Beam Search for Solving Substitution Ciphers

Malte Nuhn and **Julian Schamper** and **Hermann Ney**
Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University, Aachen, Germany
<surname>@cs.rwth-aachen.de

Abstract

In this paper we address the problem of solving substitution ciphers using a beam search approach. We present a conceptually consistent and easy to implement method that improves the current state of the art for decipherment of substitution ciphers and is able to use high order n-gram language models. We show experiments with 1:1 substitution ciphers in which the guaranteed optimal solution for 3-gram language models has 38.6% decipherment error, while our approach achieves 4.13% decipherment error in a fraction of time by using a 6-gram language model. We also apply our approach to the famous Zodiac-408 cipher and obtain slightly better (and near to optimal) results than previously published. Unlike the previous state-of-the-art approach that uses additional word lists to evaluate possible decipherments, our approach only uses a letter-based 6-gram language model. Furthermore we use our algorithm to solve large vocabulary substitution ciphers and improve the best published decipherment error rate based on the Gigaword corpus of 7.8% to 6.0% error rate.

1 Introduction

State-of-the-art statistical machine translation (SMT) systems use large amounts of parallel data to estimate translation models. However, parallel corpora are expensive and not available for every domain.

Recently different works have been published that train translation models using only non-parallel data. Although first practical applications of these approaches have been shown, the overall decipherment accuracy of the proposed algorithms is still low. Improving the core decipherment algorithms is an important step for making decipherment techniques useful for practical applications.

In this paper we present an effective beam search algorithm which provides high decipherment accuracies while having low computational requirements. The proposed approach allows using high order n-gram language models, is scalable to large vocabulary sizes and can be adjusted to account for a given amount of computational resources. We show significant improvements in decipherment accuracy in a variety of experiments while being computationally more effective than previous published works.

2 Related Work

The experiments proposed in this paper touch many of previously published works in the decipherment field.

Regarding the decipherment of 1:1 substitution ciphers various works have been published: Most older papers do not use a statistical approach and instead define some heuristic measures for scoring candidate decipherments. Approaches like (Hart, 1994) and (Olson, 2007) use a dictionary to check if a decipherment is useful. (Clark, 1998) defines other suitability measures based on n-gram counts and presents a variety of optimization techniques like simulated annealing, genetic algorithms and tabu search.

On the other hand, statistical approaches for 1:1 substitution ciphers were published in the natural language processing community: (Ravi and Knight, 2008) solve 1:1 substitution ciphers optimally by formulating the decipherment problem as an integer linear program (ILP) while (Corlett and Penn, 2010) solve the problem using A^* search. We use our own implementation of these methods to report optimal solutions to 1:1 substitution ci-

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 1568–1576,
Sofia, Bulgaria, August 4-9 2013.

phers for language model orders $n = 2$ and $n = 3$.

(Ravi and Knight, 2011a) report the first automatic decipherment of the Zodiac-408 cipher. They use a combination of a 3-gram language model and a word dictionary. We run our beam search approach on the same cipher and report better results without using an additional word dictionary—just by using a high order n-gram language model.

(Ravi and Knight, 2011b) report experiments on large vocabulary substitution ciphers based on the Transtac corpus. (Dou and Knight, 2012) improve upon these results and provide state-of-the-art results on a large vocabulary word substitution cipher based on the Gigaword corpus. We run our method on the same corpus and report improvements over the state of the art.

(Ravi and Knight, 2011b) and (Nuhn et al., 2012) have shown that—even for larger vocabulary sizes—it is possible to learn a full translation model from non-parallel data. Even though this work is currently only able to deal with substitution ciphers, phenomena like reordering, insertions and deletions can in principle be included in our approach.

3 Definitions

In the following we will use the machine translation notation and denote the **ciphertext** with $f_1^N = f_1 \dots f_j \dots f_N$ which consists of cipher tokens $f_j \in V_f$. We denote the **plaintext** with $e_1^N = e_1 \dots e_i \dots e_N$ (and its vocabulary V_e respectively). We define

$$e_0 = f_0 = e_{N+1} = f_{N+1} = \$ \tag{1}$$

with "$" being a special sentence boundary token. We use the abbreviations $\overline{V}_e = V_e \cup \{\$\}$ and $\overline{V}_f$ respectively.

