Feature-Based Decipherment for Machine Translation

Iftekhar Naim
Google
iftekhar.naim@gmail.com

Parker Riley
University of Rochester
Computer Science Department
priley3@cs.rochester.edu

Daniel Gildea
University of Rochester
Computer Science Department
gildea@cs.rochester.edu

Orthographic similarities across languages provide a strong signal for unsupervised probabilistic transduction (decipherment) for closely related language pairs. The existing decipherment models, however, are not well suited for exploiting these orthographic similarities. We propose a log-linear model with latent variables that incorporates orthographic similarity features. Maximum likelihood training is computationally expensive for the proposed log-linear model. To address this challenge, we perform approximate inference via Markov chain Monte Carlo sampling and contrastive divergence. Our results show that the proposed log-linear model with contrastive divergence outperforms the existing generative decipherment models by exploiting the orthographic features. The model both scales to large vocabularies and preserves accuracy in low- and no-resource contexts.

1. Introduction

Word-level translation models are typically learned by applying statistical word alignment algorithms on large-scale bilingual parallel corpora (Brown et al. 1993). Building a parallel corpus, however, is expensive and time-consuming. As a result, parallel data are limited or even unavailable for many language pairs. In the absence of a sufficient amount of parallel data, the accuracy of standard word alignment algorithms drops significantly. This is also true of supervised neural methods: Even with hundreds of thousands of parallel training sentences, neural methods only achieve modest results

Submission received: 9 October 2017; revised version received: 16 March 2018; accepted for publication: 15 May 2018.

doi:10.1162/COLI_a_00326

(Zoph et al. 2016). Low- and no-resource languages generally do not have parallel corpora, and even their monolingual corpora tend to be small. However, these monolingual corpora can often be downloaded from the Internet, and are much easier to obtain or produce than parallel corpora. Leveraging useful information from monolingual corpora can be extremely helpful for learning translation models for low- and no-resource language pairs.

Decipherment algorithms (so-called because of the assumption that one language is a cipher for the other) aim to exploit such monolingual corpora in order to learn translation model parameters, when parallel data are limited or unavailable (Koehn and Knight 2000; Ravi and Knight 2011; Dou, Vaswani, and Knight 2014). The key intuition is that similar words and *n*-grams tend to have similar distributional properties across languages. For example, if a bigram appears frequently in the monolingual source corpus, its translation is likely to appear frequently in the monolingual target corpus, and vice versa. This is especially true when the corpora share similar topics and context. Furthermore, for many such language pairs, we observe similar monotonic word ordering—that is, the translation of a bigram is often the same as the concatenation of the translations of individual unigrams (consider the shared use of postnominal adjectives in the French *maison bleu* and Spanish *casa azul*). Although this certainly is not always true, we assume that it is common enough to provide a useful signal. The goal of decipherment algorithms is to leverage such statistical similarities across languages, and effectively learn word-level translation probabilities from monolingual data.

Existing decipherment methods are predominantly based on probabilistic generative models (Koehn and Knight 2000; Ravi and Knight 2011; Dou and Knight 2012; Nuhn and Ney 2014). These models primarily focus on the statistical similarities between the *n*-gram frequencies in the source and the target language, and rely on the expectation maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) or its faster approximations. However, there can be many other types of statistical and linguistic similarities across languages beyond *n*-gram frequencies (similarities in spelling, word-length distribution, syntactic structure, etc.). Unfortunately, existing generative models do not allow incorporating such a wide range of linguistically motivated features. Previous research has shown the effectiveness of incorporating linguistically motivated features for many different unsupervised learning tasks, such as unsupervised part-of-speech induction (Haghighi and Klein 2006; Berg-Kirkpatrick et al. 2010), word alignment (Dyer et al. 2011; Ammar, Dyer, and Smith 2014), and grammar induction (Berg-Kirkpatrick et al. 2010).

Many pairs of related languages share vocabulary or grammatical structure due to borrowing or inheritance: the English *aquatic* and Spanish *agua* share the Latin root *aqua*, and the English *beige* was borrowed from French. As a result, orthographic features provide crucial information for determining word-level translations for closely related language pairs. Church (1993) leveraged orthographic similarity for character alignment. Haghighi, Berg-Kirkpatrick, and Klein (2008) proposed a generative model for inducing a bilingual lexicon from monolingual text by exploiting orthographic and contextual similarities among the words in two different languages. The model proposed by Haghighi et al. learns a one-to-one mapping between the words in two languages by analyzing type-level features only, while ignoring the token-level *n*-gram frequencies. We propose a decipherment model that unifies the type-level feature-based approach of Haghighi et al. with token-level EM-based approaches such as Koehn and Knight (2000) and Ravi and Knight (2011).

In addition to orthographic similarity, we also often observe similarity in the distribution of word lengths across different languages. Linguists have long noted the

relationship between word frequency and length (Zipf 1949), so the tendency of words and their translations to have similar frequencies (Rapp 1995) may apply to length as well. Our feature-rich log-linear model can easily incorporate such length-based similarity features.

One of the key challenges with the proposed latent variable log-linear model is the high computational complexity of training, as it requires normalizing globally via summing over all possible observations and latent variables. As a result, an exact implementation is impractical even for the moderate vocabulary size of most low-resource languages. To address this challenge, we perform approximate inference using Markov chain Monte Carlo (MCMC) sampling for scalable training of the log-linear decipherment models. We present a series of increasingly scalable approximations, each most suitable for a different amount of available data. They are applicable in contexts ranging from no-resource languages (such as "lost" languages, a context considered by Snyder, Barzilay, and Knight [2010]) to languages with a modest amount of data that is still insufficient for state-of-the-art unsupervised methods based on word embeddings.

The main contributions of this article are as follows.

- We propose a feature-based decipherment model for low- and no-resource languages that combines both type-level orthographic features and token-level distributional similarities. Our proposed model outperforms the existing EM-based decipherment models.
- We apply three different MCMC sampling strategies for scalable training and compare them in terms of running time and accuracy. Our results show that contrastive divergence (Hinton 2002)–based MCMC sampling can dramatically improve the speed of the training, while achieving comparable accuracy.
- We extend the contrastive divergence method to sample entire sentences, rather than bigram pairs, allowing more context to be used in reconstructing latent translations.
- Finally, we extend the model to exploit parallel as well as monolingual data, for situations in which limited amounts of parallel data may be available.

