)) \quad (5.1)$$

where $N(v)$ is the unigram count for $v \in V_{eval}$, and total number of symbols at hand, $N_{eval} = \sum_{v \in V} N(v)$. SER returns the fraction of characters in the deciphered text that are incorrect.

5.1 Experimental Setup

Plaintext Language Model: The character-level neural language model (NLM) uses a single layer multiplicative LSTM (mLSTM) with 4096 units. The model was trained for a single epoch on mini-batches of 128 subsequences of length 256 for a total of 1 million weight updates. States were initialized to zero at the beginning of each data shard and persisted across updates to simulate full-backprop and allow for the forward propagation of information outside of a given sub-sequence.

Data: We trained the plaintext (NLM) on the English Gigaword [29] corpus augmented with a short corpus of plaintext letters of about 2000 words authored by the Zodiac killer¹. The Gigaword corpus is a collection of English newswire text (New York Times, Washington Post, Bloomberg News etc.) consisting of about 1200 million words. Tailoring the data for

¹https://en.wikisource.org/wiki/Zodiac_Killer_letters

our plaintext NLM further, we lowercase the entire corpus, and remove all punctuations and special characters. To build an LM simulating the text conditions in the ciphers, we make sure that our data consists of lower-cased letters from English alphabet and numbers only. We use the same NLM for all our decipherment experiments.

Beam Search: We carry out all experiments with two versions of the beam search. The first one is the beam search algorithm with global scoring function (section 4.2). The second version of the search has global scoring function with a frequency matching heuristic (section 4.2.1). For all experiments, both results are reported separately.

5.2 Deciphering 1:1 Substitution Ciphers

In this experiment we use a synthetic 1:1 letter substitution cipher dataset from [26]. The text is from English Wikipedia articles about history², pre-processed by stripping the text of all images, tables, then lower-casing all characters, and removing all non-alphabetic and non-numeric characters. We create 50 cryptograms for each length 16, 32, 64, 128 and 256 using a random Caesar-cipher 1:1 substitution.

Length	Beam	SER(%) 1	SER(%) 2	RT(sec)
64	100	4.14	4.14	1.40
	1000	1.09	1.04	4.91
	10000	0.08	0.12	41.29
	100000	0.07	0.07	384.16
128	100	7.31	7.29	1.47
	1000	1.68	1.55	5.60
	10000	0.15	0.09	44.75
	100000	0.01	0.02	441.11

Table 5.1: Symbol Error Rates (SER) and average Run Times (RT) in seconds based on Neural Language Model and beam size (Beam) for solving 1:1 substitution ciphers of lengths 64 and 128, respectively. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with frequency matching heuristic.

Each of the 50 cryptograms of each length is deciphered using our model in two settings - one using beam search with global scoring, and the other using beam search with global scoring and frequency matching heuristic - each using different beam sizes. We report the symbol error rate on the decipherments obtained in the first setting as SER1 and those obtained in the second setting as SER2.

The decipherment results on 1:1 letter substitution ciphers of length 64 and 128 are shown in Table 5.1. Each of the scores reported in the table is the average SER obtained on deciphering 50 ciphers. Evidently, our method obtains close-to-perfect decryption of the

²<http://en.wikipedia.org/wiki/History>

longer mono-alphabetic letter substitution ciphers. However, on shorter ciphertexts of length 16 and 32, we obtain slightly inaccurate mappings resulting in higher symbol error rates.