A **general substitution cipher** uses a table $s(e|f)$ which contains for each cipher token f a probability that the token f is substituted with the plaintext token e. Such a table for substituting cipher tokens $\{A, B, C, D\}$ with plaintext tokens $\{a, b, c, d\}$ could for example look like

	a	b	c	d
A	0.1	0.2	0.3	0.4
B	0.4	0.2	0.1	0.3
C	0.4	0.1	0.2	0.3
D	0.3	0.4	0.2	0.1

The **1:1 substitution cipher** encrypts a given plaintext into a ciphertext by replacing each plaintext token with a unique substitute: This means that the table $s(e|f)$ contains all zeroes, except for one "1.0" per $f \in V_f$ and one "1.0" per $e \in V_e$. For example the text

abadcab

would be enciphered to

BCBADBC

when using the substitution

	a	b	c	d
A	0	0	0	1
B	1	0	0	0
C	0	1	0	0
D	0	0	1	0

In contrast to the 1:1 substitution cipher, the **homophonic substitution cipher** allows multiple cipher tokens per plaintext token, which means that the table $s(e|f)$ is all zero, except for one "1.0" per $f \in V_f$. For example the above plaintext could be enciphered to

ABCDECF

when using the homophonic substitution

	a	b	c	d
A	1	0	0	0
B	0	1	0	0
C	1	0	0	0
D	0	0	0	1
E	0	0	1	0
F	0	1	0	0

We will use the definition

$$n_{max} = \max_e \sum_f s(e|f) \tag{2}$$

to characterize the maximum number of different cipher symbols allowed per plaintext symbol.

We formalize the 1:1 substitutions with a *bijective* function $\phi : V_f \to V_e$ and homophonic substitutions with a *general* function $\phi : V_f \to V_e$.

Following (Corlett and Penn, 2010), we call cipher functions ϕ, for which not all $\phi(f)$'s are fixed, **partial cipher functions** . Further, ϕ' **is said to extend** ϕ, if for all f that are fixed in ϕ, it holds that f is also fixed in ϕ' with $\phi'(f) = \phi(f)$.

The **cardinality of** ϕ counts the number of fixed f's in ϕ.

When talking about partial cipher functions we use the **notation for relations**, in which $\phi \subseteq V_f \times V_e$. For example with

$$\phi = \{(A, a)\} \qquad \phi' = \{(A, a), (B, b)\}$$

it follows that $\phi \subseteq^1 \phi'$ and

$$\begin{aligned} |\phi| &= 1 & |\phi'| &= 2 \\ \phi(A) &= a & \phi'(A) &= a \\ \phi(B) &= \textit{undefined} & \phi'(B) &= b \end{aligned}$$

The general **decipherment goal** is to obtain a mapping ϕ such that the probability of the deciphered text is maximal:

$$\hat{\phi} = \arg\max_{\phi} p(\phi(f_1)\phi(f_2)\phi(f_3)...\phi(f_N)) \quad (3)$$

Here $p(\dots)$ denotes the **language model**. Depending on the structure of the language model Equation 3 can be further simplified.

4 Beam Search

In this Section we present our beam search approach to solving Equation 3. We first present the general algorithm, containing many higher level functions. We then discuss possible instances of these higher level functions.

4.1 General Algorithm

Figure 1 shows the general structure of the beam search algorithm for the decipherment of substitution ciphers. The general idea is to keep track of all partial hypotheses in two arrays H_s and H_t. During search all possible extensions of the partial hypotheses in H_s are generated and scored. Here, the function EXT_ORDER chooses which cipher symbol is used next for extension, EXT_LIMITS decides which extensions are allowed, and SCORE scores the new partial hypotheses. PRUNE then selects a subset of these hypotheses which are stored to H_t. Afterwards the array H_s is copied to H_t and the search process continues with the updated array H_s.

Due to the structure of the algorithm the cardinality of all hypotheses in H_s increases in each step. Thus only hypotheses of the same cardinality are compared in the pruning step. When H_s contains full cipher relations, the cipher relation with the maximal score is returned.[2]