The remainder of the article is organized as follows. In Section 2, we introduce the general problem formulation for monolingual decipherment, and present our notations. We discuss the background literature on different decipherment models for machine translation in Section 3. Section 4 describes the proposed feature-based decipherment model. A detailed discussion of MCMC sampling–based approximations follows in Section 5. We extend the fully monolingual model to exploit parallel data in Section 6. Our orthographic features are described in Section 7. Finally, we present our detailed results in Section 8 and conclude with our findings and discuss our future work in Section 9.

2. Problem Formulation

Given a source text $\mathcal{F}$ and an independent target corpus $\mathcal{E}$, our goal is to translate the source text $\mathcal{F}$ by learning the mapping between the words in the source and the target

Table 1
Our notations and symbols.

Symbol	Meaning
N_F	Number of unique source bigrams
N_E	Number of unique target bigrams
V_F	Source vocabulary
V_E	Target vocabulary
V	$\max(\lvert V_F\rvert, \lvert V_E\rvert)$
N	Number of samples
K	Beam size for precomputed lists
ϕ	Unigram level feature function
$\mathbf{\Phi}$	Bigram level feature function: $\mathbf{\Phi} = \phi_1 + \phi_2$

language. Although the sentences in the source and target corpus are independent of each other, there exist distributional and lexical similarities among the words of the two languages. We aim to automatically learn the translation probabilities $p(f|e)$ for all source words f and target words e by exploiting the similarities between the bigrams in $\mathcal{F}$ and $\mathcal{E}$.

As a simplification step, we break down the sentences in the source and target corpus as a collection of bigrams. Let $\mathcal{F}$ contain a collection of source bigrams $f_1 f_2$, and $\mathcal{E}$ contain a collection of target bigrams $e_1 e_2$. Let the source and target vocabulary be V_F and V_E, respectively. Let N_F and N_E be the number of unique bigrams in $\mathcal{F}$ and $\mathcal{E}$, respectively. We assume that the corpus $\mathcal{F}$ is an encrypted version of a plaintext in the target language. Each source word $f \in V_F$ is obtained by substituting one of the words $e \in V_E$ in the plaintext. However, the mappings between the words in the two languages are unknown, and are learned as latent variables. Table 1 summarizes the notations and symbols used in this article.

3. Background Research

Existing decipherment models assume that each source bigram $f_1 f_2$ in $\mathcal{F}$ is generated by first generating a target bigram $e_1 e_2$ according to the target language model, and then substituting e_1 and e_2 with f_1 and f_2, respectively. The generative process is typically modeled via a hidden Markov model (HMM), as shown in Figure 1(a). The target bigram language model $p(e_1 e_2)$ is trained from the given monolingual target corpus $\mathcal{E}$. The unknown translation probabilities $p(f|e)$ are learned by maximizing the likelihood of the observed source corpus $\mathcal{F}$:

$$P(\mathcal{F}) = \prod_{f_1 f_2 \in \mathcal{F}} p(f_1 f_2) \tag{1}$$
$$= \prod_{f_1 f_2 \in \mathcal{F}} \sum_{e_1 e_2} p(e_1 e_2) p(f_1|e_1) p(f_2|e_2),$$

where e_1 and e_2 are the latent variables, indicating the target words in V_E corresponding to f_1 and f_2, respectively. The log-likelihood function with latent variables is non-convex,

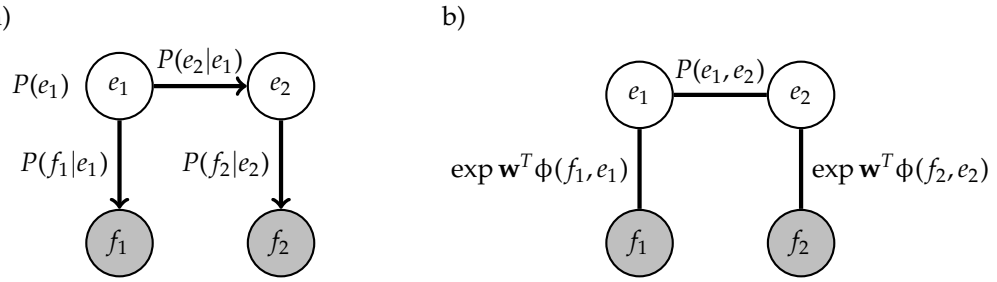

Figure 1
The graphical models for the existing directed HMM (a) and the proposed undirected MRF (b).

and several methods have been proposed for maximizing it. In this work, we seek to combine a number of them for improved performance.

3.1 Expectation Maximization (EM)

The expectation maximization (EM) (Dempster, Laird, and Rubin 1977) algorithm has been widely applied for solving the decipherment problem (Knight and Graehl 1998; Knight and Yamada 1999; Koehn and Knight 2000). In the E-step, for each source bigram $f_1 f_2$, we estimate the expected counts of the latent variables e_1 and e_2 over all the target words in V_E. In the M-step, the expected counts are normalized to obtain the translation probabilities $p(f|e)$. The computational complexity of the EM algorithm is $O(N_F V^2)$ and the memory complexity is $O(V^2)$, where N_F is the number of unique bigrams in $\mathcal{F}$ and $V = \max(|V_F|, |V_E|)$. As a result, the regular EM algorithm does not scale well to large vocabulary sizes, both in terms of running time and memory.

To address this challenge, Ravi and Knight (2011) proposed the iterative EM algorithm, which starts with the K most frequent words from $\mathcal{F}$ and $\mathcal{E}$ and performs EM-based decipherment. Next, the source and target vocabularies are iteratively extended by K new words, while pruning low-probability entries from the probability table. The computational complexity of each iteration becomes $O(N_F K^2)$.

3.2 Bayesian Decipherment Using Gibbs Sampling

Ravi and Knight (2011) proposed a Gibbs sampling–based Bayesian decipherment strategy. For each observed source bigram $f_1 f_2$, the Gibbs sampling approach starts with an initial target bigram $e_1 e_2$, and alternately fixes one of the target words and replaces the other with a randomly chosen sample. When e_1 is fixed, a new sample e_2^{new} is drawn with probability proportional to $p(e_1 e_2^{new}) p(f_2|e_2^{new})$. Next, we fix e_2 and sample e_1^{new}, and continue alternating until n samples are collected. Bayesian decipherment reduces memory consumption via Gibbs sampling. The probability table remains sparse, because only a small number of word pairs (f, e) will be observed together in the samples.