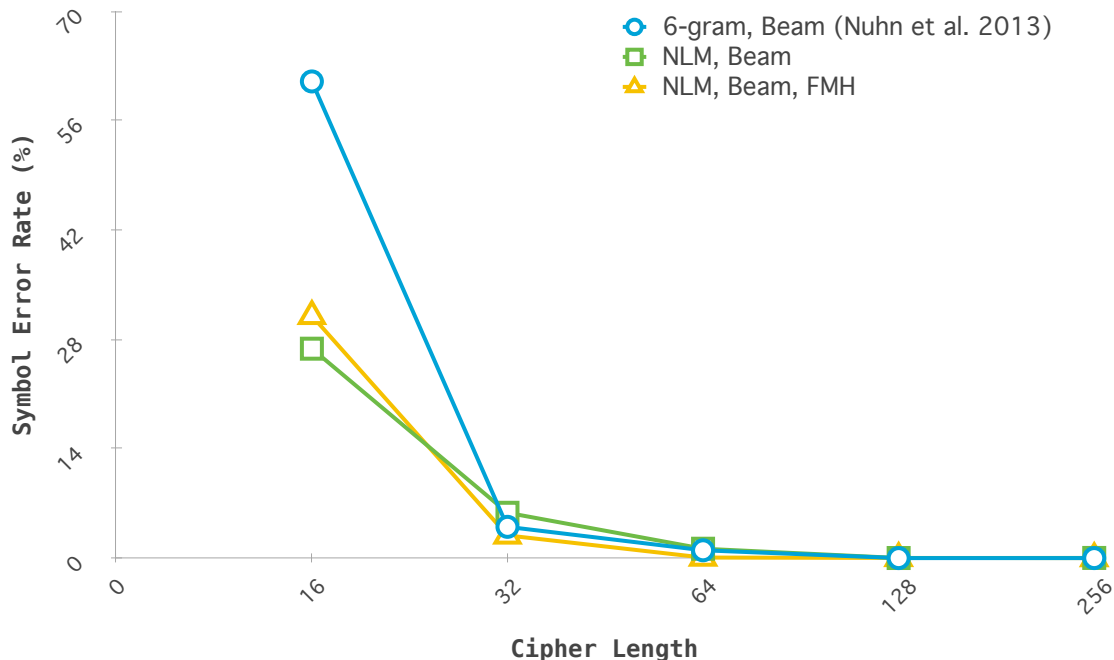


Figure 5.1: Symbol error rates for decipherment of 1:1 substitution ciphers of different lengths. The beam size is 100k. Beam 6-gram model is [26].

Fig 5.1 plots the results of our method for cipher lengths of 16, 32, 64, 128 and 256 alongside Beam 6-gram (the best performing model) model from [26]. It can be inferred from the accuracies that the effectiveness of the sampling technique depends directly of the length of the ciphertext. For shorter cipher lengths (16, 32) the sampling has a high variance, resulting in high average error rates. However, for longer ciphers, the average symbol error rates are close to zero for the 50 ciphertexts. Thus, a larger sample reduces the variance.

5.3 Deciphering Homophonic Ciphers

We perform decipherment of two important homophonic ciphers. Both of these ciphers have been computationally deciphered before, providing us with a baseline to compare against. Table 5.2 details the characteristic attributes of the ciphers used in our experiments.

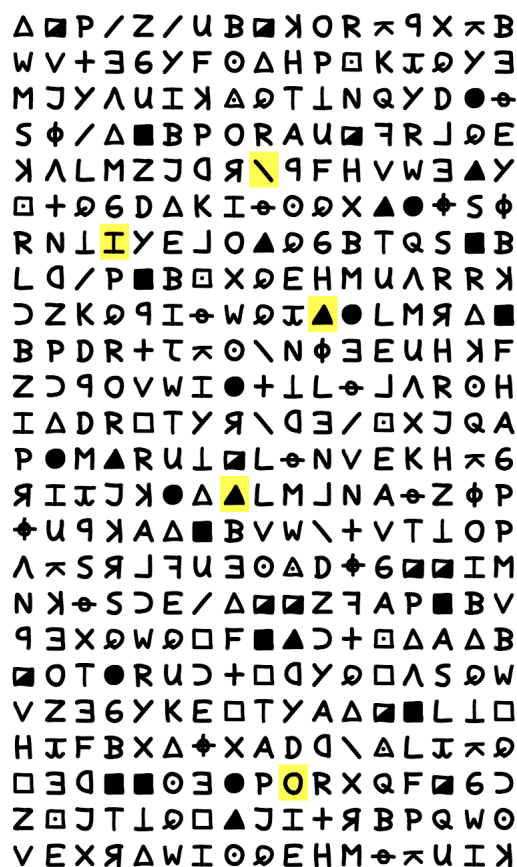
Cipher	Length	Unique Symbols	Obs/symbol
Zodiac-408	408	54	7.55
Beale Pt. 2	763	180	4.23

Table 5.2: Homophonic ciphers used in our experiments.

The quality of the results obtained on each of the decipherments is presented as two symbol error rates. SER1 represents the model with beam search using global scoring function while SER2 is the error obtained using beam search with global scoring function and frequency matching heuristic.