Figure 2 illustrates how the algorithm explores the search space for a homophonic substitution cipher. In the following we show several instances of EXT_ORDER, EXT_LIMITS, SCORE, and PRUNE.

```
1:  function BEAM_SEARCH(EXT_ORDER, EXT_LIMITS, PRUNE)
2:      init sets H_s, H_t
3:      CARDINALITY = 0
4:      H_s.ADD((∅, 0))
5:      while CARDINALITY < |V_f| do
6:          f = EXT_ORDER[CARDINALITY]
7:          for all φ ∈ H_s do
8:              for all e ∈ V_e do
9:                  φ' := φ ∪ {(e, f)}
10:                 if EXT_LIMITS(φ') then
11:                     H_t.ADD(φ', SCORE(φ'))
12:                 end if
13:             end for
14:         end for
15:         PRUNE(H_t)
16:         CARDINALITY = CARDINALITY + 1
17:         H_s = H_t
18:         H_t.CLEAR()
19:     end while
20:     return best scoring cipher function in H_s
21: end function
```

Figure 1: The general structure of the beam search algorithm for decipherment of substitution ciphers. The high level functions SCORE, EXT_ORDER, EXT_LIMITS and PRUNE are described in Section 4.

4.2 Extension Limits (EXT_LIMITS)

In addition to the implicit constraint of ϕ being a function $V_f \rightarrow V_e$, one might be interested in functions of a specific form:

For 1:1 substitution ciphers (EXT_LIMITS_SIMPLE) ϕ must fulfill that the number of cipher letters $f \in V_f$ that map to any $e \in V_e$ is at most one. Since partial hypotheses violating this condition can never "recover" when being extended, it becomes clear that these partial hypotheses can be left out from search.

[1] shorthand notation for ϕ' extends ϕ

[2] n-best output can be implemented by returning the n best scoring hypotheses in the final array H_s.

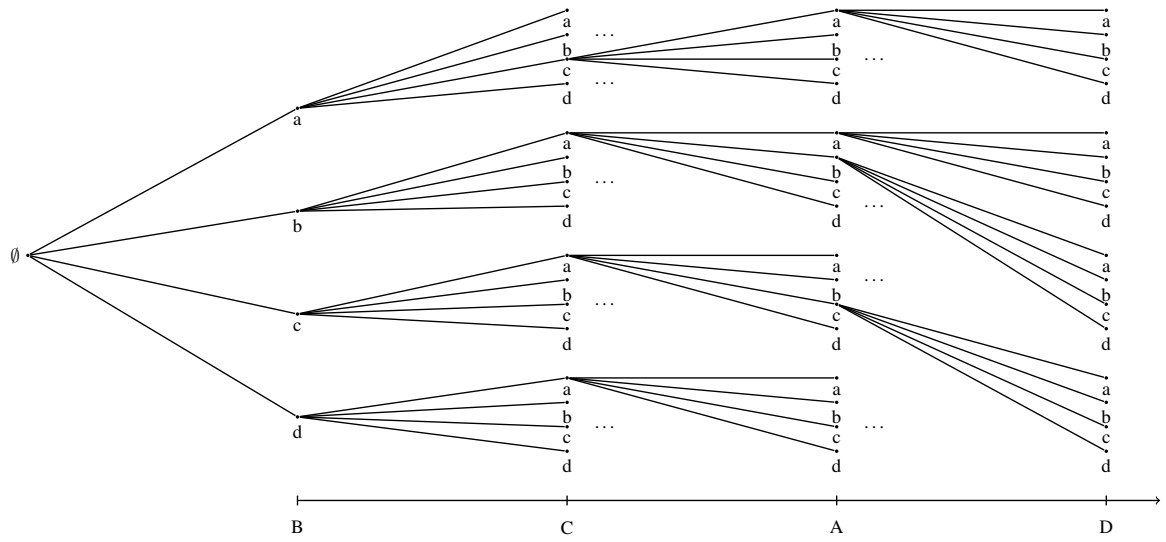

Figure 2: Illustration of the search space explored by the beam search algorithm with cipher vocabulary $V_f = \{A, B, C, D\}$, plaintext vocabulary $V_e = \{a, b, c, d\}$, EXT_ORDER = (B, C, A, D), homophonic extension limits (EXT_LIMITS_HOMOPHONIC) with $n_{max} = 4$, and histogram pruning with $n_{keep} = 4$. Hypotheses are visualized as nodes in the tree. The x-axis represents the extension order. At each level only those 4 hypotheses that survived the histogram pruning process are extended.

Homophonic substitution ciphers can be handled by the beam search algorithm, too. Here the condition that ϕ must fulfill is that the number of cipher letters $f \in V_f$ that map to any $e \in V_e$ is at most n_{max} (which we will call EXT_LIMITS_HOMOPHONIC). As soon as this condition is violated, all further extensions will also violate the condition. Thus, these partial hypotheses can be left out.