3.3 Slice Sampling

Each Gibbs sampling operation requires estimating the probability of choosing every target word $e \in V_E$, which requires $O(V)$ operations. To address this issue, Dou and Knight (2012) proposed a slice sampling approach with precomputed top-K lists for $p(e|f)$ and $p(e_1e_2)$. Slice sampling involves selecting a threshold T between 0 and the probability of the current sample, and then uniformly picking a random new sample from all candidates with probability greater than T. Using sorted top-K lists makes this faster than Gibbs sampling on average, although sometimes the top-K lists fail to provide all the candidates, in which case the method has to fall back to sampling from the entire vocabulary, which requires $O(V)$ operations.

3.4 Beam Search

Nuhn and colleagues (Nuhn, Schamper, and Ney 2013; Nuhn and Ney 2014; Nuhn, Schamper, and Ney 2015) showed that beam search can significantly improve the speed of EM-based decipherment, while providing comparable or even slightly better accuracy. Beam search prunes less-promising latent states by maintaining two constant-sized beams, one for the translation probabilities $p(f|e)$ and one for the target bigram probabilities $p(e_1e_2)$—reducing the computational complexity to $O(N_F)$. Furthermore, it saves memory because many of the word pairs (f, e) are never considered because they are not in the beam.

3.5 Feature-Based Generative Models

Haghighi, Berg-Kirkpatrick, and Klein (2008) proposed a canonical correlation analysis–based model for automatically learning the mapping between the words in two languages from monolingual corpora only. They used orthographic information (character substring features) and context information (co-occurrence statistics within a window) for their features; we use edit distance as our orthographic information, and we operate on bigrams for our context information. Although their model uses an EM-style algorithm, it does not iterate over the corpus data.

Ravi (2013) proposed a Bayesian decipherment model based on hash sampling, which takes advantage of feature-based similarities between source and target words. However, the feature representation was not integrated with their decipherment model, and was only used for efficiently sampling candidate target translations for each source word. Furthermore, the feature-based hash sampling included only contextual features (in the form of n-gram co-occurrence information), and did not consider orthographic features. In contrast, our log-linear model integrates both type-level orthographic features and token-level bigram frequencies.

3.6 Embedding-Based Models

Recent work has explored the possibility of finding a mapping between word embedding spaces using monolingual data. Artetxe, Labaka, and Agirre (2017) use a small set of seed translations to learn this mapping. Zhang et al. (2017) do not use seed

translations, but do use document-aligned Wikipedia data, and only consider words appearing at least 1,000 times. Both methods train word embeddings using data sets with millions of words, limiting their applicability to low resource languages, even more so for languages with the small amount of data that we experiment with in this work.

4. Feature-Based Decipherment

Our feature-based decipherment model is based on a chain structured Markov random field (MRF; Figure 1(b)), which jointly models the observed source bigrams $f_1 f_2$ and corresponding latent target bigram $e_1 e_2$. For each source word $f \in V_F$, we have a latent variable $e \in V_E$ indicating the corresponding target word. The joint probability distribution is:

$$p(f_1 f_2, e_1 e_2) = \frac{1}{Z_{\mathbf{w}}} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

where $\mathbf{\Phi}(f_1 f_2, e_1 e_2)$ is the feature function for the given source and the target bigrams, $\mathbf{w}$ is the model parameters, and $Z_{\mathbf{w}}$ is the normalization term. We assume that the feature function decomposes into features of aligned word pairs (motivated by the observation in Section 1 that word order is generally preserved across bigram translations):

$$\mathbf{\Phi}(f_1 f_2, e_1 e_2) = \phi(f_1, e_1) + \phi(f_2, e_2) \tag{2}$$

The features ϕ, which will be described in more detail in Section 7, include features for orthographic similarity as well as indicator features $\phi_{f,e}$ for each word pair. The normalization term is defined as:

$$Z_{\mathbf{w}} = \sum_{f_1 f_2} \sum_{e_1 e_2} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

This gives our model the conditional random field (CRF)–like dependency structure shown in Figure 1. In our model, however, the term $p(e_1, e_2)$ is estimated from a monolingual target corpus, and is held constant when training the weights $\mathbf{w}$.

We train the model on a monolingual source corpus, treating the target words as latent variables. This gives us a latent variable CRF model (Quattoni, Collins, and Darrell 2004), where the likelihood of our monolingual source corpus is:

$$L = \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \log \sum_{e_1 e_2} p(f_1 f_2, e_1 e_2) \tag{3}$$

The gradient of the log-likelihood can be written as the difference of two expectations of feature vectors:

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \log \sum_{e_1 e_2} p(f_1 f_2, e_1 e_2) \tag{4}$$

$$= \frac{\partial}{\partial \mathbf{w}} \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \log \sum_{e_1 e_2} \frac{1}{Z_{\mathbf{w}}} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \tag{5}$$

$$= \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \left[\frac{\partial}{\partial \mathbf{w}} \log \sum_{e_1 e_2} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) - \frac{\partial}{\partial \mathbf{w}} \log Z_{\mathbf{w}} \right] \tag{6}$$

$$= \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \left[\frac{1}{Z_{\mathbf{w}}(f_1 f_2)} \frac{\partial}{\partial \mathbf{w}} \sum_{e_1 e_2} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \right] - \frac{\partial}{\partial \mathbf{w}} \log Z_{\mathbf{w}} \tag{7}$$

$$= \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \left[\frac{1}{Z_{\mathbf{w}}(f_1 f_2)} \sum_{e_1 e_2} \mathbf{\Phi}(f_1 f_2, e_1 e_2) p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \right] - Z_{\mathbf{w}} \frac{\partial}{\partial \mathbf{w}} Z_{\mathbf{w}}$$

$$= \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \left[\sum_{e_1 e_2} \mathbf{\Phi}(f_1 f_2, e_1 e_2) p(f_1 f_2 | e_1 e_2) \right] - \frac{1}{Z_{\mathbf{w}}} \frac{\partial}{\partial \mathbf{w}} \sum_{f_1 f_2} \sum_{e_1 e_2} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

$$= \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \mathbb{E}_{e_1 e_2 | f_1 f_2} \left[\mathbf{\Phi}(f_1 f_2, e_1 e_2) \right] - \frac{1}{Z_{\mathbf{w}}} \sum_{f_1 f_2} \sum_{e_1 e_2} \mathbf{\Phi}(f_1 f_2, e_1 e_2) p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

$$= \frac{1}{|\mathcal{F}|} \sum_{f_1 f_2 \in \mathcal{F}} \mathbb{E}_{e_1 e_2 | f_1 f_2} \left[\mathbf{\Phi}(f_1 f_2, e_1 e_2) \right] - \mathbb{E}_{f_1 f_2, e_1 e_2} \left[\mathbf{\Phi}(f_1 f_2, e_1 e_2) \right] \tag{8}$$

$$= \mathbb{E}^{Forced} - \mathbb{E}^{Full} \tag{9}$$

Here, the first term is the expectation with respect to the empirical data distribution. We refer to it as the "forced expectation," as the source text is assumed to be given. The second term is the expectation with respect to our model distribution, and referred to as "full expectation."

