Deciphering Foreign Language

Sujith Ravi and Kevin Knight
University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi,knight}@isi.edu

Abstract

In this work, we tackle the task of machine translation (MT) without parallel training data. We frame the MT problem as a decipherment task, treating the foreign text as a cipher for English and present novel methods for training translation models from non-parallel text.

1 Introduction

Bilingual corpora are a staple of statistical machine translation (SMT) research. From these corpora, we estimate translation model parameters: word-to-word translation tables, fertilities, distortion parameters, phrase tables, syntactic transformations, etc. Starting with the classic IBM work (Brown et al., 1993), training has been viewed as a maximization problem involving hidden word alignments (a) that are assumed to underlie observed sentence pairs (e, f):

$$\arg\max_{\theta} \prod_{e,f} P_\theta(f|e) \tag{1}$$

$$= \arg\max_{\theta} \prod_{e,f} \sum_{a} P_\theta(f, a|e) \tag{2}$$

Brown et al. (1993) give various formulas that boil $P_\theta(f, a|e)$ down to the specific parameters to be estimated.

Of course, for many language pairs and domains, parallel data is not available. In this paper, we address the problem of *learning a full translation model from non-parallel data*, and we use the learned model to translate new foreign strings. As successful work develops along this line, we expect more domains and language pairs to be conquered by SMT.

How can we learn a translation model from non-parallel data? Intuitively, we try to construct translation model tables which, when applied to observed foreign text, consistently yield sensible English. This is essentially the same approach taken by cryptanalysts and epigraphers when they deal with source texts.

In our case, we observe a large number of foreign strings f, and we apply maximum likelihood training:

$$\arg\max_{\theta} \prod_{f} P_\theta(f) \tag{3}$$

Following Weaver (1955), we imagine that this corpus of foreign strings "is really written in English, but has been coded in some strange symbols," thus:

$$\arg\max_{\theta} \prod_{f} \sum_{e} P(e) \cdot P_\theta(f|e) \tag{4}$$

The variable e ranges over all possible English strings, and $P(e)$ is a language model built from large amounts of English text that is unrelated to the foreign strings. Re-writing for hidden alignments, we get:

$$\arg\max_{\theta} \prod_{f} \sum_{e} P(e) \cdot \sum_{a} P_\theta(f, a|e) \tag{5}$$

Note that this formula has the same free $P_\theta(f, a|e)$ parameters as expression (2). We seek to manipulate these parameters in order to learn the

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 12–21,
Portland, Oregon, June 19-24, 2011.

same full translation model. We note that for each f, not only is the alignment a still hidden, but now the English translation e is hidden as well.

A language model $P(e)$ is typically used in SMT decoding (Koehn, 2009), but here $P(e)$ actually plays a central role in training translation model parameters. To distinguish the two, we refer to (5) as *decipherment*, rather than decoding.

We can now draw on previous decipherment work for solving simpler substitution/transposition ciphers (Bauer, 2006; Knight et al., 2006). We must keep in mind, however, that foreign language is a much more demanding code, involving highly non-deterministic mappings and very large substitution tables.

The contributions of this paper are therefore:

- We give first results for training a full translation model from non-parallel text, and we apply the model to translate previously-unseen text. This work is thus distinguished from prior work on extracting or augmenting partial lexicons using non-parallel corpora (Rapp, 1995; Fung and McKeown, 1997; Koehn and Knight, 2000; Haghighi et al., 2008). It also contrasts with self-training (McClosky et al., 2006), which requires a parallel seed and often does not engage in iterative maximization.

- We develop novel methods to deal with large-scale vocabularies inherent in MT problems.

2 Word Substitution Decipherment

Before we tackle machine translation without parallel data, we first solve a simpler problem—word substitution decipherment. Here, we do not have to worry about hidden alignments since there is only one alignment. In a word substitution cipher, every word in the natural language (plaintext) sequence is substituted by a cipher token, according to a substitution key. The key is deterministic—there exists a 1-to-1 mapping between cipher units and the plaintext words they encode.

