Large Scale Decipherment for Out-of-Domain Machine Translation

Qing Dou and Kevin Knight
Information Sciences Institute
Department of Computer Science
University of Southern California
{qdou,knight}@isi.edu

Abstract

We apply slice sampling to Bayesian decipherment and use our new decipherment framework to improve out-of-domain machine translation. Compared with the state of the art algorithm, our approach is highly scalable and produces better results, which allows us to decipher ciphertext with billions of tokens and hundreds of thousands of word types with high accuracy. We decipher a large amount of monolingual data to improve out-of-domain translation and achieve significant gains of up to 3.8 BLEU points.

1 Introduction

Nowadays, state of the art statistical machine translation (SMT) systems are built using large amounts of bilingual parallel corpora. Those corpora are used to estimate probabilities of word-to-word translation, word sequences rearrangement, and even syntactic transformation. Unfortunately, as parallel corpora are expensive and not available for every domain, performance of SMT systems drops significantly when translating out-of-domain texts (Callison-Burch et al., 2008).

In general, it is easier to obtain in-domain monolingual corpora. Is it possible to use domain specific monolingual data to improve an MT system trained on parallel texts from a different domain? Some researchers have attempted to do this by adding a domain specific dictionary (Wu et al., 2008), or mining unseen words (Daumé and Jagarlamudi, 2011) using one of several translation lexicon induction techniques (Haghighi et al., 2008; Koehn and Knight, 2002; Rapp, 1995). However, a dictionary is not always available, and it is difficult to assign probabilities to a translation lexicon.

(Ravi and Knight, 2011b) have shown that one can use decipherment to learn a full translation model from non-parallel data. Their approach is able to find translations, and assign probabilities to them. But their work also has certain limitations. First of all, the corpus they use to build the translation system has a very small vocabulary. Secondly, although their algorithm is able to handle word substitution ciphers with limited vocabulary, its deciphering accuracy is low.

The contributions of this work are:

- We improve previous decipherment work by introducing a more efficient sampling algorithm. In experiments, our new method improves deciphering accuracy from 82.5% to 88.1% on (Ravi and Knight, 2011b)'s domain specific data set. Furthermore, we also solve a very large word substitution cipher built from the English Gigaword corpus and achieve 92.2% deciphering accuracy on news text.

- With the ability to handle a much larger vocabulary, we learn a domain specific translation table from a large amount of monolingual data and use the translation table to improve out-of-domain machine translation. In experiments, we observe significant gains of up to 3.8 BLEU points. Unlike previous works, the translation table we build from monolingual data do not only contain unseen words but also words seen in parallel data.

2 Word Substitution Ciphers

Before we present our new decipherment framework, we quickly review word substitution decipherment.

Recently, there has been an increasing interest in decipherment work (Ravi and Knight, 2011a; Ravi and Knight, 2008). While letter substitution ciphers can be solved easily, nobody has been able to solve a word substitution cipher with high accuracy.

As shown in Figure 1, a word substitution cipher is generated by replacing each word in a natural language (plaintext) sequence with a cipher token according to a substitution table. The mapping in the table is deterministic – each plaintext word type is only encoded with one unique cipher token. Solving a word substitution cipher means recovering the original plaintext from the ciphertext without knowing the substitution table. The only thing we rely on is knowledge about the underlying language.

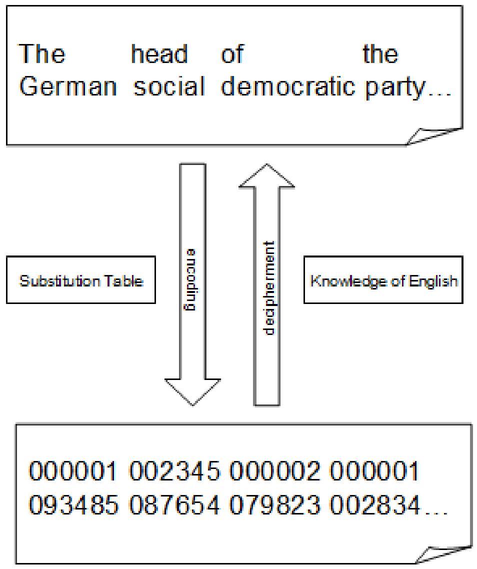

Figure 1: Encoding and Decipherment of a Word Substitution Cipher

How can we solve a word substitution cipher? The approach is similar to those taken by cryptanalysts who try to recover keys that convert encrypted texts to readable texts. Suppose we observe a large cipher string f and want to decipher it into English e. We can follow the work in (Ravi and Knight, 2011b) and assume that the cipher string f is generated in the following way:

- Generate English plaintext sequence $e = e_1, e_2...e_n$ with probability P(e).
- Replace each English plaintext token e_i with a cipher token f_i with probability $P(f_i|e_i)$.

Based on the above generative story, we write the probability of the cipher string f as:

$$P(f)_\theta = \sum_e P(e) \cdot \prod_i^n P_\theta(f_i|e_i) \qquad (1)$$

We use this equation as an objective function for maximum likelihood training. In the equation, $P(e)$ is given by an ngram language model, which is trained using a large amount of monolingual texts. The rest of the task is to manipulate channel probabilities $P_\theta(f_i|e_i)$ so that the probability of the observed texts $P(f)_\theta$ is maximized.

Theoretically, we can directly apply EM, as proposed in (Knight et al., 2006), or Bayesian decipherment (Ravi and Knight, 2011a) to solve the problem. However, unlike letter substitution ciphers, word substitution ciphers pose much greater challenges to algorithm scalability. To solve a word substitution cipher, the EM algorithm has a computational complexity of $O(N \cdot V^2 \cdot R)$ and the complexity of Bayesian method is $O(N \cdot V \cdot R)$, where V is the size of plaintext vocabulary, N is the length of ciphertext, and R is the number of iterations. In the world of word substitution ciphers, both V and N are very large, making these approaches impractical. (Ravi and Knight, 2011b) propose several modifications to the existing algorithms. However, the modified algorithms are only an approximation of the original algorithms and produce poor deciphering accuracy, and they are still unable to handle very large scale ciphers.

To address the above problems, we propose the following two new improvements to previous decipherment methods.

- We apply slice sampling (Neal, 2000) to scale up to ciphers with a very large vocabulary.
- Instead of deciphering using the original ciphertext, we break the ciphertext into bigrams, collect their counts, and use the bigrams with their counts for decipherment.

The new improvements allow us to solve a word substitution cipher with billions of tokens and hundreds of thousands of word types. Through better approximation, we achieve a significant increase in deciphering accuracy. In the following section, we present details of our new approach.

3 Slice Sampling for Bayesian Decipherment

In this section, we first give an introduction to Bayesian decipherment and then describe how to use slice sampling for it.

3.1 Bayesian Decipherment

Bayesian inference has been widely used in natural language processing (Goldwater and Griffiths, 2007; Blunsom et al., 2009; Ravi and Knight, 2011b). It is very attractive for problems like word substitution ciphers for the following reasons. First, there are no memory bottlenecks as compared to EM, which has an $O(N \cdot V^2)$ space complexity. Second, priors encourage the model to learn a sparse distribution.

The inference is usually performed using Gibbs sampling. For decipherment, a Gibbs sampler keeps drawing samples from plaintext sequences according to derivation probability $P(d)$:

$$P(d) = P(e) \cdot \prod_{i}^{n} P(c_i|e_i) \tag{2}$$

In Bayesian inference, $P(e)$ is still given by an ngram language model, while the channel probability is modeled by the Chinese Restaurant Process (CRP):

$$P(c_i|e_i) = \frac{\alpha \cdot prior + count(c_i, e_i)}{\alpha + count(e_i)} \tag{3}$$

Where $prior$ is the base distribution (we set prior to $\frac{1}{C}$ in all our experiments, where C is the number of word types in ciphertext), and count, also called "cache", records events that occurred in the history. Each sampling operation involves changing a plaintext token e_i, which has V possible choices, where V is the plaintext vocabulary size, and the final sample is chosen with probability $\frac{P(d)}{\sum_{n=1}^{V} P(d)}$.