4.1 Estimating Forced Expectation ($\mathbb{E}^{Forced}$)

We first estimate the forced expectation, which we defined in Equation (8) to be:

$$\mathbb{E}^{Forced} = \sum_{f_1 f_2 \in \mathcal{F}} \frac{1}{Z_{\mathbf{w}}(f_1 f_2)} \sum_{e_1 e_2 \in V_E^2} \left[p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \right] \mathbf{\Phi}(f_1 f_2, e_1 e_2) \tag{10}$$

where $Z(f_1 f_2)$ is the normalization term given $f_1 f_2$:

$$Z_{\mathbf{w}}(f_1 f_2) = \sum_{e_1 e_2 \in V_E^2} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

For each observed $f_1 f_2 \in \mathcal{F}$, we sum over all possible $e_1 e_2 \in V_E^2$, which requires $O(N_F V^2)$ computations.

4.2 Estimating Full Expectation ($\mathbb{E}^{Full}$)

For the full expectation, we assume that both the source text and latent variables are unknown, resulting in a sum over all the possible source bigrams $f_1 f_2$, and associated latent variables $e_1 e_2$:

$$\mathbb{E}^{Full} = \frac{1}{Z_g} \sum_{f_1 f_2 \in V_F^2} \sum_{e_1 e_2 \in V_E^2} \left[p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \right] \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

where Z_g is the global normalization term:

$$Z_g = \sum_{f_1 f_2 \in V_F^2} \sum_{e_1 e_2 \in V_E^2} p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

The computational complexity is $O(V^4)$.

5. MCMC Sampling for Faster Training

The overall computational complexity of estimating the exact gradient is $O(N_F V^2 + V^4)$, which is impractical even for a modest-sized vocabulary. We apply several MCMC sampling methods to approximate the forced and full expectations. We first propose using Gibbs sampling for both the forced and full expectation terms. We then propose a faster approximation using independent Metropolis Hastings sampling for just the forced expectation term. We then propose an even faster approximation using contrastive divergence for estimating both terms. We then extend this method to sample at the sentence level rather than at the bigram level, with the goal of increasing accuracy.

Computation times for the methods presented are summarized in Table 2.

5.1 Gibbs Sampling

5.1.1 Gibbs Sampling for Forced Expectation. Rather than summing over all target bigrams $e_1 e_2$, we approximate the forced expectation by taking N samples of $e_1 e_2$ for each

Table 2
The worst case computational complexities per iteration for different decipherment algorithms. Note that we observed that N_F tended to scale linearly with V.

Method	Complexity
EM	$O(N_F V^2)$
Feature HMM	$O(N_F V^2)$
Log-linear/MRF Exact	$O(N_F V^2 + V^4)$
Log-linear + Gibbs	$O(N_F V N + V N^2)$
Log-linear + IMH	$O(N_F N + V N^2)$
Log-linear + CD	$O(N_F N + V N^2)$
Log-linear + CD, Sentence	$O(\lvert\mathcal{F}\rvert N)$

observed $f_1 f_2$, and take an average of the features for these samples. For each observed $f_1 f_2$, the following steps are taken:

- Start with an initial target bigram $e_1 e_2$.
- Fix e_2 and sample e_1 according to the following probability distribution:

$$P(e_1|e_2, f_1 f_2) = \frac{1}{Z_{gibbs}} \left[p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \right]$$

  where Z_{gibbs} is the normalization term over all possible e_1 in the target vocabulary.
- Next, fix e_1 and draw a new sample e_2 similarly according to $P(e_2|e_1, f_1 f_2)$, and continue sampling e_1 and e_2 alternately until N samples are drawn.

Drawing each sample requires $O(V)$ operations, as we need to estimate the normalization term Z_{gibbs}. The computational complexity of estimating the forced expectation becomes $O(N_F V N)$, which is expensive as V can be large (and N_F generally scales with V).

5.1.2 Gibbs Sampling for Full Expectation. To efficiently estimate the full expectation, we sample N source bigrams $f_1 f_2$ from our model. The Gibbs sampling procedure is:

- Start with an initial random $f_1 f_2$.
- Fix f_2, and sample a new f_1 according to $p(f_1|f_2)$:

$$p(f_1|f_2) = \frac{1}{Z'_{gibbs}} \sum_{e_1} \sum_{e_2} \left[p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \right]$$

  where

$$Z'_{gibbs} = \sum_{f_1} \sum_{e_1} \sum_{e_2} \left[p(e_1 e_2) \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2) \right]$$

- Next fix f_1 and sample f_2 according to $P(f_2|f_1)$. Continue alternating until N samples are drawn.

The computational complexity of exactly estimating $p(f_1|f_2)$ is $O(V^3)$, resulting in the computational complexity $O(V^3 N)$, which is impractical for all but the smallest vocabularies. However, rather than summing over all possible $e_1 e_2$, we can approximate via sampling. For each $f_1 f_2$, we first sample N samples $e_1 e_2$ according to $p(e_1 e_2)$. Let S be the set of N samples of target bigrams. Next, we approximate $p(f_1|f_2)$ as

$$p(f_1|f_2) = \frac{1}{Z_{approx}} \sum_{e_1 e_2 \in S} \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$$

where $Z_{approx} = \sum_{f_1} \sum_{e_1 e_2 \in S} \exp \mathbf{w}^T \mathbf{\Phi}(f_1 f_2, e_1 e_2)$. This reduces the computational complexity to $O(VN^2)$.

5.2 Independent Metropolis Hastings (IMH)

In our experiments, the Gibbs sampling for our log-linear model was still somewhat slow, and will not scale well to larger experimental settings. To address this challenge, we apply IMH sampling, which relies on a proposal distribution and does not require normalization. However, finding an appropriate proposal distribution can sometimes be challenging, as it needs to be close to the true distribution for faster mixing and must be easy to sample from.

For the forced expectation, one possibility is to use the bigram language model $p(e_1e_2)$ as a proposal distribution. However, the bigram language model did not work well in practice. Because $p(e_1e_2)$ does not depend on f_1f_2, it resulted in slow mixing and exhibited a bias toward highly frequent target words.