For example, the following English plaintext sequences:

```
I SAW THE BOY .
THE BOY RAN .
```

may be enciphered as:

```
xyzz fxyy crqq tmnz lxwz
crqq tmnz gdxx lxwz
```

according to the key:

```
THE → crqq, SAW → fxyy, RAN → gdxx,
.  → lxwz, BOY → tmnz, I → xyzz
```

The goal of word substitution decipherment is to guess the original plaintext from given cipher data without any knowledge of the substitution key.

Word substitution decipherment is a good test-bed for unsupervised statistical NLP techniques for two reasons—(1) we face large vocabularies and corpora sizes typically seen in large-scale MT problems, so our methods need to scale well, (2) similar decipherment techniques can be applied for solving NLP problems such as unsupervised part-of-speech tagging.

Probabilistic decipherment: Our decipherment method follows a noisy-channel approach. We first model the process by which the ciphertext sequence $c = c_1...c_n$ is generated. The generative story for decipherment is described here:

1. Generate an English plaintext sequence $e = e_1...e_n$, with probability $P(e)$.

2. Substitute each plaintext word e_i with a ciphertext token c_i, with probability $P_\theta(c_i|e_i)$ in order to generate the ciphertext sequence $c = c_1...c_n$.

We model $P(e)$ using a statistical word n-gram English language model (LM). During decipherment, our goal is to estimate the channel model parameters θ. Re-writing Equations 3 and 4 for word substitution decipherment, we get:

$$\arg\max_{\theta} \prod_{c} P_\theta(c) \qquad (6)$$

$$= \arg\max_{\theta} \prod_{c} \sum_{e} P(e) \cdot \prod_{i=1}^{n} P_\theta(c_i|e_i) \qquad (7)$$

Challenges: Unlike letter substitution ciphers (having only 26 plaintext letters), here we have to deal with large-scale vocabularies (10k-1M word types) and corpora sizes (100k cipher tokens). This poses some serious scalability challenges for word substitution decipherment.

We propose novel methods that can deal with these challenges effectively and solve word substitution ciphers:

1. *EM solution:* We would like to use the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to estimate θ from Equation 7, but EM training is not feasible in our case. First, EM cannot scale to such large vocabulary sizes (running the forward-backward algorithm for each iteration requires $O(V^2)$ time). Secondly, we need to instantiate the entire channel and resulting derivation lattice before we can run EM, and this is too big to be stored in memory. So, we introduce a new training method (*Iterative EM*) that fixes these problems.

2. *Bayesian decipherment:* We also propose a novel decipherment approach using Bayesian inference. Typically, Bayesian inference is very slow when applied to such large-scale problems. Our method overcomes these challenges and does fast, efficient inference using (a) a novel strategy for selecting sampling choices, and (b) a parallelized sampling scheme.

In the next two sections, we describe these methods in detail.

2.1 Iterative EM

We devise a method which overcomes memory and running time efficiency issues faced by EM. Instead of instantiating the entire channel model (with all its parameters), we iteratively train the model in small steps. The training procedure is described here:

1. Identify the top K frequent word types in both the plaintext and ciphertext data. Replace all other word tokens with *Unknown*. Now, instantiate a small channel with just $(K + 1)^2$ parameters and use the EM algorithm to train this model to maximize likelihood of cipher data.

2. Extend the plaintext and ciphertext vocabularies from the previous step by adding the next K most frequent word types (so the new vocabulary size becomes $2K + 1$). Regenerate the plaintext and ciphertext data.

3. Instantiate a new $(2K+1) \times (2K+1)$ channel model. From the previous EM-trained channel, identify all the $e \rightarrow c$ mappings that were assigned a probability $P(c|e) > 0.5$. Fix these mappings in the new channel, i.e. set $P(c|e) = 1.0$. From the new channel, eliminate all other parameters $e \rightarrow c_j$ associated with the plaintext word type e (where $c_j \neq c$). This yields a much smaller channel with size $< (2K + 1)^2$. Retrain the new channel using EM algorithm.