3.2 Slice Sampling

With Gibbs sampling, one has to evaluate all possible plaintext word types (10k—1M) for each sample decision. This become intractable when the vocabulary is large and the ciphertext is long. Slice sampling (Neal, 2000) can solve this problem by automatically adjusting the number of samples to be considered for each sampling operation.

Suppose the derivation probability for current sample is $P(current\_s)$. Then slice sampling draws a sample in two steps:

- Select a threshold T uniformly from the range $\{0, P(current\_s)\}$.
- Draw a new sample new_s uniformly from a pool of candidates: $\{new\_s|P(new\_s) > T\}$.

From the above two steps, we can see that given a threshold T, we only need to consider those samples whose probability is higher than the threshold. This will lead to a significant reduction on the number of samples to be considered, if probabilities of the most samples are below T. In practice, the first step is easy to implement, while it is difficult to make the second step efficient. An obvious way to collect candidate samples is to go over all possible samples and record those with probabilities higher than T. However, doing this will not save any time. Fortunately, for Bayesian decipherment, we are able to complete the second step efficiently.

According to Equation 1, the probability of the current sample is given by a language model $P(e)$ and a channel model $P(c|e)$. The language model is usually an ngram language model. Suppose our current sample $current\_s$ contains English tokens X, Y, and Z at position $i-1$, i, and $i+1$ respectively. Let c_i be the cipher token at position i. To obtain a new sample, we just need to change token Y to Y'. Since the rest of the sample stays the same, we only need to calculate the probability of any trigram [1]: $P(XY'Z)$ and the channel model probability: $P(c_i|Y')$, and multiply them together as shown in Equation 4.

$$P(XY'Z) \cdot P(c_i|Y') \tag{4}$$

[1]The probability is given by a bigram language model.

In slice sampling, each sampling operation has two steps. For the first step, we choose a threshold T uniformly between 0 and $P(XYZ) \cdot P(c_i|Y)$. For the second step, there are two cases.

First, we notice that two types of Y' are more likely to pass the threshold T: (1) Those that have a very high trigram probability , and (2) those that have high channel model probability. To find candidates that have high trigram probability, we build sorted lists ranked by $P(XY'Z)$, which can be precomputed off-line. We only keep the top K English words for each of the sorted list. When the last item Y_K in the list satisfies $P(XY_kZ) \cdot prior < T$, We draw a sample in the following way: set $A = \{Y'|P(XY'Z) \cdot prior > T\}$ and set $B = \{Y'|count(c_i, Y') > 0\}$, then we only need to sample Y' uniformly from $A \cup B$ until Equation 4 is greater than T. [2]

Second, what happens when the last item Y_K in the list does not satisfy $P(XY_kZ) \cdot prior < T$? Then we always choose a word Y' randomly and accept it as a new sample if Equation 4 is greater than T.

Our algorithm alternates between the two cases. The actual number of choices the algorithm looks at depends on the K and the total number of possible choices. In different experiments, we find that when $K = 500$, the algorithm only looks at 0.06% of all possible choices. When $K = 2000$, this further reduces to 0.007%.

3.3 Deciphering with Bigrams

Since we can decipher with a bigram language model, we posit that a frequency list of ciphertext bigrams may contain enough information for decipherment. In our letter substitution experiments, we find that breaking ciphertext into bigrams doesn't hurt decipherment accuracy. Table 1 shows how full English sentences in the original data are broken into bigrams and their counts.

	took	our	2
man they took our land .	they	took	2
they took our arable land .	land	.	2
	man	they	1
	arable	land	1

Table 1: Converting full sentences to bigrams

Instead of doing sampling on full sentences, we treat each bigram as a full "sentence". There are two advantages to use bigrams and their counts for decipherment.

First of all, the bigrams and counts are a much more compact representation of the original ciphertext with full sentences. For instance, after breaking a billion tokens from the English Gigaword corpus, we find only 29m bigrams and 58m tokens, which is only 1/17 of the original text. In practice, we further discard all bigrams with low frequency, which makes the ciphertext even shorter.