4.3 Score Estimation (SCORE)

The score estimation function needs to predict how good or bad a *partial* hypothesis (cipher function) might become. We propose simple heuristics that use the n-gram counts rather than the original ciphertext. The following formulas consider the 2-gram case. Equations for higher n-gram orders can be obtained analogously.

With Equation 3 in mind, we want to estimate the best possible score

$$\prod_{j=1}^{N+1} p(\phi'(f_j)|\phi'(f_{j-1})) \tag{4}$$

which can be obtained by extensions $\phi' \supseteq \phi$. By defining counts[3]

$$N_{ff'} = \sum_{i=1}^{N+1} \delta(f, f_{i-1})\delta(f', f_i) \tag{5}$$

we can equivalently use the scores

$$\sum_{f,f' \in \overline{V}_f} N_{ff'} \log p(\phi'(f')|\phi'(f)) \tag{6}$$

Using this formulation it is easy to propose a whole class of heuristics: We only present the simplest heuristic, which we call TRIVIAL_HEURISTIC. Its name stems from the fact that it only evaluates those parts of a given ϕ' that are already fixed, and thus does not estimate any future costs. Its score is calculated as

$$\sum_{f,f' \in \phi'} N_{ff'} \log p(\phi'(f')|\phi'(f)). \tag{7}$$

Here $f, f' \in \phi'$ denotes that f and f' need to be covered in ϕ'. This heuristic is *optimistic* since we implicitly use "0" as estimate for the non fixed parts of the sum, for which $N_{ff'} \log p(\cdot|\cdot) \leq 0$ holds.

It should be noted that this heuristic can be implemented very efficiently. Given a partial hypothesis ϕ with given SCORE(ϕ) the score of an extension ϕ' can be calculated as

$$\text{SCORE}(\phi') = \text{SCORE}(\phi) + \text{NEWLY\_FIXED}(\phi, \phi') \tag{8}$$

where NEWLY_FIXED only includes scores for n-grams that have been newly fixed in ϕ' during the extension step from ϕ to ϕ'.

[3] δ denotes the Kronecker delta.

4.4 Extension Order (EXT_ORDER)

For the choice which ciphertext symbol should be fixed next during search, several possibilities exist: The overall goal is to choose an extension order that leads to an overall low error rate. Intuitively it seems a good idea to first try to decipher higher frequent words rather than the lowest frequent ones. It is also clear that the choice of a good extension order is dependent on the score estimation function SCORE: The extension order should lead to informative scores early on so that misleading hypotheses can be pruned out early.

In most of our experiments we will make use of a very simple extension order: HIGHEST_UNIGRAM_FREQUENCY simply fixes the most frequent symbols first.

In case of the Zodiac-408, we use another strategy that we call HIGHEST_NGRAM_COUNT extension order. In each step it greedily chooses the symbol that will maximize the number of fixed ciphertext n-grams. This strategy is useful because the SCORE function we use is TRIVIAL_HEURISTIC, which is not able to provide informative scores if only few full n-grams are fixed.

4.5 Pruning (PRUNE)

We propose two pruning methods: HISTOGRAM_PRUNING sorts all hypotheses according to their score and then keeps only the best n_{keep} hypotheses.

THRESHOLD_PRUNING keeps only those hypotheses ϕ_{keep} for which

$$\text{SCORE}(\phi_{keep}) \geq \text{SCORE}(\phi_{best}) - \beta \qquad (9)$$

holds for a given parameter $\beta \geq 0$. Even though THRESHOLD_PRUNING has the advantage of not needing to sort all hypotheses, it has proven difficult to choose proper values for β. Due to this, all experiments presented in this paper only use HISTOGRAM_PRUNING.

5 Iterative Beam Search

(Ravi and Knight, 2011b) propose a so called "iterative EM algorithm". The basic idea is to run a decipherment algorithm—in their case an EM algorithm based approach—on a subset of the vocabulary. After having obtained the results from the restricted vocabulary run, these results are used to initialize a decipherment run with a larger vocabulary. The results from this run will then be used for a further decipherment run with an even larger vocabulary and so on. In our large vocabulary word substitution cipher experiments we iteratively increase the vocabulary from the 1000 most frequent words, until we finally reach the 50000 most frequent words.