4. Goto Step 2 and repeat the procedure, extending the channel size iteratively in each stage.

Finally, we decode the given ciphertext c by using the Viterbi algorithm to choose the plaintext decoding e that maximizes $P(e) \cdot P_{\theta_{trained}}(c|e)^3$, stretching the channel probabilities (Knight et al., 2006).

2.2 Bayesian Decipherment

Bayesian inference methods have become popular in natural language processing (Goldwater and Griffiths, 2007; Finkel et al., 2005; Blunsom et al., 2009; Chiang et al., 2010; Snyder et al., 2010). These methods are attractive for their ability to manage uncertainty about model parameters and allow one to incorporate prior knowledge during inference.

Here, we propose a novel decipherment approach using Bayesian learning. Our method holds several other advantages over the EM approach—(1) inference using smart sampling strategies permits efficient training, allowing us to scale to large data/vocabulary sizes, (2) incremental scoring of derivations during sampling allows efficient inference even when we use higher-order n-gram LMs, (3) there are no memory bottlenecks since the full channel model and derivation lattice are never instantiated during training, and (4) prior specification allows us to learn *skewed* distributions that are useful here—word substitution ciphers exhibit 1-to-1 correspondence between plaintext and cipher types.

We use the same generative story as before for decipherment, except that we use Chinese Restaurant Process (CRP) formulations for the source and channel probabilities. We use an English word bigram LM as the base distribution (P_0) for the source model and specify a uniform P_0 distribution for the channel.[1] We perform inference using point-wise Gibbs sampling (Geman and Geman, 1984). We define a sampling operator that samples plaintext word choices for every cipher token, one at a time. Using the exchangeability property, we efficiently score the probability of each derivation in an incremental fashion. In addition, we make further improvements to the sampling procedure which makes it faster.

Smart sample-choice selection: In the original sampling step, for each cipher token we have to sample from a list of all possible plaintext choices (10k-1M English words). There are 100k cipher tokens in our data which means we have to perform $\sim 10^9$ sampling operations to make one entire pass through the data. We have to then repeat this process for 2000 iterations. Instead, we now reduce our choices in each sampling step.

Say that our current plaintext hypothesis contains English words X, Y and Z at positions $i-1$, i and $i{+}1$ respectively. In order to sample at position i, we choose the top K English words Y ranked by $P(\text{X Y Z})$, which can be computed offline from a statistical word bigram LM. If this probability is 0 (i.e., X and Z never co-occurred), we randomly pick K words from the plaintext vocabulary. We set K = 100 in our experiments. This significantly reduces the sampling possibilities (10k-1M reduces to 100) at each step and allows us to scale to large plaintext vocabulary sizes without enumerating all possible choices at every cipher position.[2]

Parallelized Gibbs sampling: Secondly, we parallelize our sampling step using a Map-Reduce framework. In the past, others have proposed parallelized sampling schemes for topic modeling applications (Newman et al., 2009). In our method, we split the entire corpus into separate chunks and we run the sampling procedure on each chunk in parallel. At the end of each sampling iteration, we combine the samples corresponding to each chunk and collect the counts of all events—this forms our cache for the next sampling iteration. In practice, we observe that the parallelized sampling run converges quickly and runs much faster than the conventional point-wise sampling—for example, 3.1 hours (using 10 nodes) versus 11 hours for one of the word substitution experiments. We also notice a higher speedup when scaling to larger vocabularies.[3]

Decoding the ciphertext: After the sampling run has finished, we choose the final sample and extract a trained version of the channel model $P_\theta(c|e)$ from this sample following the technique of Chiang et al. (2010). We then use the Viterbi algorithm to choose the English plaintext e that maximizes $P(e) \cdot P_{\theta_{trained}}(c|e)^3$.