Secondly, using bigrams significantly reduces the number of sorted lists (from $|V|^2$ to $2|V|$) mentioned in the previous section. The number of lists reduces from $|V|^2$ to $2|V|$ because words in a bigram only have one neighbor. Therefore, for any word W in a bigram, we need only $2|V|$ lists ("words to the right of W" and "words to the left of W") instead of $|V|^2$ lists ("pairs of words that surround W").

3.4 Iterative Sampling

Although we can directly apply slice sampling on a large number of bigrams, we find that gradually including less frequent bigrams into a sampling process saves deciphering time – we call this iterative sampling:

- Break the ciphertext into bigrams and collect their counts
- Keep bigrams whose counts are greater than a threshold α. Then initialize the first sample randomly and use slice sampling to perform maximum likelihood training. In the end, extract a translation table T according to the final sample.
- Lower the threshold α to include more bigrams into the sampling process. Initialize the first sample using the translation table obtained from the previous sampling run (for each ci-

[2]It is easy to prove that all other candidates that are not in the sorted list and with $count(c_i, Y') = 0$ have a upper bound probability: $P(XY_kZ) \cdot prior$. Therefore, they are ignored when $P(XY_kZ) \cdot prior < T$.

pher token f, choose a plaintext token e whose $P(e|f)$ is the largest). Perform sampling again.

- Repeat until $\alpha = 1$.

3.5 Parallel Sampling

Inspired by (Newman et al., 2009), our parallel sampling procedure is described below:

- Collect bigrams and their counts from ciphertext and split the bigrams into N parts.
- Run slice sampling on each part for 5 iterations independently.
- Combine counts from each part to form a new count table and run sampling again on each part using the new table.[3]

4 Decipherment Experiments

In this section, we evaluate our new sampling algorithm in two different experiments. In the first experiment, we compare our method with (Ravi and Knight, 2011b) on their data set to prove correctness of our approach. In the second experiment, we scale up to the whole English Gigaword corpus and achieve a much higher deciphering accuracy.

4.1 Deciphering Transtac Corpus

4.1.1 Data

We split the Transtac corpus the same way it was split in (Ravi and Knight, 2011b). The data used to create ciphertext consists of 1 million tokens, and 3397 word types. The data for language model training contains 2.7 million tokens and 25761 word types.[4] The ciphertext is created by replacing each English word with a cipher word.

We use a bigram language model for decipherment training. When the training terminates, a translation table with probability $P(c|e)$ is built based on the counts collected from the final sample. For decoding, we employ a trigram language model using full sentences. We use Moses (Koehn et al., 2007) to perform the decoding. We set the distortion limit to 0 and cube the translation probabilities. Essentially, Moses tries to find an English sequence e that maximizes $P(e) \cdot P(c|e)^3$

[3]Except for combining the counts to form a new count table, other parameters remain the same. For instance, each part i has its own prior set to $\frac{1}{C_i}$, where C_i is the number of word types in that part of ciphertext.

[4]In practice, we replaced singletons with a "UNK" symbol, leaving around 16904 word types.

Method	Deciphering Accuracy
Ravi and Knight	80.0 (with bigram LM)
	82.5 (with trigram LM)
Slice Sampling	88.1 (with bigram LM)

Table 2: Decipherment Accuracy on Transtac Corpus from (Ravi and Knight, 2011b)

Gold	Decoded
man i've come to file a complaint against some people .	man i've come to hand a telephone lines some people .
man they took our land .	man they took our farm .
they took our arable land .	they took our slide door .
okay man .	okay man .
eighty donums .	miflih donums .

Table 3: Sample Decoding Results on Transtac Corpus from (Ravi and Knight, 2011b)

4.1.2 Results

We evaluate the performance of our algorithm by decipherment accuracy, which measures the percentage of correctly deciphered cipher tokens. Table 2 compares the deciphering accuracy with the state of the art algorithm.

Results show that our algorithm improves the deciphering accuracy to 88.1%, which amounts to 33% reduction in error rate. This justifies our claim: doing better approximation using slice sampling improves decipherment accuracy.