6 Experimental Evaluation

We conduct experiments on letter based 1:1 substitution ciphers, the homophonic substitution cipher Zodiac-408, and word based 1:1 substitution ciphers.

For a given reference mapping ϕ_{ref}, we evaluate candidate mappings ϕ using two error measures: **Mapping Error Rate** $\text{MER}(\phi, \phi_{ref})$ and **Symbol Error Rate** $\text{SER}(\phi, \phi_{ref})$. Roughly speaking, SER reports the fraction of symbols in the deciphered text that are not correct, while MER reports the fraction of incorrect mappings in ϕ.

Given a set of symbols V_{eval} with unigram counts $N(v)$ for $v \in V_{eval}$, and the total amount of running symbols $N_{eval} = \sum_{v \in V_{eval}} N(v)$ we define

$$\text{MER} = 1 - \sum_{v \in V_{eval}} \frac{1}{|V_{eval}|} \cdot \delta(\phi(v), \phi_{ref}(v)) \qquad (10)$$

$$\text{SER} = 1 - \sum_{v \in V_{eval}} \frac{N(v)}{N_{eval}} \cdot \delta(\phi(v), \phi_{ref}(v)) \qquad (11)$$

Thus the SER can be seen as a weighted form of the MER, emphasizing errors for frequent words. In decipherment experiments, SER will often be lower than MER, since it is often easier to decipher frequent words.

6.1 Letter Substitution Ciphers

As **ciphertext** we use the text of the English Wikipedia article about *History*[4], remove all pictures, tables, and captions, convert all letters to lowercase, and then remove all non-letter and non-space symbols. This corpus forms the basis for shorter cryptograms of size $2, 4, 8, 16, 32, 64, 128,$ and 256—of which we generate 50 each. We make sure that these shorter cryptograms do not end or start in the middle of a word. We create the ciphertext using a **1:1 substitution cipher** in which we fix the mapping of the space symbol '_'. This

[4] http://en.wikipedia.org/wiki/History

Order	Beam	MER [%]	SER [%]	RT [s]
3	10	33.15	25.27	0.01
	100	12.00	6.95	0.06
	1k	7.37	3.06	0.53
	10k	5.10	1.42	5.33
	100k	4.93	1.31	47.70
	∞^*	4.93	1.31	19 700.00
4	10	55.97	48.19	0.02
	100	18.15	14.41	0.10
	1k	5.13	3.42	0.89
	10k	1.55	1.00	8.57
	100k	0.39	0.06	81.34
5	10	69.19	60.13	0.02
	100	35.57	29.02	0.14
	1k	10.89	8.47	1.29
	10k	0.38	0.06	11.91
	100k	0.38	0.06	120.38
6	10	74.65	64.77	0.03
	100	40.26	33.38	0.17
	1k	13.53	10.08	1.58
	10k	2.45	1.28	15.77
	100k	0.09	0.05	151.85

Table 1: Symbol error rates (SER), Mapping error rates (MER) and runtimes (RT) in dependence of language model order (ORDER) and histogram pruning size (BEAM) for decipherment of letter substitution ciphers of length 128. Runtimes are reported on a single core machine. Results for beam size "∞" were obtained using A^* search.

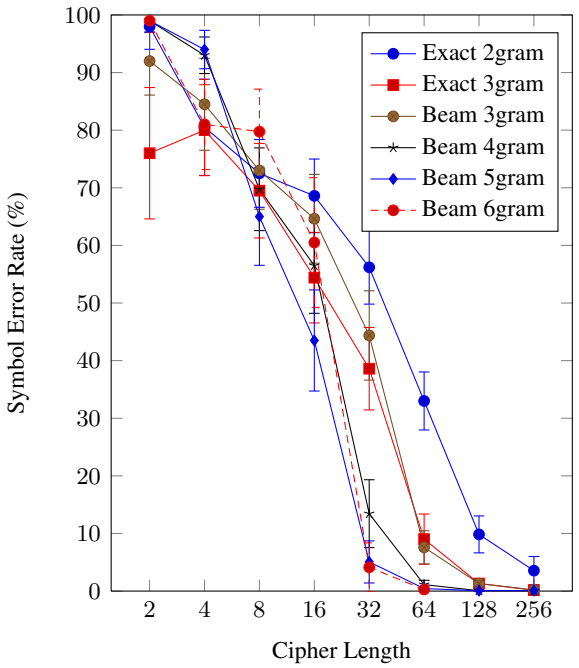

Figure 3: Symbol error rates for decipherment of letter substitution ciphers of different lengths. Error bars show the 95% confidence interval based on decipherment on 50 different ciphers. Beam search was performed with a beam size of "100k".

makes our experiments comparable to those conducted in (Ravi and Knight, 2008). Note that fixing the '_' symbol makes the problem much easier: The exact methods show much higher computational demands for lengths beyond 256 letters when not fixing the space symbol.