2.3 Experiments and Results

Data: For the word substitution experiments, we use two corpora:

- *Temporal expression corpus* containing short English temporal expressions such as “THE NEXT MONTH”, “THE LAST THREE YEARS”, etc. The cipher data contains 5000 expressions (9619 tokens, 153 word types). We also have access to a separate English corpus (which is not parallel to the ciphertext) containing 125k temporal expressions (242k word tokens, 201 word types) for LM training.
- *Transtac corpus* containing full English sentences. The data consists of 10k cipher sentences (102k tokens, 3397 word types); and a plaintext corpus of 402k English sentences (2.7M word tokens, 25761 word types) for LM training. We use all the cipher data for decipherment training but evaluate on the first 1000 cipher sentences.

The cipher data was originally generated from English text by substituting each English word with a unique cipher word. We use the plaintext corpus to

[1] For word substitution decipherment, we want to keep the language model probabilities fixed during training, and hence we set the prior on that model to be high ($\alpha = 10^4$). We use a sparse Dirichlet prior for the channel ($\beta = 0.01$). We use the output from Iterative EM decoding (using 101 x 101 channel) as initial sample and run the sampler for 2000 iterations. During sampling, we use a linear annealing schedule decreasing the temperature from $1 \rightarrow 0.08$.

[2] Since we now sample from an approximate distribution, we have to correct this with the Metropolis-Hastings algorithm. But in practice we observe that samples from our proposal distribution are accepted with probability > 0.99, so we skip this step.

[3] *Type sampling* could be applied on top of our methods to further optimize performance. But more complex problems like MT do not follow the same principles (1-to-1 key mappings) as seen in word substitution ciphers, which makes it difficult to identify type dependencies.

Method	**Decipherment Accuracy (%)**		
	Temporal expr.	*Transtac*	
		9k	100k
0. EM with 2-gram LM	87.8	Intractable	
1. Iterative EM			
with 2-gram LM	87.8	70.5	71.8
2. Bayesian			
with 2-gram LM	**88.6**	60.1	80.0
with 3-gram LM	-		**82.5**

Figure 1: Comparison of word substitution decipherment results using (1) Iterative EM, and (2) Bayesian method. For the *Transtac* corpus, decipherment performance is also shown for different training data sizes (9k versus 100k cipher tokens).

build an English word n-gram LM, which is used in the decipherment process.

Evaluation: We compute the accuracy of a particular decipherment as the percentage of cipher tokens that were correctly deciphered from the whole corpus. We run the two methods (Iterative EM[4] and Bayesian) and then compare them in terms of word substitution decipherment accuracies.

Results: Figure 1 compares the word substitution results from Iterative EM and Bayesian decipherment. Both methods achieve high accuracies, decoding 70-90% of the two word substitution ciphers. Overall, Bayesian decipherment (with sparse priors) performs better than Iterative EM and achieves the best results on this task. We also observe that both methods benefit from better LMs and more (cipher) training data. Figure 2 shows sample outputs from Bayesian decipherment.

3 Machine Translation as a Decipherment Task

We now turn to the problem of MT without parallel data. From a decipherment perspective, machine translation is a much more complex task than word substitution decipherment and poses several technical challenges: (1) scalability due to large corpora sizes and huge translation tables, (2) nondeterminism in translation mappings (a word can have multiple translations), (3) re-ordering of words or phrases, (4) a single word can translate into a phrase, and (5) insertion/deletion of words.

C:	*3894 9411 4357 8446 5433*
O:	a diploma that's good .
D:	a **fence** that's good .
C:	*8593 7932 3627 9166 3671*
O:	three families living here ?
D:	three **brothers** living here ?
C:	*6283 8827 7592 6959 5120 6137 9723 3671*
O:	okay and what did they tell you ?
D:	okay and what did they tell you ?
C:	*9723 3601 5834 5838 3805 4887 7961 9723 3174 4518 9067 4488 9551 7538 7239 9166 3671*
O:	you mean if we come to see you in the afternoon after five you'll be here ?
D:	**i** mean if we come to see you in the afternoon after **thirty** you'll be here ?
	...

Figure 2: Comparison of the original (O) English plaintext with output from Bayesian word substitution decipherment (D) for a few samples cipher (C) sentences from the *Transtac* corpus.