5.3.1 A Simple Cipher: Zodiac-408

Zodiac-408 is probably the most popular homophonic cipher seen in history. It was used by the notorious Zodiac serial killer 1960s in a series of mocking enciphered-letters sent to the newspapers. Though the identity of the killer remains a mystery, the encrypted messages attracted several cryptanalysts - professionals and amateurs alike. The first automatic decipherment of this cipher was reported by Ravi and Knight et al. [33]. Zodiac-408 cipher has since become a commonly used homophonic cipher to evaluate decipherment algorithms.



The image shows a 20x20 grid of 400 symbols representing the Zodiac-408 cipher. The symbols are a mix of letters, numbers, and various special characters like slashes, plus signs, and geometric shapes. Some symbols are highlighted in yellow, including 'I', 'T', 'A', 'N', and 'O'.

(a) Zodiac-408 cipher.



The image shows the deciphered version of the Zodiac-408 cipher text. It is a 20x20 grid of 400 characters. The text is:

I L I K E K I L L I N G P E O P L

E B E C A U S E I T I S S O M U C

H F U N I T I S M O R E F U N T H

A N K I L L I N G W I L D G A M E

I N T H E F O R T E S T B E C A U

S E M A N I S T H E M O S T D A N

G E R T U E A N A M A L O F A L L

T O K I L L S O M E T H I N G G I

V E S M E T H E M O A T T H R I L

L I N G E X P E R E N C E I T I S

E V E N B E T T E R T H A N G E T

T I N G Y O U R R O C K S O F F W

I T H A G I R L T H E B E S T P A

R T O F I T I A T H A E W H E N I

D I E I W I L L B E R E B O R N I

N P A R A D I C E A N D A L L T H

E I H A V E K I L L E D W I L L B

E C O M E M Y S L A V E S I W I L

L N O T G I V E Y O U M Y N A M E

B E C A U S E Y O U W I L L T R Y

T O S L O I D O W N O R A T O P M

Y C O L L E C T I N G O F S L A V

E S F O R M Y A F T E R L I F E E

B E O R I E T E M E T H H P I T I
 The words 'TUEANAMALOFALL', 'ATTHRILLING EXPERIENCE', 'PARADISE', and 'BEORIE TEMETHHPITI' are highlighted in yellow, cyan, and red respectively.

(b) Our decipherment of Zodiac-408 cipher.

Figure 5.2: Deciphering the Zodiac-408 cipher.

Getting its name from the length of the ciphertext, Zodiac-408 (Figure 5.5a) has 408 characters. It has 54 unique symbols and a per symbol observation of 7.5. Though these statistics present the Zodiac-408 cipher as easy to crack, there are several challenges that the

ciphertext poses. Except for 1 symbol in the cipher, every other symbol shows redundancy. This makes the cipher very non-deterministic. It also contains words like “PARADICE” and “FORREST” which are intentionally misspelled, and the last 18 characters in the cracked cipher don’t mean anything. All these attributes (color coded in Figure 5.5b) make the Zodiac-408 cipher a very good candidate to benchmark decipherment systems designed to break substitution ciphers.

Beam	SER (%) 1	SER (%) 2
10k	3.92	3.18
100k	2.40	1.91
1M	1.47	1.22

Table 5.3: Symbol Error Rates (SER) based on Neural Language Model and beam size (Beam) for deciphering Zodiac-408, respectively. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with the frequency matching heuristic.

Table 5.3 shows the performance of our system with different beam sizes. Our improved Beam Search algorithm is able to achieve an accuracy of 98.8% with a beam size of 1 million. Even with much lower beam size of 100k, we are able to get an accuracy of about 98%. Since beam search with higher beam sizes is computationally expensive, choosing a lower beam size at an expense of marginally lower accuracy turns out to be a good trade-off.

Our decipherment model with rest cost estimation and frequency matching heuristic with a beam size of 1M records an error of 1.2%. Figure (5.5b) shows the deciphered text we obtained with our system. Mis-mapped tokens have a bold font and are marked in yellow.

5.3.2 A Difficult Cipher: Beale Pt 2

The Beale Papers is a collection of three historical ciphertexts discovered in 1800s that are believed to lead to a hidden treasure. Though the first and the third part of the ciphertext remain unsolved, the second part has been deciphered. The first automatic decipherment of this cipher was present by Nuhn et al. [27]. Since this gives us a ground truth to compare our version of the decipherment against, we apply our system to the second part of the Beale Cipher, and report our results.