Instead, we chose an approximation of $p(e_1e_2|f_1f_2)$ as our proposal distribution. To simplify sampling, we assume e_1 and e_2 to be independent of each other for any given f_1f_2. Therefore, the proposal distribution $q(e_1e_2|f_1f_2) = q_u(e_1|f_1)q_u(e_2|f_2)$, where $q_u(e|f)$ is a probability distribution over target unigrams for a given source unigram. We define $q_u(e|f)$ as follows:

$$q_u(e|f) = (1 - p_b)q_s(e|f) + p_b\frac{1}{V}$$

where p_b is a small back-off probability with which we fall back to the uniform distribution over target unigrams. The other term $q_s(e|f)$ is a distribution over the target words e for which the weight $w_{f,e}$ of the word pair feature $\phi_{f,e}$ is non-zero:

$$q_s(e|f) = \begin{cases} \frac{1}{Z_{imh}} \exp \mathbf{w}^T\phi(f,e), & \text{if } w_{f,e} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

Here, Z_{imh} is a normalization term over all the e such that $w_{f,e} \neq 0$. The weight vector $\mathbf{w}$ is sparse, as only a small number of translation features (f, e) (Section 7) are observed during sampling. Furthermore, we update q_s only once every five iterations of gradient descent.

The actual target distribution is:

$$p(e_1e_2|f_1f_2) \propto p(e_1e_2) \exp \mathbf{w}^T\mathbf{\Phi}(f_1f_2, e_1e_2) \tag{11}$$

For each $f_1f_2 \in \mathcal{F}$, we take the following steps during sampling:

- Start with an initial English bigram: $\langle e_1e_2\rangle^0$.
- Let the current sample be $\langle e_1e_2\rangle^i$. Next, sample $\langle e_1e_2\rangle^{i+1}$ from the proposal distribution $q(e_1e_2|f_1f_2)$.
- Accept the new sample with the probability:

$$P_a = \frac{p(\langle e_1e_2\rangle^{i+1}|f_1f_2)}{p(\langle e_1e_2\rangle^i|f_1f_2)} \frac{q(\langle e_1e_2\rangle^i|f_1f_2)}{q(\langle e_1e_2\rangle^{i+1}|f_1f_2)}$$

The IMH sampling reduces the complexity of the forced expectation estimation to $O(N_F N)$,[1] which is significantly less than the complexity of $O(N_F VN)$ in the case of Gibbs sampling. However, we could not apply IMH while estimating the full expectation, as finding a suitable proposal distribution is more complicated. Therefore, the overall complexity remains: $O(N_F N + VN^2)$.

5.3 Contrastive Divergence-Based Sampling

The main reason for the slow training of the proposed log-linear MRF model is the high computational cost of estimating the partition function Z_g when estimating the full expectation. A similar problem arises while training deep neural networks. An increasingly popular technique to address this issue is to perform contrastive divergence (Hinton 2002), which allows us to avoid estimating the partition function.

For each observed source bigram $f_1 f_2 \in \mathcal{F}$, contrastive divergence sampling works as follows:

- Sample a target bigram $e_1 e_2$ according to the distribution $p(e_1 e_2 | f_1 f_2)$. We perform this step using IMH, as discussed in the previous section.
- Sample a reconstructed source bigram $\langle f_1 f_2 \rangle^{recon}$ by sampling from the distribution $p(f_1 f_2 | e_1 e_2)$, again via IMH.

We take n such samples of $e_1 e_2$ and corresponding $\langle f_1 f_2 \rangle^{recon}$. For each sample and reconstruction pair, we update the weight vector by an approximation of the gradient:

$$\frac{\partial L}{\partial \mathbf{w}} \approx \mathbf{\Phi}(\langle f_1 f_2 \rangle^{data}, e_1 e_2) - \mathbf{\Phi}(\langle f_1 f_2 \rangle^{recon}, e_1 e_2)$$

5.4 Sentence-Level Sampling

Up to this point, we have considered parallel source/target bigram pairs in isolation, but it may be helpful to take larger contexts into account in decipherment. In this section, we extend the sampling procedures to resample an entire source/target sentence pair at each iteration. Although our features are functions of individual bigrams, sentence-level sampling gives us the benefit of looking at an individual word's left and right context when considering alternative translations. More generally, the HMM-like feature structure also allows information to flow through the entire sentence from beginning to end.

Mathematically, we assume, as we did in the bigram case, that our features can be written as a function $\phi(f, e)$ of a pair of French and English words. We use the notation

$$\mathbf{\Phi}(\mathbf{f}, \mathbf{e}) = \sum_{i=0}^{|\mathbf{f}|} \phi(f_i, e_i)$$

to denote the feature vector for an entire sentence pair; we will assume that the French and English sequences have the same length. Analogously to Equation (4), the gradient

1 Ignoring the cost of estimating $q_s(e|f)$, which occurs only once every five iterations.

of the log-likelihood can be written as the difference between a forced expectation and full expectation, now at the level of sentences rather than bigrams:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}} &= \mathbb{E}_{\mathbf{e}|\mathbf{f}}\left[\mathbf{\Phi}(\mathbf{f},\mathbf{e})\right] - \mathbb{E}_{\mathbf{e},\mathbf{f}}\left[\mathbf{\Phi}(\mathbf{f},\mathbf{e})\right] \\ &= \mathbb{E}^{Forced} - \mathbb{E}^{Full}\end{aligned}$$

We estimate these two terms with a sentence-level sampling algorithm based on contrastive divergence. At a high level, given an observed French sentence, it samples a hidden English sequence according to $p(\mathbf{e}|\mathbf{f})$ in order to estimate the forced expectation term of the update, and then samples a French sentence according to $p(\mathbf{f}|\mathbf{e})$ to estimate the full expectation, as shown in Algorithm 1. However, because the individual English words are not independent, due to the bigram language model, the sampling of $p(\mathbf{e}|\mathbf{f})$ is itself broken down into a sequence of Gibbs sampling steps, sampling one word at a time while holding the others fixed, as shown in Algorithm 2. This process is iterated to produce a total of N samples of the English sequence, with each sample initialized with the previous sample (line 4 of Algorithm 1). The entire process is initialized with a Viterbi decoding of the best English sequence under the current parameters (line 2 of Algorithm 1). Empirically, we found that this initialization sped up training by reducing the number of samples necessary.