Table 3 shows the first 5 decoding results and compares them with the gold plaintext. From the table we can see that the algorithm recovered the majority of the plaintext correctly.

4.2 Deciphering Gigaword Corpus

To prove the scalability of our new approach, we apply it to solve a much larger word substitution cipher built from English Gigaword corpus. The corpus contains news articles from different news agencies and has a much larger vocabulary compared with the Transtac corpus.

4.2.1 Data

We split the corpus into two parts chronologically. Each part contains approximately 1.2 billion tokens. We uses the first part to build a word substitution cipher, which is 10k times longer than the one in the previous experiment, and the second part to build a bigram language model. [5]

4.2.2 Results

We first use a single machine and apply iterative sampling to solve a 68 million token cipher. Then we use the result from the first step to initialize our parallel sampling process, which uses as many as 100 machines. For evaluation, we calculate deciphering accuracy over the first 1000 sentences (33k tokens).

After 2000 iterations of the parallel sampling process, the deciphering accuracy reaches 92.2%. Figure 2 shows the learning curve of the algorithm. It can be seen from the graph that both token and type accuracy increase as more and more data becomes available.

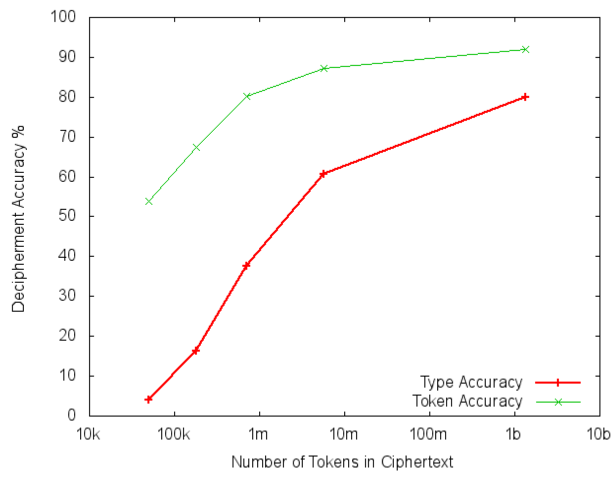

Figure 2: Learning curve for a very large word substitution cipher: Both token and type accuracy rise as more and more ciphertext becomes available.

[5]Before building the language model, we replace low frequency word types with an ”UNK” symbol, leaving 129k unique word types.

5 Improving Out-of-Domain Machine Translation

Domain specific machine translation (MT) is a challenge for statistical machine translation (SMT) systems trained on parallel corpora. It is common to see a significant drop in translation quality when translating out-of-domain texts. Although it is hard to find parallel corpora for any specific domain, it is relatively easy to find domain specific monolingual corpora. In this section, we show how to use our new decipherment framework to learn a domain specific translation table and use it to improve out-of-domain translations.

5.1 Baseline SMT System

We build a state of the art phrase-based SMT system using Moses (Koehn et al., 2007). The baseline system has 3 models: a translation model, a reordering model, and a language model. The language model can be trained on monolingual data, and the rest are trained on parallel data. By default, Moses uses the following 8 features to score a candidate translation:

- direct and inverse translation probabilities
- direct and inverse lexical weighting
- phrase penalty
- a language model
- a re-ordering model
- word penalty

Each of the 8 features has its own weight, which can be tuned on a held-out set using minimum error rate training. (Och, 2003). In the following sections, we describe how to use decipherment to learn domain specific translation probabilities, and use the new features to improve the baseline.

5.2 Learning a New Translation Table with Decipherment

From a decipherment perspective, machine translation is a much more complex task than solving a word substitution cipher and poses three major challenges:

- Mappings between languages are nondeterministic, as words can have multiple translations

- Re-ordering of words
- Insertion and deletion of words

Fortunately, our decipherment model does not assume deterministic mapping and is able to discover multiple translations. For the reordering problem, we treat Spanish as a simple word substitution for French. Despite the simplification in the assumption, we still expect to learn a useful word-to-word lexicon via decipherment and use the lexicon to improve our baseline.