Part two of the Beale Cipher (Figure 5.4) is 763 characters long with 180 unique symbols amounting to 4.23 observations per symbol of the ciphertext. Only 43 of these symbols occur once while others are seen to be repetitive. The symbol **807** is the most frequent symbol, occurring 18 times. The high repetition of symbols, and the enormous search space of solutions makes Beale Pt 2 a challenging homophonic cipher to crack compared to the ciphers we’ve seen in the previous sections.

Beam	SER (%) 1	SER (%) 2
10k	41.67	48.33
100k	7.20	10.09
1M	4.98	5.50

Table 5.4: Symbol Error Rates (SER) based on Neural Language Model and beam size (Beam) for deciphering Part 2 of the Beale Cipher. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with the frequency matching heuristic.

With the deciphered text obtained using our model, illustrated in Figure 5.3, we find that Beale cipher is indeed a hard cipher to crack. The symbol errors in decipherment are marked in yellow.

I HAVE DEPOSITED IN THE COUNTY Y OF BEDFORD ABOUT FOUR MI
LES FROM BUFORDS IN ANE X CAVATION OR VAULTS IX FEET BEL
OW THE SURFACE OF THE GROUND THE FOLLOWING ARTICLES BE
LONGING JOINTLY Y TO THE PARTY Y ES WHOSE NAMES ARE GIVEN I
N NUMBER THREE BERE WITH THE FIRST DEPOSITE EONSISTE DO
FTEN HUNDRED AND FORTY X RTEEN POUNDS OF GOLD AND THIRTY Y EI
GH THUNDRED AND TWELVE K POUNDS OF SILVER DEPOSITED NON
EIGHTEEN NINETEEN THE SECOND O AS MADE I CEIGHTEEN TW
ENT Y FNEA J D CONSI Q TED OF NINETEEN HUNDRED AN A SEVEN P
OUNDS OF GOLD AND TWELVE HUNDRED AND EIGHT Y EIGHT OF SI
LVER ALSO FEWEL Q OBTAINED IN ST LOUIS IX CHANGETO SA
VET TRANSPOUTATION AND VALUE O AT THIRTEEN THOUSAND DO
LLARSTHEATOVE IN SECURELY Y PACKED IN IR Q NPOTS WITH IR
ON COVERSTHE VAULT IS ROUGHL Y LINED WITH STONE AND THE
VESSELS REST ON SOLID B TONE AND ARE COVERED WITH Q THER
SPAPER NUMBER ONE DESCRIBESTHE EXACT LOCALITY Y OF THE
VAULTS THAT NO DIFFICULT Y WILL BEHAD IN FINDING IT

Figure 5.3: Our decipherment of Beale cipher Pt-2.

Table 5.4 shows the performance of our decipherment system with different beam sizes on this task. Our improved Beam Search algorithm is able to achieve an accuracy of 95% with a beam size of 1 million. Much lower beam size of 100k yields an accuracy of about 93%.