Algorithm 1 Sentence-level contrastive divergence algorithm

1: **procedure** SENTCONTRASTIVEDIVERGENCE($\mathbf{f}$)
2: $\quad\mathbf{e}^{(0)} \leftarrow$ VITERBI($\mathbf{f}, \mathbf{w}$)
3: $\quad$**for** n in $1, \ldots, N$ **do**
4: $\quad\quad(\mathbf{e}^{(n)}, \mathbf{f}^{(n)})$ = SAMPLESENTENCEPAIR($\mathbf{e}^{(n-1)}, \mathbf{f}$)
5: $\quad\mathbf{w} \leftarrow \mathbf{w} + \frac{1}{N}\sum_n \left(\mathbf{\Phi}(\mathbf{f}, \mathbf{e}^{(n)}) - \mathbf{\Phi}(\mathbf{f}^{(n)}, \mathbf{e}^{(n)})\right)$

Algorithm 2 Sentence-level sampling algorithm

procedure SAMPLESENTENCEPAIR(**e**,**f**)
$\quad$**for** $i \leftarrow 1, \ldots, |\mathbf{f}|$ **do**
$\quad\quad e_i \sim \frac{1}{Z} p(e_{i-1}e_i)p(e_ie_{i+1}) \exp \mathbf{w}^T \phi(e_i, f_i)$
$\quad$**for** $i \leftarrow 1, \ldots, |\mathbf{f}|$ **do**
$\quad\quad f_i \sim \frac{1}{Z} \exp \mathbf{w}^T \phi(e_i, f_i)$
$\quad$**return** $(\mathbf{e}, \mathbf{f})$

6. Exploiting Parallel Data

We now turn to consider the setting in which a small amount of parallel data may be available for the two languages in question, along with a larger amount of monolingual data for each of the languages. Our hope is that even a small amount of parallel data may allow the model to learn the correspondence between very frequent words, such as function words. For many language pairs, including French–English, function words do not exhibit orthographic similarity, despite the high proportion of orthographically similar content words. Reducing errors among function words that are observed in the

parallel data may help prevent the decipherment model from aligning words that are spuriously similar "false friends" when analyzing the monolingual data.

Mathematically, we wish to define a single probability model that can apply to both parallel and monolingual data, and choose feature weights $\mathbf{w}$ that optimize the total likelihood of the parallel and monolingual data together. Probability models for word alignment of parallel data are one of the original problems studied in statistical machine translation (Brown et al. 1993). We wish to apply our log-linear feature-based model to parallel data, making the problem setting similar to that of Dyer et al. (2011). For simplicity, we assume a bag of words model that does not take into account the order of the words in the English sentence, resulting in a log-linear feature-based version of IBM Model 1.

We implement training for this model by modifying our sentence-level contrastive divergence method described in Section 5.4. We constrain the sampling of the English words $\mathbf{e}_{forced}$ used to approximate the $\mathbb{E}^{Forced}$ term by allowing only English words that appear in the English side of the parallel sentence pair. We sample a separate sequence of English words $\mathbf{e}$ for the $\mathbb{E}^{Full}$ term as before. The algorithm for parallel data is shown in Algorithm 3, where the English side of the parallel sentence pair is provided as an additional argument $\hat{\mathbf{e}}$. This set of words is used to constrain the choices of the Viterbi initialization of $\mathbf{e}_{forced}$ (line 2). The observed English sentence $\hat{\mathbf{e}}$ is also used to constrain the choice of sample in Algorithm 4; the indicator function $I(e_i \in \hat{\mathbf{e}})$ ensures that any English words not present in $\hat{\mathbf{e}}$ have zero probability of being sampled.

Algorithm 3 Constrained contrastive divergence algorithm

1: **procedure** CONSTRAINEDSENTCONTRASTIVEDIVERGENCE($\mathbf{f}, \hat{\mathbf{e}}$)
2: $\quad\mathbf{e}^{(0)}_{forced} \leftarrow$ CONSTRAINEDVITERBI($\mathbf{f}, \mathbf{w}, \hat{\mathbf{e}}$)
3: $\quad\mathbf{e}^{(0)} \leftarrow$ VITERBI($\mathbf{f}, \mathbf{w}$)
4: $\quad$**for** n in $1, \ldots, N$ **do**
5: $\quad\quad\mathbf{e}^{(n)}_{forced}$ = CONSTRAINEDSAMPLESENTENCE($\mathbf{e}^{(n-1)}_{forced}, \mathbf{f}, \hat{\mathbf{e}}$)
6: $\quad\quad(\mathbf{e}^{(n)}, \mathbf{f}^{(n)})$ = SAMPLESENTENCEPAIR($\mathbf{e}^{(n-1)}, \mathbf{f}$)
7: $\quad\mathbf{w} \leftarrow \mathbf{w} + \frac{1}{N}\sum_n \left(\mathbf{\Phi}(\mathbf{f}, \mathbf{e}^{(n)}_{forced}) - \mathbf{\Phi}(\mathbf{f}^{(n)}, \mathbf{e}^{(n)})\right)$

Algorithm 4 Constrained sentence-level sampling algorithm

1: **procedure** CONSTRAINEDSAMPLESENTENCE($\mathbf{e}, \mathbf{f}, \hat{\mathbf{e}}$)
2: $\quad$**for** $i \leftarrow 1, \ldots, |\mathbf{f}|$ **do**
3: $\quad\quad e_i \sim \frac{1}{Z} I(e_i \in \hat{\mathbf{e}}) p(e_{i-1}e_i) p(e_i e_{i+1}) \exp \mathbf{w}^T \phi(e_i, f_i)$
4: $\quad$**return** $\mathbf{e}$

Although our algorithm does not take into account the order of the observed English sentence $\hat{\mathbf{e}}$, we note that, unlike the training procedure for IBM Model 1, our algorithm does take advantage of the English bigram language model in constructing the alignment between the English and French sentences. Thus, whereas an English word pair is not more likely to align to adjacent French words if it is adjacent in the English sentence, it is more likely to align to adjacent French words if the English words are frequently adjacent in general; the motivation is that, as mentioned in Section 1, n-gram frequencies between the two languages are assumed to be similar. This is beneficial both because it provides the model with more information than is available

to IBM Model 1, and because it allows us to use a unified probability model for parallel and monolingual data.

7. Feature Design

We included the following unigram-level features:

- *Translation Features:* Each (f, e) word pair, where $f \in V_F$ and $e \in V_E$, is a potential feature in our model. Although there are $O(V^2)$ such possible features, we only include the ones that are observed during sampling. Therefore, our feature weights vector $\mathbf{w}$ is sparse, with most of the entries equal to zero.
- *Orthographic Features:* We incorporated an orthographic feature based on the normalized edit-distance between two words. The normalized edit distance between a word pair (f, e) is defined as follows:

$$NED(f, e) = \frac{ED(e, f)}{\max(|e|, |f|)}$$

  where $ED(e, f)$ is their string edit distance (minimum total number of required insertions, deletions, and substitutions) and $|e|$ and $|f|$ represent their lengths. When normalized edit distance between two words is larger than a threshold, it usually indicates that the words are orthographically dissimilar, and the exact value of the normalized edit distance does not carry much information. Based on this intuition, we chose our orthographic features to be boolean-valued features. For a word pair (f, e), the orthographic feature is triggered if the normalized edit distance $NED(f, e)$ is less than a threshold (set to 0.3 in our experiments).
- *Length Difference:* Because source words and their target translations often tend to have similar lengths, we added the absolute value of their length difference as a feature.

