CS181 Artificial Intelligence I: Final Project Report

Xinhang Liu, Yichen Zhu, Jiaben Chen, Jiale Yu, Xiao Yu
School of Information Science and Technology
ShanghaiTech University
Shanghai, China
{liuxh2,zhuych,chenjb1,yujl,yuxiao}@shanghaitech.edu.cn

Abstract

In this project, our group tries to perform decipherment on substitution cipher, using the method of Artificial Intelligence and the knowledge from this course. To be more specific, our method involves tools and techniques like Markov model(n-gram language model), beam search and CSP. Our pipeline works well on most ciphertexts with around 70 words and has an accpetable rate of error.

Keywords: Substitution cipher, Markov model, Beam search, CSP

1. Introduction

In cryptography, a substitution cipher is a method of encrypting in which units of plaintext are replaced with ciphertext, according to a fixed system. In this project, we focus on a substitution cipher with "units" being single letters. And we study the 1:1 substitution cipher which encrypts a given plaintext into a ciphertext by replacing each plaintext token with a unique substitute. We further give our problem formulation here: we formalize the 1:1 substition with a bijective function: $\phi : V_f \to V_e$, where $V_f, V_e \in \{a, b, ..., y, z\}$.

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 :

Plaintext character	A	B	C	D
Ciphertext character	C	A	D	B

In this key, the letters of English alphabet are plaintext tokens $e \in \mathbb{V}_e$ and beneath them are the corresponding ciphertext characters $e \in \mathbb{V}_f$ 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.Using this key, the word "BAD" can be ciphered into "ACB", which is shown in table1.

	A	B	C	D
A	0	0	1	0
B	1	0	0	0
C	0	0	0	1
D	0	1	0	0

Table 1. Substitution table for simple substitution cipher

Our 2-stage method for solving this kind of substitution cipher include beam-searching a system of substitution which achieve highest score from n-gram, our probablistic model, and then using a CSP formulation and a tool for the correction of

spelling to get the solution we want. Our pipeline works well on most ciphertexts with around 70 words and has an accpetable rate of error. Moreover, our methods is by no means expensive in the sense of time complexity.

A demo of our project can be seen at 1. As is demonstrated, the first red line is the original plaintext and the second blue line is the encrypted ciphertext. The third line is the decrypted plaintext from the ciphertext by our algorithm, so we can see it is identical to the original plaintext.

```
artificial intelligence is intelligence demonstrated by machines unlike the natural……
elkaqaxaen ajkonnasojxo au ajkonnasojxo rowhjuklekor pd wexvajou tjnaco kvo jektlen……
artificial intelligence is intelligence demonstrated by machines unlike the natural……
```

Figure 1. A demo of our project

The main contribution we make can be summarized as:

- We implement the n-gram model and give a pretrained model using data from the English version of *War and Peace*.
- We implement a beam search algorithm and make some clear specification of details in our formulation.
- We build a tool which can detect spelling errors in words, using a dataset of all words in *Wikipedia* and their frequency of appearance.
- We formulate this problem, by the first time, as a CSP model and use the combination of this CSP model and the tool for the correction of spelling error to handle the defects of beam search and n-gram.

2. Related Work

Various works have been published with regard to the decipherment of substitution cipher which uses a 1:1 cipher key. Knight et al.[1] used an HMM-based EM algorithm to solve decipherment problems. Ravi and Knight [2] extend this HMM-based EM algorithm with a Bayesian approach. Corlett and Penn[3] proposed an efficient A* search algorithm to solve substitution ciphers. Nowadays, the most widely used method for decipherment of mono-alphaetic substitution cipher is the use of beam search with n-gram language models(LMs) proposed by Nuhn et al.[4]. Most state-of-the-art approaches are n-gram LM-based and many works have been published in the natural language processing community. Nuhn et al.[5] present multifarious improvements to their previous beam search algorithm, including an improved rest cost estimation together with an optimized strategy for obtaining the order of symbols used in decipherment. Hauer et al.[6] propose a novel approach combining both character-level and word-levelo language models in order to decipher short monoalphabetic ciphers, which used Monte Carlo Tree Search (MCTS) as a fast alternative to beam search. Greydanus[7] frames the cipher decryption process as a sequence-to-sequence translation task, using a deep LTSM-based model. While this method needs supervision, Kambhatla et al.[8] proposes an approach using a pre-trained neural LM. They modify the beam search algorithm and extend it to use global scoring of plaintext message using large neural LMs.