Problem formulation: By ignoring word reorderings, we can formulate MT decipherment problem as word substitution decipherment. We view source language f as ciphertext and target language e as plaintext. Our goal is to decipher f into e and estimate translation probabilities based on the decipherment.

Probabilistic decipherment: Similar to solving a word substitution cipher, all we have to do here is to estimate the translation model parameters $P_\theta(f|e)$ using a large amount of monolingual data in f and e respectively. According to Equation 5, our objective is to estimate the model parameters so that the probability of source text P(f) is maximized.

$$\arg\max_\theta \sum_e P(e) \cdot \prod_i^n P_\theta(f_i|e_i) \tag{5}$$

Building a translation table: Once the sampling process completes, we estimate translation probability $P(f|e)$ from the final sample using maximum likelihood estimation. We also decipher from the reverse direction to estimate $P(e|f)$. Finally, we build a phrase table by taking translation pairs seen in both decipherments.

5.3 Combining Phrase Tables

We now have two phrase tables: one learnt from parallel corpus and one learnt from non-parallel monolingual corpus through decipherment. The phrase table learnt through decipherment only contains word to word translations, and each translation option only has two scores. Moses has a function to decode with multiple phrase tables, so we just need to add the newly learnt phrase table and specify two more weights for the scores in it. During decoding, if a source word only appears in the decipherment table, that table's translation will be used. If a source word exists in both tables, Moses will create two separate decoding paths and choose the best one after taking other features into account. If a word is not seen in either of the tables, it is copied literally to the output.

Train	French: 28.5 million tokens Spanish: 26.6 million tokens
Tune	French: 28k tokens Spanish: 26k tokens
Test	French: 30k tokens Spanish: 28k tokens

Table 4: Europarl Training, Tuning, and Testing Data

6 MT Experiments and Results

6.1 Data

In our MT experiments, we translate French into Spanish and use the following corpora to learn our translation systems:

- Europarl Corpus (Koehn, 2005): The Europarl parallel corpus is extracted from the proceedings of the European Parliament and includes versions in 11 European languages. The corpus contains articles from the political domain and is used to train our baseline system. We use the 6th version of the corpus. After cleaning, there are 1.3 million lines left for training. We use the last 2k lines for tuning and testing (1k for each), and the rest for training. Details of training, tuning, and testing data are listed in Table 4.

- EMEA Corpus (Tiedemann, 2009): EMEA is a parallel corpus made out of PDF documents from the European Medicines Agency. It contains articles from the medical domain, which is a good test bed for out-of-domain tasks. We use the first 2k pairs of sentences for tuning and testing (1k for each), and use the rest (1.1 million lines) for decipherment training. We split the training corpus in ways that no parallel sentences are included in the training set. The splitting methods are listed in Table 5.

For decipherment training, we use lexical translation tables learned from the Europarl corpus to ini-

Comparable EMEA : French: Every odd line, 8.7 million tokens Spanish: Every even line, 8.1 million tokens
Non-parallel EMEA: French: First 550k sentences, 9.1 million tokens Spanish: Last 550k sentences, 7.7 million tokens

Table 5: EMEA Decipherment Training Data

tialize our sampling process.

6.2 Results

BLEU (Papineni et al., 2002) is used as a standard evaluation metric. We compare the following 3 systems in our experiments, and present the results in Table 6.

- **Baseline:** Trained on Europarl
- **Decipher-CP:** Trained on Europarl + Comparable EMEA
- **Decipher-NP:** Trained on Europarl + Non-Parallel EMEA

Our baseline system achieves 38.2 BLEU score on Europarl test set. In the second row of Table 6, the test set changes to EMEA, and the baseline BLEU score drops to 24.9. In the third row, the baseline score rises to 30.5 with a language model built from EMEA corpus. Although it is much higher than the previous baseline, we further improve it by including a new phrase table learnt from domain specific monolingual data. In a real out-of-domain task, we are unlikely to have any parallel data to tune weights for the new phrase table. Therefore, we can only set it manually. In experiments, each score in the new phrase table has a weight of 5, and the BLEU score rises up to 33.2. In the fourth row of the table, we assume that there is a small amount of domain specific parallel data for tuning. With better weights, our baseline BLEU score increases to 37.3, and our combined systems increase to 41.1 and 39.7 respectively. In the last row of the table, we compare the combined systems with an even better baseline. This time, the baseline is given half of the EMEA tuning set for training and uses the other half for weight tuning. Results show that our combined systems still outperform the baseline.