115, 73, 24, 807, 37, 52, 49, 17, 31, 62, 647, 22, 7, 15, 140, 47, 29, 107, 79,
 84, 56, 239, 10, 26, 811, 5, 196, 308, 85, 52, 160, 136, 59, 211, 36, 9, 46, 316,
 554, 122, 106, 95, 53, 58, 2, 42, 7, 35, 122, 53, 31, 82, 77, 250, 196, 56, 96,
 118, 71, 140, 287, 28, 353, 37, 1005, 65, 147, 807, 24, 3, 8, 12, 47, 43, 59, 807,
 45, 316, 101, 41, 78, 154, 1005, 122, 138, 191, 16, 77, 49, 102, 57, 72, 34, 73,
 85, 35, 371, 59, 196, 81, 92, 191, 106, 273, 60, 394, 620, 270, 220, 106, 388,
 287, 63, 3, 6, 191, 122, 43, 234, 400, 106, 290, 314, 47, 48, 81, 96, 26, 115, 92,
 158, 191, 110, 77, 85, 197, 46, 10, 113, 140, 353, 48, 120, 106, 2, 607, 61, 420,
 811, 29, 125, 14, 20, 37, 105, 28, 248, 16, 159, 7, 35, 19, 301, 125, 110, 486,
 287, 98, 117, 511, 62, 51, 220, 37, 113, 140, 807, 138, 540, 8, 44, 287, 388, 117,
 18, 79, 344, 34, 20, 59, 511, 548, 107, 603, 220, 7, 66, 154, 41, 20, 50, 6, 575,
 122, 154, 248, 110, 61, 52, 33, 30, 5, 38, 8, 14, 84, 57, 540, 217, 115, 71, 29,
 84, 63, 43, 131, 29, 138, 47, 73, 239, 540, 52, 53, 79, 118, 51, 44, 63, 196, 12,
 239, 112, 3, 49, 79, 353, 105, 56, 371, 557, 211, 505, 125, 360, 133, 143, 101,
 15, 284, 540, 252, 14, 205, 140, 344, 26, 811, 138, 115, 48, 73, 34, 205, 316,
 607, 63, 220, 7, 52, 150, 44, 52, 16, 40, 37, 158, 807, 37, 121, 12, 95, 10, 15,
 35, 12, 131, 62, 115, 102, 807, 49, 53, 135, 138, 30, 31, 62, 67, 41, 85, 63, 10,
 106, 807, 138, 8, 113, 20, 32, 33, 37, 353, 287, 140, 47, 85, 50, 37, 49, 47, 64,
 6, 7, 71, 33, 4, 43, 47, 63, 1, 27, 600, 208, 230, 15, 191, 246, 85, 94, 511, 2,
 270, 20, 39, 7, 33, 44, 22, 40, 7, 10, 3, 811, 106, 44, 486, 230, 353, 211, 200,
 31, 10, 38, 140, 297, 61, 603, 320, 302, 666, 287, 2, 44, 33, 32, 511, 548, 10, 6,
 250, 557, 246, 53, 37, 52, 83, 47, 320, 38, 33, 807, 7, 44, 30, 31, 250, 10, 15,
 35, 106, 160, 113, 31, 102, 406, 230, 540, 320, 29, 66, 33, 101, 807, 138, 301,
 316, 353, 320, 220, 37, 52, 28, 540, 320, 33, 8, 48, 107, 50, 811, 7, 2, 113, 73,
 16, 125, 11, 110, 67, 102, 807, 33, 59, 81, 158, 38, 43, 581, 138, 19, 85, 400,
 38, 43, 77, 14, 27, 8, 47, 138, 63, 140, 44, 35, 22, 177, 106, 250, 314, 217, 2,
 10, 7, 1005, 4, 20, 25, 44, 48, 7, 26, 46, 110, 230, 807, 191, 34, 112, 147, 44,
 110, 121, 125, 96, 41, 51, 50, 140, 56, 47, 152, 540, 63, 807, 28, 42, 250, 138,
 582, 98, 643, 32, 107, 140, 112, 26, 85, 138, 540, 53, 20, 125, 371, 38, 36, 10,
 52, 118, 136, 102, 420, 150, 112, 71, 14, 20, 7, 24, 18, 12, 807, 37, 67, 110,
 62, 33, 21, 95, 220, 511, 102, 811, 30, 83, 84, 305, 620, 15, 2, 10, 8, 220, 106,
 353, 105, 106, 60, 275, 72, 8, 50, 205, 185, 112, 125, 540, 65, 106, 807, 138, 96,
 110, 16, 73, 33, 807, 150, 409, 400, 50, 154, 285, 96, 106, 316, 270, 205, 101,
 811, 400, 8, 44, 37, 52, 40, 241, 34, 205, 38, 16, 46, 47, 85, 24, 44, 15, 64, 73,
 138, 807, 85, 78, 110, 33, 420, 505, 53, 37, 38, 22, 31, 10, 110, 106, 101, 140,
 15, 38, 3, 5, 44, 7, 98, 287, 135, 150, 96, 33, 84, 125, 807, 191, 96, 511, 118,
 40, 370, 643, 466, 106, 41, 107, 603, 220, 275, 30, 150, 105, 49, 53, 287, 250,
 208, 134, 7, 53, 12, 47, 85, 63, 138, 110, 21, 112, 140, 485, 486, 505, 14, 73,
 84, 575, 1005, 150, 200, 16, 42, 5, 4, 25, 42, 8, 16, 811, 125, 160, 32, 205, 603,
 807, 81, 96, 405, 41, 600, 136, 14, 20, 28, 26, 353, 302, 246, 8, 131, 160, 140,
 84, 440, 42, 16, 811, 40, 67, 101, 102, 194, 138, 205, 51, 63, 241, 540, 122, 8,
 10, 63, 140, 47, 48, 140, 288

Figure 5.4: Beale cipher Pt-2.