In CS152 course at ShanghaiTech University, Prof. Liangfeng Zhang also introduces a method for cryptanalysis of the substitution cipher which utilizes the probability of English letters introduced by Berker et al.[9]. E occurs with the biggest probability of about 0.120, followed by T,A,O,I,N,S,H,R with probability between 0.06 and 0.09. To begin with, we obtain the frequency of occurence of the 26 ciphertext letters. We conjecture that the ciphertext letter with highest occuring frequency to be the ciphertext of plaintext ”e”.The remaining ciphertext characters that occur at least ten times are expected to be encryption of t,a,o,i,n,s,h,r in some order. However, the frequencies really do not vary enough to tell us what the correspondence might be. Hence, we also need to refer to digrams and trigrams. By looking at the 30 most common digrams and the 12 most common trigrams, we can conjecture the plaintext of adjacent letters of the most frequent ciphertext letter currently. If the deciphered-plaintext is not semantic logical, we trace back to previous level and perform another round of hypothesis. By repeating this process, we can finally filtrate a semantic logical substitution table for decipherment.

3. Main Methods

3.1. Model

The general decipherment goal is to to obtain a mapping $\phi \subseteq \mathbb{V}_f \times \mathbb{V}_e$ such that the probability of the deciphered text is maximal:

$$\hat{\phi} = \arg\max_{\phi} p(\phi)$$

Here we use n-gram Markov Model as the probability language model. In this model, the probability of a letter depends only on the previous n-1 letters. Thus, the general equation for this n-gram approximation to the conditional probability of the next letter in a sequence is

$$P(L_n|L_{1:n-1}) \approx P(L_n|L_{n-N+1:n-1})$$

Apply chain rule to letters,we get

$$\begin{aligned} P(L_{1:n}) &= P(L_1)P(L_2|L_1)P(L_3|L_1L_2)...P(L_n|L_{1:n-1}) \\ &= \prod_{k=1}^{n} P(L_k|L_{1:k-1}) \\ &= \prod_{k=1}^{N} P(L_k|L_{1:k-1}) \prod_{k=N+1}^{n} P(L_k|L_{k-N+1:k-1}) \end{aligned}$$

To learn the required probability, we use Maximum Likelihood Estimation. We get the MLE estimate for the parameters of an n-gram model by getting counts from a corpus, and normalizing the counts so that they lie between 0 and 1.

$$P(L_k|L_{k-N+1:k-1}) = \frac{C(L_{k-N+1:k-1}L_k)}{C(L_{k-N+1:k-1})}$$

Here C(...) means the number of input in the training set.

In practice, Since probabilities are (by definition) less than or equal to 1, the more probabilities we multiply together, the smaller the product becomes. Multiplying enough n-grams together would result in numerical underflow. By using log probabilities instead of raw probabilities, we get numbers that are not as small.

$$Score(L_{1:n}) = \sum_{k=1}^{N} \log(P(L_k|L_{1:k-1})) \sum_{k=N+1}^{n} \log(P(L_k|L_{k-N+1:k-1}))$$

So the general becomes:

$$\hat{\phi} = \arg\max_{\phi} Score(\phi(f))$$

3.2. Beam Search

3.2.1 General Algorithm

Algorithm 3.2.1 shows the general structure of the beam search based algorithm for the decipherment of substitution ciphers. The general idea is: For the first time step, choose [k] assignments with largest conditional probabilities for initial potential candidates for final output assignment. Then for each time step afterwards, choose the best [k]assignments based on the previous output sequence to be the potential output sequences for this time step. We always keep [k] potential candidates for output assignments. Finally, we choose the best one from the [k] output assignments. This beam search algorithm is a improvement of greedy search algorithm, which expand the searching space and still has linear time complexity. In this way, we achieve pruning by keeping only the [k] best hypotheses. Figure 2 demonstrates a sketch of our beam search algorithm.