French	Spanish	$P(fr\|es)$	$P(es\|fr)$
<	<	0.32	1.00
hépatique	hepático	0.88	0.08
	hepática	0.76	0.85
injectable	inyectable	0.91	0.92
dl	dl	1.00	0.70
>	>	0.32	1.00
ribavirine	ribavirina	0.40	1.00
olanzapine	olanzapina	0.57	1.00
clairance	aclaramiento	0.99	0.64
pelliculéss	recubiertos	1.00	1.00
pharmaco-cinétique	farmaco-cinético	1.00	1.00

Table 7: 10 most frequent OOV words in the table learnt from non-parallel EMEA corpus

The phrase table learnt from monolingual data consists of both observed and unknown words. Table 7 shows the top 10 most frequent OOV words in the table learnt from non-parallel EMEA corpus. Among the 10 words, 9 have correct translations. It is interesting to see that our algorithm finds multiple correct translations for the word "hépatique". The only mistake in the table is sensible as French word "pelliculés" is translated into "recubiertos con película" in Spanish.

7 Conclusion and Future Work

We apply slice sampling to Bayesian Decipherment and show significant improvement in deciphering accuracy compared with the state of the art algorithm. Our method is not only accurate but also highly scalable. In experiments, we decipher at the scale of the English Gigaword corpus, which contains over billions of tokens and hundreds of thousands word types. We further show the value of our new decipherment algorithm by using it to improve out-of-domain translation. In the future, we will work with more language pairs, especially those with significant word re-orderings. Moreover, the monolingual corpora used in the experiments are far smaller than what our algorithm can handle. We will continue to work in scenarios where large amount of monolingual data is readily available.

Train Data	Tune Data	Tune LM	Test Data	Test LM	Baseline	Decipher-CP	Decipher-NP
Europarl	Europarl	Europarl	Europarl	Europarl	38.2		
Europarl	Europarl	Europarl	EMEA	Europarl	24.9		
Europarl	Europarl	Europarl	EMEA	EMEA	30.5	33.2 (+2.7)	32.4 (+1.9)
Europarl	EMEA	EMEA	EMEA	EMEA	37.3	41.1 (+3.8)	39.7 (+2.4)
Europarl + EMEA	EMEA	EMEA	EMEA	EMEA	67.4	68.7 (+1.3)	68.7 (+1.3)

Table 6: MT experiment results: The table shows how much the combined systems outperform the baseline system in different experiments. Each row has a different set of training, tuning, and testing data. Baseline is trained on parallel data only. Tune LM and Test LM specify language models used for tuning and testing respectively. Decipher-CP and Decipher-NP use a phrase table learnt from comparable and non-parallel EMEA corpus respectively.

8 Acknowledgments

This work was supported by NSF Grant 0904684. The authors would like to thank Philip Koehen, David Chiang, Jason Riesa, Ashish Vaswani, and Hui Zhang for their comments and suggestions.

References

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics.

Hal Daumé, III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Sharon Goldwater and Tom Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition*. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *In Proceedings of the Tenth Machine Translation Summit*, Phuket, Thailand. Asia-Pacific Association for Machine Translation.

Radford Neal. 2000. Slice sampling. *Annals of Statistics*, 31.

David Newman, Arthur Asuncion, Padhrai Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

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*. 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: Human Language Technologies*. 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*. Association for Computational Linguistics.

Jörg Tiedemann. 2009. News from OPUS – a collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing V*, volume 309 of *Current Issues in Linguistic Theory*. John Benjamins.

Hua Wu, Haifeng Wang, and Chengqing Zong. 2008. Domain adaptation for statistical machine translation with domain dictionary and monolingual corpora. In *Proceedings of the 22nd International Conference on Computational Linguistics*. Association for Computational Linguistics.