Problem Formulation: We formulate the MT decipherment problem as—given a foreign text f (i.e., foreign word sequences $f_1 ... f_m$) and a monolingual English corpus, our goal is to decipher the foreign text and produce an English translation.

Probabilistic decipherment: Unlike parallel training, here we have to estimate the translation model $P_\theta(f|e)$ parameters using only monolingual data. During decipherment training, our objective is to estimate the model parameters θ in order to maximize the probability of the foreign corpus f. From Equation 4 we have:

$$\arg\max_{\theta} \prod_{f} \sum_{e} P(e) \cdot P_\theta(f|e)$$

For $P(e)$, we use a word n-gram LM trained on monolingual English data. We then estimate parameters of the translation model $P_\theta(f|e)$ during training. Next, we present two novel decipherment approaches for MT training without parallel data.

1. *EM Decipherment:* We propose a new translation model for MT decipherment which can be efficiently trained using the EM algorithm.

2. *Bayesian Decipherment:* We introduce a novel method for estimating IBM Model 3 parameters without parallel data, using Bayesian learning. Unlike EM, this method does not face any

[4] For Iterative EM, we start with a channel of size 101x101 (K=100) and in every pass we iteratively increase the vocabulary sizes by 50, repeating the training procedure until the channel size becomes 351x351.

memory issues and we use sampling to perform efficient inference during training.

3.1 EM Decipherment

For the translation model $P_\theta(f|e)$, we would like to use a well-known statistical model such as IBM Model 3 and subsequently train it using the EM algorithm. But without parallel training data, EM training for IBM Model 3 becomes intractable due to (1) scalability and efficiency issues because of large-sized fertility and distortion parameter tables, and (2) the resulting derivation lattices become too big to be stored in memory.

Instead, we propose a simpler generative story for MT without parallel data. Our model accounts for (word) substitutions, insertions, deletions and local re-ordering during the translation process but does not incorporate fertilities or global re-ordering. We describe the generative process here:

1. Generate an English string $e = e_1...e_l$, with probability $P(e)$.

2. Insert a NULL word at any position in the English string, with uniform probability.

3. For each English word token e_i (including NULLs), choose a foreign word translation f_i, with probability $P_\theta(f_i|e_i)$. The foreign word may be NULL.

4. Swap any pair of adjacent foreign words f_{i-1}, f_i, with probability $P_\theta(swap)$. We set this value to 0.1.

5. Output the foreign string $f = f_1...f_m$, skipping over NULLs.

We use the EM algorithm to estimate all the parameters θ in order to maximize likelihood of the foreign corpus. Finally, we use the Viterbi algorithm to decode the foreign sentence f and produce an English translation e that maximizes $P(e) \cdot P_{\theta_{trained}}(f|e)$.

Linguistic knowledge for decipherment: To help limit translation model size and deal with data sparsity problem, we use prior linguistic knowledge. We use identity mappings for numeric values (for example, “8” maps to “8”), and we split nouns into morpheme units prior to decipherment training (for example, “YEARS” → “YEAR” “+S”).

Whole-segment Language Models: When using word n-gram models of English for decipherment, we find that some of the foreign sentences are decoded into sequences (such as “THANK YOU TALKING ABOUT ?”) that are not good English. This stems from the fact that n-gram LMs have no global information about what constitutes a valid English segment. To learn this information automatically, we build a $P(e)$ model that only recognizes English whole-segments (entire sentences or expressions) observed in the monolingual training data. We then use this model (in place of word n-gram LMs) for decipherment training and decoding.

3.2 Bayesian Method

Brown et al. (1993) provide an efficient algorithm for training IBM Model 3 translation model when parallel sentence pairs are available. But we wish to perform IBM Model 3 training under non-parallel conditions, which is intractable using EM training. Instead, we take a Bayesian approach.