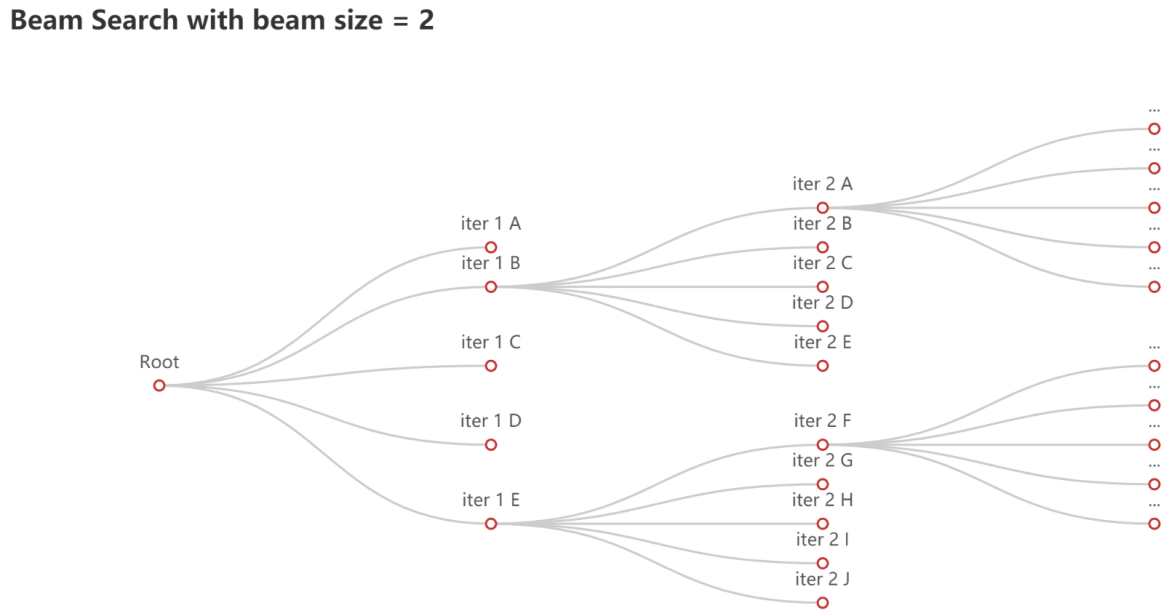

Figure 2. Sketch of beam search algorithm

Algorithm 1 Beam Search

Input: init_klist,maxitertion,ϵ
Output: best_klist
External: Score() , random() ,probability model;
kscore = Score(init_klist);
Sort init_klist by kscore;
best_klist = init_klist[:5];
for *i=1:maxitertion* **do**
    new_klist = best_klist;
    for *k in best_klist* **do**
        if *random()*$< \epsilon$ **then**
            Random swap two substitution in k;
        else
            Choose two substitution in k according to probability model;
        end
        new_klist.append(new_k);
    end
    n_kscore = Score(new_klist);
    Sort new_klist by n_kscore;
    best_klist = new_klist[:5];
end

3.2.2 Detail Implementation

1) Input:
The inputs are generated depending on the probability of single letter. For example, letter 'e' has probability much more

larger than any other letters in the training set. It's more likely that most frequent letter in cipher text is translated into 'e' than 'z'. By this method, we can guarantee a good initial state.

2) Beam Size:

In practice, we set beam size 5. In every iterations, we generate 3 new ciphers from one of the father ciphers. Together with the 5 father ciphers, we have 20 candidates. Keep the 5 best scored ciphers as the fathers for next iteration.

3) Generating Method:

To ensure the correctness of edge cases, we set a probability ϵ. With probability ϵ, we randomly swap the two substitution of two letters in father cipher to generate new cipher. With probability $1 - \epsilon$, we swap the two substitution of two letters with similar probability learned from training set.