The set of features can further be extended by including context window–based features (Haghighi, Berg-Kirkpatrick, and Klein 2008; Ravi 2013) and topic model and word embedding features. Character rewriting features could be used to model when the two languages use different characters for the same sound; these could be coupled with the edit distance feature to approximate phonetic distance. Additionally, in this work we did not perform any character normalization; a simple extension of this system could treat similar characters (é, e, è) as identical for edit distance calculations.

8. Experiments and Results

8.1 Data Sets

We experimented with two closely related language pairs: (1) Spanish and English and (2) French and English. For Spanish–English, we experimented with a subset of the OPUS Subtitle corpus (Tiedemann 2009). For French–English, we used the Hansard corpus (Brown, Lai, and Mercer 1991), containing parallel French and English text from the proceedings of the Canadian Parliament. In order to have a non-parallel set-up,

Table 3
Statistics on the data sets used in our experiments.

Data set	Num. Sentences	$\|V_E\|$	$\|V_F\|$
OPUS	9.89*K* (997 unique)	375	530
Hansard-100	100	358	371
Hansard-1000	1,000	2,957	3,082

we extracted monolingual text from different sections of the French and English text. A detailed description of the two data sets is now provided.

OPUS Subtitle Data set: the OPUS data set is a smaller pre-processed subset of the original larger OPUS Spanish–English parallel corpora. The data set consists of short sentences in Spanish and English, each of which is a movie subtitle. The same data set has been used in several previous decipherment experiments (Ravi and Knight 2011; Nuhn and Ney 2014). We use the first 9,885 French sentences and the second 9,885 English sentences.

Hansard Data set: The Hansard data set contains parallel text from the Canadian Parliament Proceedings. We experimented with two data sets:

- **Hansard-100:** The French text consists of the first 100 sentences and the English text consists of the second 100 sentences.
- **Hansard-1000:** The French text consists of the first 1,000 sentences and the English text consists of the second 1,000 sentences.

Table 3 provides some statistics on the three data sets used in our experiments. The OPUS and Hansard-100 data sets have relatively smaller vocabularies, whereas the Hansard-1000 data set has a significantly larger vocabulary.

For each data set, we draw parallel data from a section that is disjointed from the monolingual sections. This data is only used in the "X% Parallel" settings, for which X% of the total data is drawn from the parallel section instead of the monolingual sections; for example, the "10% Parallel" setting for Hansard-1000 consists of 900 monolingual sentences in each language and 100 parallel sentence pairs.

8.2 Evaluation

We evaluate the accuracy of decipherment by the percentage of source words that are mapped to the correct target translation. We find the maximum-probability mapping for all source words; precision could be increased at the expense of recall by imposing some threshold, below which no mapping would be made for a given source word. The correct translation for each source word was determined automatically using the Google Translation API. Although the Google Translation API did a fair job of translating the French and Spanish words to English, it returned only a single target translation. We noticed occasional cases where the decipherment algorithm retrieved a correct translation, but it did not get credit because of not matching the translation from the API.

Additionally, we performed Viterbi decoding on the sentences in a small held-out test corpus from the OPUS data set, and compared the BLEU scores with the previously published results on the same test set as Ravi and Knight (2011) and Nuhn and Ney

(2014). Our training set, however, was different: Their data were parallel, so we split the data set into two disjoint sections, one for each language. This reduced our model's performance (as expected), but we still achieve a higher BLEU score than the baselines.

8.3 Results

We experimented with three versions of our log-linear MRF decipherment models: (1) Gibbs sampling, (2) IMH sampling, and (3) contrastive divergence (CD). We also tested the effect of exploiting parallel data under the CD model. To determine the impact of the orthographic and length features, the contrastive divergence–based log-linear model was tested both with and without these features. In addition to the proposed undirected MRF models, we also explored the directed Feature-HMM model (Berg-Kirkpatrick et al. 2010), which is trained via an EM-style algorithm, and has the same computational complexity as EM. We compared the feature-based models with the exact EM algorithm (Koehn and Knight 2000; Ravi and Knight 2011). We used Kneser-Ney smoothing (Kneser and Ney 1995) for training bigram language models. The number of iterations was fixed to 15 for all five methods; we did not see improvement beyond roughly 10 iterations during development. For the sampling based methods, we set the number of samples $N = 50$, which seemed to strike a good balance between accuracy and speed during our small-scale experiments during development.

For the log-linear model with no orthographic/length features, we initialized all the feature weights to zero. When we included the orthographic features, we initialized the weight of the orthographic match feature to 1.0 to encourage translation pairs with high orthographic similarity. Furthermore, for each word pair (f, e) with high orthographic similarity, we assigned a small positive weight (0.1). This initialization allowed the proposal distribution to sample orthographically similar target words for each source word. The value 0.1 seemed to work well in initial small-scale experiments. For exact EM, we initialized the translation probabilities uniformly and stored the entire probability table.

Table 4 reports the accuracy and running time per iteration for exact EM, Feature HMM, and our log-linear models on the OPUS, Hansard-100, and Hansard-1000 data sets. However, on the Hansard-1000 data set, we only applied the contrastive divergence and IMH based log-linear models because of its large vocabulary size. Table 2 summarizes the computational complexity of each method; recall that N_F scales with V (theoretically $N_F \in O(V^2)$, although empirically we found $N_F \in O(V)$). From this, we

Table 4
The running time per iteration and accuracy of decipherment.

Method	OPUS		Hansard-100		Hansard-1000	
	Time (sec)	Acc (%)	Time (sec)	Acc (%)	Time (sec)	Acc (%)
EM	417.2	2.63	188.0	2.96	–	–
Feature HMM	379.6	7.71	189.9	14.17	–	–
Log-linear + Gibbs	738.1	6.77	357.9	14.01	–	–
Log-linear + IMH	75.7	6.77	53.0	13.10	605.5	12.45
Log-linear + CD	19.1	6.13	10.6	11.53	324.1	11.19
Log-linear + CD, Sentence	21.6	8.08	12.7	12.60	458.3	12.02
Log-linear + CD, Sentence, No ortho/len	22.4	0.56	11.9	1.88	492.2	0.36

Table 5
The effect on accuracy of incorporating parallel data for our model (first three columns) and IBM Model 1 and Model 4 (on Hansard-1000 parallel data).