The **plaintext language model** we use a letter based ($V_e = \{a, \ldots, z, \_\}$) language model trained on a subset of the Gigaword corpus (Graff et al., 2007).

We use **extension limits** fitting the 1:1 substitution cipher $n_{max} = 1$ and **histogram pruning** with different beam sizes.

For comparison we reimplemented the ILP approach from (Ravi and Knight, 2008) as well as the A^* approach from (Corlett and Penn, 2010).

Figure 3 shows the **results** of our algorithm for different cipher length. We use a beam size of $100k$ for the 4, 5 and 6-gram case. Most remarkably our 6-gram beam search results are significantly better than all methods presented in the literature. For the cipher length of 32 we obtain a symbol error rate of just 4.1% where the optimal solution (i.e. without search errors) for a 3-gram language model has a symbol error rate as high as 38.3%.

Table 1 shows error rates and runtimes of our algorithm for different beam sizes and language model orders given a fixed ciphertext length of 128 letters. It can be seen that achieving close to optimal results is possible in a fraction of the CPU time needed for the optimal solution: In the 3-gram case the optimal solution is found in $\frac{1}{400}$th of the time needed using A^* search. It can also be seen that increasing the language model order does not increase the runtime much while providing better results if the beam size is large enough: If the beam size is not large enough, the decipherment accuracy decreases when increasing the language model order: This is because the higher order heuristics do not give reliable scores if only few n-grams are fixed.

To summarize: The beam search method is significantly faster and obtains significantly better results than previously published methods. Furthermore it offers a good trade-off between CPU time and decipherment accuracy.

Figure 4: First 136 letters of the Zodiac-408 cipher and its decipherment.

Order	Beam	MER [%]	SER [%]	RT [s]
4	10k	71.43	67.16	222
	100k	66.07	61.52	1 460
	1M	39.29	34.80	12 701
	10M	19.64	16.18	125 056
5	10k	94.64	96.57	257
	100k	10.71	5.39	1 706
	1M	8.93	3.19	14 724
	10M	8.93	3.19	152 764
6	10k	87.50	84.80	262
	100k	94.64	94.61	1 992
	1M	8.93	2.70	17 701
	10M	**7.14**	**1.96**	167 181

Table 2: Symbol error rates (SER), Mapping error rates (MER) and runtimes (RT) in dependence of language model order (ORDER) and histogram pruning size (BEAM) for the decipherment of the Zodiac-408 cipher. Runtimes are reported on a 128-core machine.

6.2 Zodiac-408 Cipher

As **ciphertext** we use a transcription of the Zodiac-408 cipher. It consists of 54 different symbols and has a length of 408 symbols.[5] The cipher has been deciphered by hand before. It contains some mistakes and ambiguities: For example, it contains misspelled words like *forrest* (vs. *forest*), *experence* (vs. *experience*), or *paradice* (vs. *paradise*). Furthermore, the last 17 letters of the cipher do not form understandable English when applying the same homophonic substitution that deciphers the rest of the cipher. This makes the Zodiac-408 a good candidate for testing the robustness of a decipherment algorithm.

We assume a **homophonic substitution cipher**, even though the cipher is not strictly homophonic: It contains three cipher symbols that correspond to two or more plaintext symbols. We ignore this fact for our experiments, and count—in case of the MER *only*—the decipherment for these symbols as correct when the obtained mapping is contained in the set of reference symbols. We use **extension limits** with $n_{max} = 8$ and **histogram pruning** with beam sizes of $10k$ up to $10M$.

The **plaintext language model** is based on the same subset of Gigaword (Graff et al., 2007) data as the experiments for the letter substitution ciphers. However, we first removed all space symbols from the training corpus before training the actual letter based 4-gram, 5-gram, and 6-gram language model on it. Other than (Ravi and Knight, 2011a) we do not use any word lists and by that avoid any degrees of freedom in how to integrate it into the search process: Only an n-gram language model is used.