3.2.3 Statistics Analysis

Figure.3 shows the performance and convergence of beam search. This experiment is carried out with n=2 and text with 75 words. As the process of iteration goes on, the score has a tendency of ascent. It should be noticed that the algorithm is non-optimal and it may converge to a local optima. In Figure.3, the line in red and the line in blue denote cases when the algorithm converges to the global optima and the line in green and the line in yellow denote cases when the algorithm converges to a local optima.

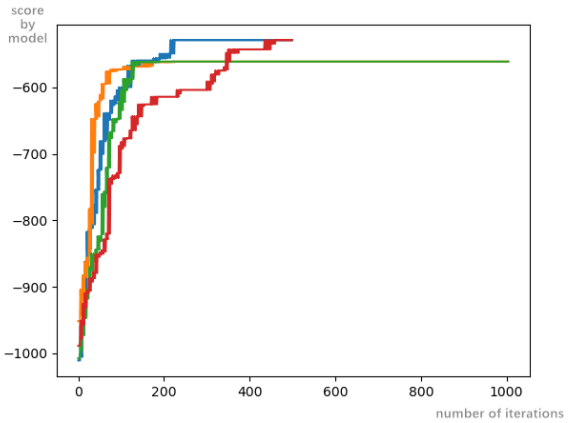

Figure 3. The convergence of beam search

3.3. CSP

There are some problems with the search algorithm introduced so far. Firstly, similar to Greedy Search, beam search dosen't guarantee an optimal solution. Secondly, applying the n-gram LM model solely is impossible to achieve one hundred percentage accuracy. In some of the experiments we conducted, the higest-score assignment given by n-gram LM model is not consistent with the correct assignment. Aiming at solving these problems, we use the Constraint Satisfaction Problems(CSP) knowledge learned in AI class this semester. In our decipherment problem, in order to assign a plaintext letter to each ciphertext letter, we are actually trying to find a group of correct assignments. What's more, we can assume that when there is no spelling error in the deciphered plaintext according to the assignment, then the assignment is correct. Hence, we can model this decipherment problem as a CSP. The constraint of this problem is the elimination of spelling errors. We make use of the words and words occurence frequencies provided by Wikipedia to implement a tool for spelling error detection and rectification. Back to our problem, we use the output assignment of beam search as the initial assignment of CSP. Afterwards, we try to reduce spelling errors in each step, which means we try to reduce the number of local changes violating the constraints. And we pick the way which can reduce the most spelling errors to change the assignment. When there are no spelling errors, we terminate the algorithm. The algorithm for this process can be seen in Algorithm 3.3.

After formulating this problem as a CSP and using speller to correct spelling, our pipeline can be obviously improved. This improvement is apparent especially for 2-gram. See Figure.4 and we can see cases of SER around 0.15 can be corrected perfectly by CSP formulation and speller. This experiment shows that the procedure can effiently deal with minor error and several wrong matching, caused by the inaccuracy of n-gram model and the non-optimality of beam search. For $n \geq 3$, the

improvement is less obvious. This can be explained by the fact that at the cost of larger time and space complexity, Markov model with $n \geq 3$ can achieve a comparatively good performance, sufficient to give a trustworthy plaintext.

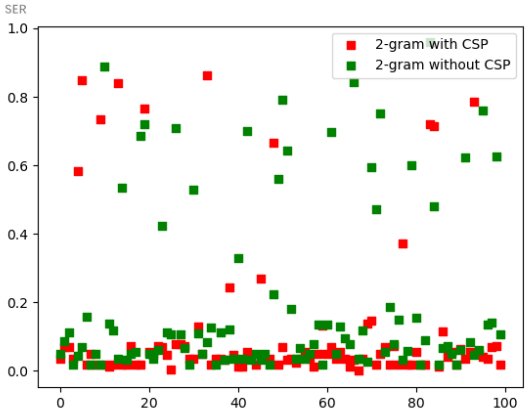

Figure 4. The improvement by CSP speller on 2-gram

Algorithm 2 Refinement using CSP

Input: assignment, words
Output: assignment
External: updateTable(), checkSpelling(), updateAssignment();
errorTable = zeros(26, 26);
while *True* **do**
  for *i in words* **do**
    updateTable(errorTable, checkSpelling(assignment(i)));
  end
  i, j = argmax(errorTable);
  if *errorTable(i,j)==0* **then**
    break
  end
  updateAssignment(assgnment, i, j);
end