Following Equation 5, we represent the translation model as $P_\theta(f, a|e)$ in terms of hidden alignments a. Recall the generative story for IBM Model 3 translation which has the following formula:

$$\begin{aligned} P_\theta(f,a|e) &= \prod_{i=0}^{l} t_\theta(f_{a_j}|e_i) \cdot \prod_{i=1}^{l} n_\theta(\phi_i|e_i) \\ &\quad \cdot \prod_{a_j \neq 0, j=1}^{m} d_\theta(a_j|i,l,m) \cdot \prod_{i=0}^{l} \phi_i! \\ &\quad \cdot \frac{1}{\phi_0!} \cdot \binom{m-\phi_0}{\phi_0} \\ &\quad \cdot p_{1_\theta}^{\phi_0} \cdot p_{0_\theta}^{m-2\phi_0} \end{aligned} \tag{8}$$

The alignment a is represented as a vector; $a_j = i$ implies that the foreign word f_j is produced by the English word e_i during translation.

Bayesian Formulation: Our goal is to learn the probability tables t (translation parameters) n (fertility parameters), d (distortion parameters), and p (English NULL word probabilities) without parallel data. In order to apply Bayesian inference for decipherment, we model each of these tables using a

Chinese Restaurant Process (CRP) formulation. For example, to model the translation probabilities, we use the formula:

$$t_\theta(f_j|e_i) \quad = \quad \frac{\alpha \cdot P_0(f_j|e_i) + C_{history}(e_i, f_j)}{\alpha + C_{history}(e_i)} \tag{9}$$

where, P_0 represents the base distribution (which is set to uniform) and $C_{history}$ represents the count of events occurring in the history (cache). Similarly, we use CRP formulations for the other probabilities (n, d and p). We use sparse Dirichlet priors for all these models (i.e., low values for α) and plug these probabilities into Equation 8 to get $P_\theta(f, a|e)$.

Sampling IBM Model 3: We use point-wise Gibbs sampling to estimate the IBM Model 3 parameters. The sampler is seeded with an initial English sample translation and a corresponding alignment for every foreign sentence. We define several sampling operators, which are applied in sequence one after the other to generate English samples for the entire foreign corpus. Some of the sampling operators are described below:

- TranslateWord(j): Sample a new English word translation for foreign word f_j, from all possibilities (including NULL).
- SwapSegment(i_1, i_2): Swap the alignment links for English words e_{i_1} and e_{i_2}.
- JoinWords(i_1, i_2): Eliminate the English word e_{i_1} and transfer its links to the word e_{i_2}.

During sampling, we apply each of these operators to generate a new derivation e, a for the foreign text f and compute its score as $P(e) \cdot P_\theta(f, a|e)$. These small-change operators are similar to the heuristic techniques used for greedy decoding by German et al. (2001). But unlike the greedy method, which can easily get stuck, our Bayesian approach guarantees that once the sampler converges we will be sampling from the true *posterior* distribution.

As with Bayesian decipherment for word substitution, we compute the probability of each new derivation incrementally, which makes sampling efficient. We also apply blocked sampling on top of point-wise sampling—we treat all occurrences of a particular foreign sentence as a single block and sample a single derivation for the entire block. We also parallelize the sampling procedure (as described in Section 2.2).[5]

Choosing the best translation: Once the sampling run finishes, we select the final sample and extract the corresponding English translations for every foreign sentence. This yields the final decipherment output.

3.3 MT Experiments and Results

Data: We work with the Spanish/English language pair and use the following corpora in our MT experiments:

- *Time corpus:* We mined English newswire text on the Web and collected 295k temporal expressions such as “LAST YEAR”, “THE FOURTH QUARTER”, “IN JAN 1968”, etc. We first process the data and normalize numbers and names of months/weekdays—for example, “1968” is replaced with “NNNN”, “JANUARY” with “[MONTH]”, and so on. We then translate the English temporal phrases into Spanish using an automatic translation software (Google Translate) followed by manual annotation to correct mistakes made by the software. We create the following splits out of the resulting parallel corpus:

  TRAIN (English): 195k temporal expressions (7588 unique), 382k word tokens, 163 types.

  TEST (Spanish): 100k temporal expressions (2343 unique), 204k word tokens, 269 types.