Figure 4 shows the first parts of the cipher and our best decipherment. Table 2 shows the **results** of our algorithm on the Zodiac-408 cipher for different language model orders and pruning settings.

To summarize: Our final decipherment—for which we only use a 6-gram language model—has a symbol error rate of only 2.0%, which is slightly better than the best decipherment reported in (Ravi and Knight, 2011a). They used an n-gram language model together with a word dictionary and obtained a symbol error rate of 2.2%. We thus obtain better results with less modeling.

6.3 Word Substitution Ciphers

As **ciphertext**, we use parts of the JRC corpus (Steinberger et al., 2006) and the Gigaword corpus (Graff et al., 2007). While the full JRC corpus contains roughly 180k word types and consists of approximately 70M running words, the full Gigaword corpus contains around 2M word types and roughly 1.5G running words.

We run experiments for three different setups: The "JRC" and "Gigaword" setups use the first half of the respective corpus as ciphertext, while the **plaintext language model** of order $n = 3$ was

[5] hence its name

Setup	Top	MER [%]	SER [%]	RT [hh:mm]
Gigaword	1k	81.91	27.38	03h 10m
	10k	30.29	8.55	09h 21m
	20k	21.78	6.51	16h 25m
	50k	19.40	5.96	49h 02m
JRC	1k	73.28	15.42	00h 32m
	10k	15.82	2.61	13h 03m
JRC-Shuf	1k	76.83	19.04	00h 31m
	10k	15.08	2.58	13h 03m

Table 3: Word error rates (WER), Mapping error rates (MER) and runtimes (RT) for iterative decipherment run on the (TOP) most frequent words. Error rates are evaluated on the full vocabulary. Runtimes are reported on a 128-core machine.

trained on the second half. The "JRC-Shuf" setup is created by randomly selecting half of the sentences of the JRC corpus as ciphertext, while the language model was trained on the complementary half of the corpus.

We encrypt the ciphertext using a **1:1 substitution cipher** on word level, imposing a much larger vocabulary size. We use **histogram pruning** with a beam size of 128 and use **extension limits** of $n_{max} = 1$. Different to the previous experiments, we use **iterative beam search** with iterations as shown in Table 3.

The results for the Gigaword task are directly comparable to the word substitution experiments presented in (Dou and Knight, 2012). Their final decipherment has a symbol error rate of 7.8%. Our algorithm obtains 6.0% symbol error rate. It should be noted that the improvements of 1.8% symbol error rate correspond to a larger improvement in terms of mapping error rate. This can also be seen when looking at Table 3: An improvement of the symbol error rate from 6.51% to 5.96% corresponds to an improvement of mapping error rate from 21.78% to 19.40%.

To summarize: Using our beam search algorithm in an iterative fashion, we are able to improve the state-of-the-art decipherment accuracy for word substitution ciphers.

7 Conclusion

We have presented a simple and effective beam search approach to the decipherment problem. We have shown in a variety of experiments—letter substitution ciphers, the Zodiac-408, and word substitution ciphers—that our approach outperforms the current state of the art while being conceptually simpler and keeping computational demands low.

We want to note that the presented algorithm is not restricted to 1:1 and homophonic substitution ciphers: It is possible to extend the algorithm to solve n:m mappings. Along with more sophisticated pruning strategies, score estimation functions, and extension orders, this will be left for future research.

Acknowledgements

This work was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation. Experiments were performed with computing resources granted by JARA-HPC from RWTH Aachen University under project "jara0040".

References

Andrew J. Clark. 1998. *Optimisation heuristics for cryptology*. Ph.D. thesis, Faculty of Information Technology, Queensland University of Technology.

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

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 266–275, Jeju Island, Korea, July. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2007. English Gigaword Third Edition. Linguistic Data Consortium, Philadelphia.

George W. Hart. 1994. To decode short cryptograms. *Communications of the Association for Computing Machinery (CACM)*, 37(9):102–108, September.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 156–164, Jeju, Republic of Korea, July. Association for Computational Linguistics.

Edwin Olson. 2007. Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342, October.

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

Sujith Ravi and Kevin Knight. 2011a. Bayesian inference for Zodiac and other homophonic ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 239–247, Portland, Oregon, June. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2011b. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 12–21, Portland, Oregon, USA, June. Association for Computational Linguistics.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, and Dan Tufiş. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2142–2147. European Language Resources Association.