4. Numerical Results

Figure.5 shows some experiments we carried out on models with different n and messages with different length. In these figures, a point denotes one experiment, and the y-axis denotes the SER($= \frac{\text{the number of wrong characters}}{\text{the number of all characters}}$). As we can see, in most cases, our pipeline provides an SER equal to 0 or near 0, meaning the plaintext we derive is readable and trustworthy. Moreover, the accuracy increases if we have a longer meassage and use model with larger n. In these figures, "iteration" in the label means one case of experiment.

We carry out experiments on 2-gram and 3-gram to show the relationship between the score by the model and the length of the text, measured in words. Considering this score also depend on the length of the text, we further carry out experiments to explore the relationship between SER($= \frac{\text{the number of wrong characters}}{\text{the number of all characters}}$) and the length of the text, measured in words. See Figure.6 and Figure.7 and we find out that as the length of the text increases, the accuracy of our pipeline gets better. For both of 2-gram and 3-gram, 70 words or above can guarantee an allowable accuracy. Intuitively, we can interpret this by imagining a ciphertext with only a few words or characters. For such a ciphertext, we may translate it into a number of reasonable plaintext, or it may be too hard for us to get the sense, so the difficulty lies in the cases with a comparatively small number of words or characters.

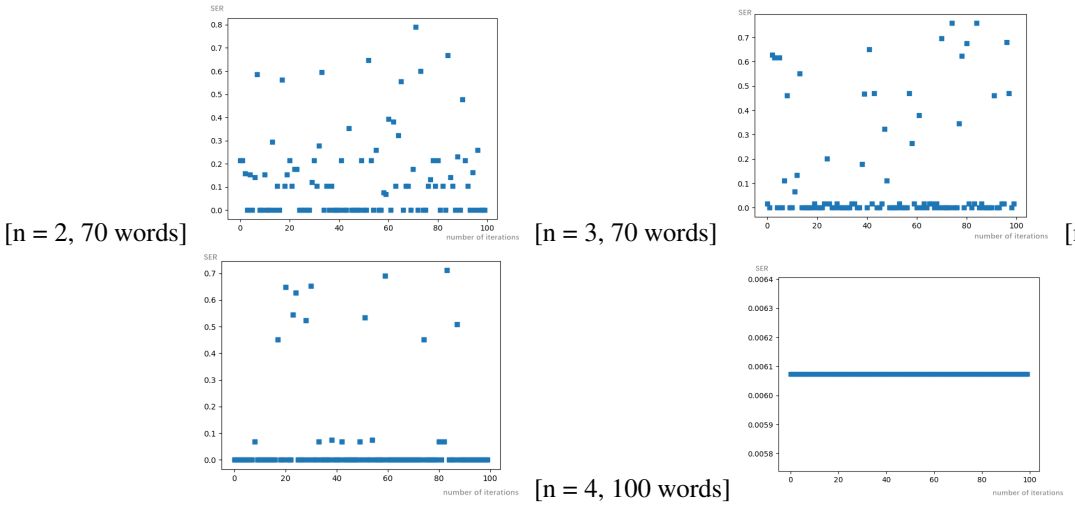

Figure 5. Numerical reults.

We further conduct a time cost experiment of n-gram LM. As is depicted in table 4, with higher gram model, the decipher accuracy became better, but the decipher time grows exponentially.

The experiments of Figure.6 uses the model of 2-gram. Table 3 compare performances by means of whether use CSP refinement and the value of n.