- *OPUS movie subtitle corpus:* This is a large open source collection of parallel corpora available for multiple language pairs (Tiedemann, 2009). We downloaded the parallel Spanish/English subtitle corpus which consists of aligned Spanish/English sentences from a collection of movie subtitles. For our MT experiments, we select only Spanish/English sentences with frequency > 10 and create the following train/test splits:

[5] For Bayesian MT decipherment, we set a high prior value on the language model (10^4) and use sparse priors for the IBM 3 model parameters t, n, d, p ($0.01, 0.01, 0.01, 0.01$). We use the output from EM decipherment as the initial sample and run the sampler for 2000 iterations, during which we apply annealing with a linear schedule (2 → 0.08).

Method	Decipherment Accuracy	
	Time expressions	*OPUS subtitles*
1a. *Parallel training* (MOSES)		
with 2-gram LM	5.6 (85.6)	26.8 (63.6)
with 5-gram LM	4.7 (88.0)	
1b. *Parallel training* (IBM 3 without distortion)		
with 2-gram LM	10.1 (78.9)	29.9 (59.6)
with whole-segment LM	9.0 (79.2)	
2a. *Decipherment* (EM)		
with 2-gram LM	37.6 (44.6)	67.2 (15.3)
with whole-segment LM	28.7 (48.7)	65.1 (19.3)
2b. *Decipherment* (Bayesian IBM 3)		
with 2-gram LM	34.0 (30.2)	66.6 (15.1)

Figure 3: Comparison of Spanish/English MT performance on the *Time* and *OPUS* test corpora achieved by various MT systems trained under (1) **parallel**—(a) MOSES, (b) IBM 3 without distortion, and (2) **decipherment** settings—(a) EM, (b) Bayesian. The scores reported here are normalized edit distance values with BLEU scores shown in parentheses.

TRAIN (English): 19770 sentences (1128 unique), 62k word tokens, 411 word types.

TEST (Spanish): 13181 sentences (1127 unique), 39k word tokens, 562 word types.

Both Spanish/English sides of **TRAIN** are used for parallel MT training, whereas decipherment uses only monolingual English data for training LMs.

MT Systems: We build and compare different MT systems under two training scenarios:

1. *Parallel training* using: (a) **MOSES**, a phrase translation system (Koehn et al., 2007) widely used in MT literature, and (b) a simpler version of **IBM Model 3 (without distortion parameters)** which can be trained tractably using the strategy of Knight and Al-Onaizan (1998).

2. *Decipherment without parallel data* using: (a) **EM method** (from Section 3.1), and (b) **Bayesian method** (from Section 3.2).

Evaluation: All the MT systems are run on the Spanish test data and the quality of the resulting English translations are evaluated using two different measures—(1) Normalized edit distance score (Navarro, 2001),[6] and (2) BLEU (Papineni et al., 2002), a standard MT evaluation measure.

Results: Figure 3 compares the results of various MT systems (using parallel versus decipherment training) on the two test corpora in terms of edit distance scores (a lower score indicates closer match to the gold translation). The figure also shows the corresponding BLEU scores in parentheses for comparison (higher scores indicate better MT output).

We observe that even without parallel training data, our decipherment strategies achieve MT accuracies comparable to parallel-trained systems. On the *Time* corpus, the best decipherment (Method 2a in the figure) achieves an edit distance score of 28.7 (versus 4.7 for MOSES). Better LMs yield better MT results for both parallel and decipherment training—for example, using a segment-based English LM instead of a 2-gram LM yields a 24% reduction in edit distance and a 9% improvement in BLEU score for EM decipherment.

We also investigate how the performance of different MT systems vary with the size of the training data. Figure 4 plots the BLEU scores versus training sizes for different MT systems on the *Time* corpus. Clearly, using more training data yields better performance for all systems. However, higher improvements are observed when using parallel data in comparison to decipherment training which only uses monolingual data. We also notice that the scores do not improve much when going beyond 10,000 train-

[6] When computing edit distance, we account for substitutions, insertions, deletions as well as local-swap edit operations required to convert a given English string into the (gold) reference translation.