5.3.3 Reference Plaintexts

The following Figure 5.5 shows the reference plaintexts for Zodiac-408 cipher, as used by Ravi and Knight [33], and Beale Cipher Pt-2 which were used to compare and evaluate our decipherments of these ciphers. The yellow marks in the reference plaintexts above show the letters which were wrongly deciphered with our system.

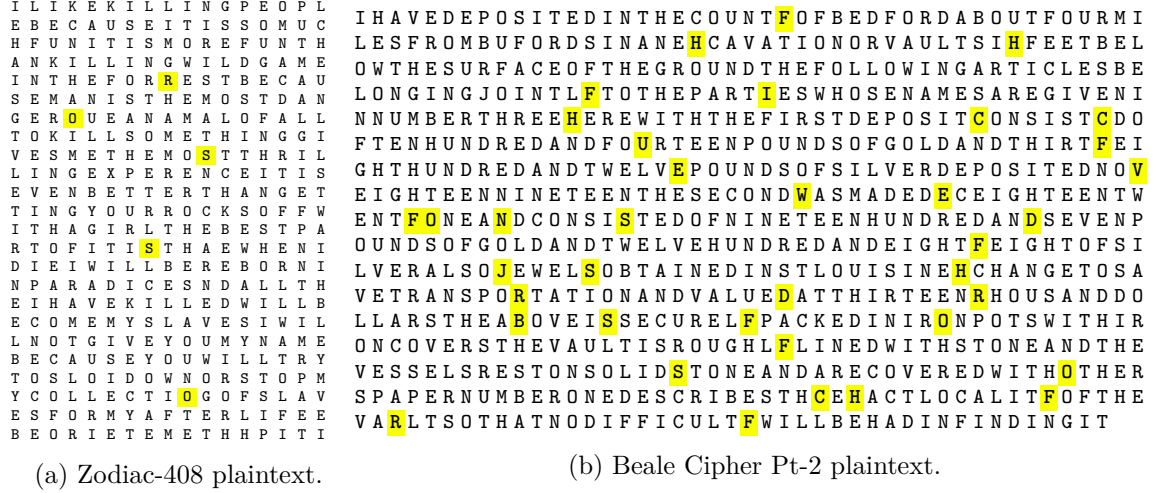


Figure 5.5: Reference plaintexts for Zodiac-408 and Beale Pt-2 ciphers used for evaluation.

5.3.4 Summary

We applied our novel decipherment system for the task of deciphering two historical homophonic ciphers : Zodiac-408 and Beale Pt 2, and observed that the method performs well and is comparable to the previous fully automatic statistical decipherment models. The following is a summary of the results.

Zodiac-408: Our best result on this cipher yields an SER of 1.2% compared to the beam search algorithm in [26] with beam size of 10M with a 6-gram LM that gives an SER of 2%. In [27], however, the authors present an extension to [26] and report a full decipherment of the cipher.

Beale Pt 2: On this cipher, we report a decipherment error of 5%. Our system outperforms the state of the art [27] (5.4% with 8-gram LM and a beam size of 10M) on this task while employing beam search with a beam size of 1M.

Discussion: We observe that for challenging ciphers such as Beale Pt 2, we obtain lower error rates with smaller beam sizes when compared to the state of the art [27] in decipherment for such ciphers. We note that the frequency matching heuristic may not be entirely useful in some scenarios. On simple letter substitution ciphers, we get almost identical

results using both variants of beam search, while on Zodiac-408, beam search with frequency matching heuristic performs better. However, on Beale Pt 2, beam search without the heuristic seems to be producing lower error rates.

Chapter 6

Conclusion

This thesis describes a novel neural LM based method for decipherment that effectively breaks a variety of substitution ciphers. This work, to our knowledge, is the first application of large pre-trained neural LMs to the decipherment problem. Unlike previous methods, our approach scores the candidate decipherments globally and employs a letter frequency matching heuristic, yielding a robust decipherment system. We modify the beam search algorithm for decipherment from [26, 27] to use global scoring of the plaintext message using neural LMs. To enable full plaintext scoring we use the neural LM to sample plaintext characters which improves convergence times. We use two versions of beam search for our experiments - a version with the improved rest cost estimator for evaluating partial hypotheses, and the other with the rest cost estimator augmented with the heuristic. This work, to our knowledge, is the first to use a neural language model in decipherment. We record the performance of our system on both versions of the beam search separately. We empirically evaluate our method through experiments on several substitution ciphers - synthetic 1:1 ciphers, Zodiac-408 and much harder Beale Pt 2 - and observe that the results have high accuracy.