Order	Time
2-gram	5s
3-gram	20s-40s
4-gram	At least 6 min

Table 2. Decipher time using n-gram

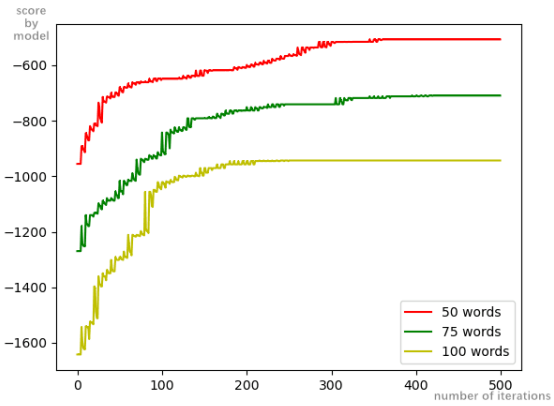

Figure 6. score by the model vs text length

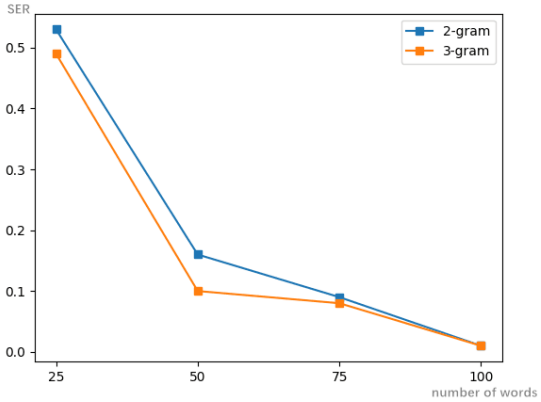

Figure 7. SER vs text length

Table 3. Performance

	50 words	75 words	100 words
2-gram without CSP refinement	0.16	0.11	0.02
2-gram with CSP refinement	0.14	0.09	0.02
3-gram with CSP refinement	0.10	0.08	0.02

5. Conclusion and Thoughts

- We have presented a simple beam search approach with n-gram Markov model to the substitution cipher problem. We've conducted a variety of experiments and the algorithm perform well with 3-gram LM with CSP refinement.
- For further development, we can solve some shortages discovered during experiments.For text less than 50 words, the algorithm has poor performance which is likely to converge into suboptimal solutions.To solve this problem, applying Expectimax algorithm instead of Beam search may be more effective.
- The CSP refinement we provided only works for separate word. When the input ciphertext is compound, we need to invent an effective tool to split the words with spelling error.

6. External resource

The training set used in the experiments is the book "War and Peace" from https://gutenberg.org/ebooks/2600.
The test set used in the experiments is from the book "Moby Dick" from https://gutenberg.org/ebooks/2701.
The data base is from https://github.com/IlyaSemenov/wikipedia-word-frequency.
[4] gives us some hints about applying n-gram and beam search.
In the program, no AI module such as Tensorflow or pytorch are imported in program. All codes are done by the group members independently.

References

[1] K. Knight, A. Nair, N. Rathod, and K. Yamada, "Unsupervised analysis for decipherment problems," in *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 2006, pp. 499–506.

[2] S. Ravi and K. 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*, 2011, pp. 239–247.

[3] E. Corlett and G. Penn, "An exact a* method for deciphering letter-substitution ciphers," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 1040–1047.

[4] M. Nuhn, J. Schamper, and H. Ney, "Beam search for solving substitution ciphers," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 1568–1576.

[5] ——, "Improved decipherment of homophonic ciphers," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1764–1768.

[6] B. Hauer, R. Hayward, and G. Kondrak, "Solving substitution ciphers with combined language models," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 2314–2325.

[7] S. Greydanus, "Learning the enigma with recurrent neural networks," *arXiv preprint arXiv:1708.07576*, 2017.

[8] N. Kambhatla, "Decipherment of substitution ciphers with neural language models," Ph.D. dissertation, Applied Sciences: School of Computing Science, 2018.

[9] B. Baker, M. Johns, and E. Fridjonsson, "Cipher systems," 1982.