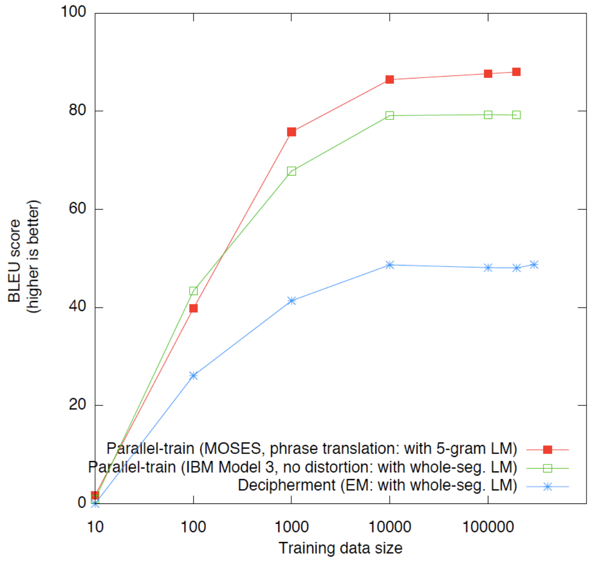

Figure 4: Comparison of *training data size* versus *MT accuracy* in terms of BLEU score under different training conditions: (1) **Parallel training**—(a) MOSES, (b) IBM Model 3 without distortion, and (2) **Decipherment** without parallel data using EM method (from Section 3.1).

ing instances for this domain.

It is interesting to quantify the value of parallel versus non-parallel data for any given MT task. In other words, "*how much non-parallel data is worth how much parallel data in order to achieve the same MT accuracy?*" Figure 4 provides a reasonable answer to this question for the Spanish/English MT task described here. We see that deciphering with 10k monolingual Spanish sentences yields the same performance as training with around 200-500 parallel English/Spanish sentence pairs. This is the first attempt at such a quantitative comparison for MT and our results are encouraging. We envision that further developments in unsupervised methods will help reduce this gap further.

4 Conclusion

Our work is the first attempt at doing MT without parallel data. We discussed several novel decipherment approaches for achieving this goal. Along the way, we developed efficient training methods that can deal with large-scale vocabularies and data sizes. For future work, it will be interesting to see if we can exploit both parallel and non-parallel data to improve on both.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation (NSF) under Grant No. IIS-0904684 and the Defense Advanced Research Projects Agency (DARPA) through the Department of Interior/National Business Center under Contract No. NBCHD040058. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of the Interior/National Business Center.

References

Friedrich L. Bauer. 2006. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer-Verlag.

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 Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 782–790.

Peter Brown, Vincent Della Pietra, Stephen Della Pietra, and Robert Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

David Chiang, Jonathan Graehl, Kevin Knight, Adam Pauls, and Sujith Ravi. 2010. Bayesian inference for finite-state transducers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL/HLT)*, pages 447–455.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Jenny Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370.

Pascal Fung and Kathleen McKeown. 1997. Finding terminology translations from non-parallel corpora. In *Proceedings of the Fifth Annual Workshop on Very Large Corpora*, pages 192–202.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 228–235.

Sharon Goldwater and Thomas 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*, pages 744–751.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics - Human Language Technologies (ACL/HLT)*, pages 771–779.

Kevin Knight and Yaser Al-Onaizan. 1998. Translation with finite-state devices. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Machine Translation and the Information Soup*, volume 1529 of *Lecture Notes in Computer Science*, pages 421–437. Springer Berlin / Heidelberg.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 499–506.

Philipp Koehn and Kevin Knight. 2000. Estimating word translation probabilities from unrelated monolingual corpora using the EM algorithm. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 711–715.

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*.

Philip Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys*, 33:31–88, March.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828.

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*, pages 311–318.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the Conference of the Association for Computational Linguistics*, pages 320–322.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. 2010. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1048–1057.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia.

Warren Weaver. 1955. Translation (1949). Reproduced in W.N. Locke, A.D. Booth (eds.). In *Machine Translation of Languages*, pages 15–23. MIT Press.