Decipherment is an intriguing venue of research. Apart from the computational aspect of it, the idea itself has varied practical applications ranging from raw cryptography tasks like breaking cyber-codes to discovering ancient languages only found on clay tablets. This opens up a huge venue for computational assistance in the form of universal cipher-solvers in all of these applications, accelerating the whole ‘discovery’ process. Unfortunately, we don’t yet have perfect fully-automatic decipherment systems that are able to chew a cipher and spit out its plaintext. This work contributes towards a better, robust model for solving substitution ciphers. It may not yet be able to solve the mystery of the clay tablets of Mesopotamia, but it is powerful enough to decipher *Alienese*, the language of aliens.

Bibliography

- [1] L. Adkins and R. Adkins. *The Keys of Egypt: The Race to Crack the Hieroglyph Code*. HarperCollins, 2001.
- [2] Ibrahim A Al-Kadit. Origins of cryptology: The arab contributions. *Cryptologia*, 16(2):97–126, 1992.
- [3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.
- [4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [5] 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*, pages 874–878, 2013.
- [6] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.
- [7] Alex Biryukov and Eyal Kushilevitz. From differential cryptanalysis to ciphertext-only attacks. In *Annual International Cryptology Conference*, pages 72–88. Springer, 1998.
- [8] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1. In *Annual International Cryptology Conference*, pages 1–12. Springer, 1998.
- [9] Mikael Boden. A guide to recurrent neural networks and backpropagation. *the Dallas project*, 2002.
- [10] Jack Copeland. Alan turing: The codebreaker who saved 'millions of lives'. *BBC News Technology*, June 19, 2012.
- [11] 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. Association for Computational Linguistics, 2010.
- [12] Aidan N. Gomez, Sicong Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and ukasz Kaiser. Unsupervised cipher cracking using discrete gans. *arXiv preprint arXiv:1801.04883*, 2018.

- [13] Sam Greydanus. Learning the enigma with recurrent neural networks. *arXiv preprint arXiv:1708.07576*, 2017.
- [14] George W Hart. To decode short cryptograms. *Communications of the ACM*, 37(9):102–108, 1994.
- [15] 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, 2014.
- [16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [17] John Hutchins. First steps in mechanical translation. 1997.
- [18] Thomas Jakobsen. A fast method for cryptanalysis of substitution ciphers. *Cryptologia*, 19(3):265–274, 1995.
- [19] 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*, pages 499–506. Association for Computational Linguistics, 2006.
- [20] Kevin Knight and Kenji Yamada. A computational approach to deciphering unknown scripts. *Unsupervised Learning in Natural Language Processing*, 1999.
- [21] Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016.
- [22] W. N. Locke and D. A. Booth. Eds. *Machine Translation of Languages : fourteen essays*, pages 1–14, 1995.
- [23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [24] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.
- [25] Malte Nuhn, Arne Mauser, and Hermann Ney. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 156–164. Association for Computational Linguistics, 2012.
- [26] 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)*, volume 1, pages 1568–1576, 2013.
- [27] 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, 2014.

- [28] Edwin Olson. Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342, 2007.
- [29] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fifth edition (2011). *Linguistic Data Consortium, Philadelphia, PA, USA*, 2011.
- [30] Shmuel Peleg and Azriel Rosenfeld. Breaking substitution ciphers using a relaxation algorithm. *Communications of the ACM*, 22(11):598–605, 1979.
- [31] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.
- [32] 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*, pages 812–819. Association for Computational Linguistics, 2008.
- [33] 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*, pages 239–247. Association for Computational Linguistics, 2011.
- [34] Sujith Ravi and Kevin Knight. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 12–21. Association for Computational Linguistics, 2011.
- [35] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.
- [36] Warren Weaver. Translation. Reproduced in : W. N. Locke and D. A. Booth., eds. (1955). 1947.
- [37] Eric W Weisstein. Kronecker delta. 2002.
- [38] David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics, 2000.