Accuracy (%)	Opus	Hansard-100	Hansard-1000	IBM Model 1	IBM Model 4
Monolingual-Only	8.08	12.60	12.02	0.00	0.00
10% Parallel	9.45	16.81	13.58	6.74	7.29
20% Parallel	10.81	17.62	13.41	9.67	10.75
50% Parallel	15.04	18.83	18.55	20.03	20.40

Table 6
Comparison of MT performance on the OPUS data set.

Method	BLEU (%)
EM (Ravi and Knight, 2011)	15.3
EM + Beam (Nuhn and Ney, 2014)	15.7
Feature HMM	18.90
Log-linear + Gibbs	21.43
Log-linear + IMH	21.46
Log-linear + CD	21.36
Log-linear + CD, Sentence	20.71
Log-linear + CD, Sentence, No ortho/len	19.36
Log-linear + CD, Sentence, 10% Parallel	20.83
Log-linear + CD, Sentence, 20% Parallel	21.18
Log-linear + CD, Sentence, 50% Parallel	29.30

can loosely estimate that the Gibbs sampling would take roughly a week to execute 15 iterations, whereas the EM and Feature HMM methods would take roughly a month.

Table 5 reports the accuracy for the methods that utilize parallel data on the three data sets. For comparison, the final two columns of Table 5 report the accuracy of IBM Model 1 and Model 4 (Brown et al. 1993) when trained on the parallel data used in the corresponding Hansard-1000 experiment; to allow for direct comparison, both were evaluated over the same vocabulary as in the Hansard-1000 experiment. Because our training procedure includes random sampling, the results of each run on a given data set can vary. We observe only very small variations between executions, but all reported results for sampling-based methods are the average of 10 separate executions of the system. A bigram language model was used for all the models.

The BLEU scores for translation on the OPUS data set are reported in Table 6. We outperform previous approaches on this data set that use no parallel data. Although we are not aware of any work on the OPUS data set using small amounts of parallel data, Zoph et al. (2016) describe one recent alternative approach to translation with very limited parallel data for Urdu–English. Their hybrid system using a string-to-tree statistical translation model combined with a neural model achieved a BLEU score of 19.1. This result utilized three times as much data as in our experiments; 100% of it was parallel, and the model was pre-trained with a much larger corpus of parallel French–English data.

Table 7 shows a few examples for which the log-linear model performed better because of orthographic features.

Table 7
A few examples for which orthographic features helped.

OPUS		Hansard-1000	
Spanish	*English*	*French*	*English*
excelente	excellent	criminel	criminal
minuto	minute	particulier	particular
silencio	silence	sociaux	social
perfecto	perfect	secteur	sector

9. Discussion and Future Work

We notice that all the feature-based models (both directed Feature-HMM and undirected log-linear models) with orthographic and length features outperformed the EM-based decipherment approach. The only log-linear model that performed much worse was the one which lacked the orthographic and length features. This result emphasizes the importance of orthographic features for decipherment between closely related language pairs. The margin of improvement due to orthographic features was bigger for the Hansard data sets than that for the OPUS data set. This is expected, as French has had a much larger historical influence on English than Spanish has, largely through the Norman Conquest; this is a major cause for the higher lexical similarity between French and English than between Spanish and English. Quantitatively, 42.72% of the pairs in our English–French gold dictionary were within the normalized edit distance threshold used for our corresponding feature, whereas only 20.97% of the English–Spanish pairs were. The contrastive divergence-based log-linear model achieved overall comparable accuracy to the two other sampling approaches (Gibbs and IMH + Gibbs), despite being orders of magnitude faster. Sentence-level sampling was slightly slower, but achieved higher accuracy than bigram-only sampling. Furthermore, the feature-based models resulted in better translations, as they obtained a higher BLEU score on the OPUS data set (Table 6).

Although the orthographic features provide huge improvements in decipherment accuracy, they also introduce new errors. For example, the Spanish word "madre" means "mother" in English, but our model gave the highest score to the English word "made" due to the high orthographic similarity. However, such error cases are rare compared with the improvement.

The contrastive divergence model that was modified to incorporate parallel data generally showed significant gains in accuracy as the proportion of parallel data was increased. This is expected: Parallel data provide a stronger signal for translation than monolingual data. We notice that, even when parallel data are provided, the model still learns additional information from the monolingual data. This is illustrated in Table 5, where we compare IBM Model 1 and Model 4 (which can only make use of parallel data) against our model and observe that more correct word translations are learned when additional monolingual data is provided. The exception to this trend is in the 50% Parallel setting, where the addition of monolingual data results in fewer correct translations. This may be because monolingual data is a noisy signal for translation, and incorporating too little with the parallel data actually confuses the model. Given that our technique is most applicable for low- and no-resource languages, for which having 50% parallel data is less realistic, we do not believe that this is a serious concern.

The accuracies reported here are significantly lower than those achieved by modern supervised methods (and unsupervised methods with large corpora). However, our results required no more than 1,000 lines of data from each language, and preserved accuracy with as little as 100 lines of data. Thus, this method in its current form is most applicable to languages with extremely limited available data. This can include "lost" languages and any of the numerous modern languages that do not have much data easily accessible online. Our model is also very scalable, and can be applied to settings with more data than we experiment with here but still insufficient data for modern embedding-based unsupervised methods.

For understudied languages, our system can also be used to *infer* the similarity of two languages. The final weight of the edit distance feature can be interpreted as the model's estimate of similarity. In our experiments, the edit distance weight for the Hansard experiments was roughly four times that of the OPUS experiments, which matches our expectations, given the increased lexical similarity between French and English. For future work, our feature-based models can be extended by allowing local reordering of neighboring words and considering word fertilities (Ravi 2013). We would also like to extend the features to handle languages with different alphabets or systematically different use of certain characters, perhaps using transliteration techniques such as in Knight and Graehl (1998). Finally, we would like to incorporate more flexible non-local features in MRF, which may not be supported by the directed Feature-HMM model.

10. Conclusion

We presented a feature-based decipherment system using latent variable log-linear models. The proposed models take advantage of the orthographic similarities between closely related languages, and outperform the existing EM-based models. The contrastive divergence–based variant with sentence-level sampling provided the best trade-off between speed and accuracy. We also showed that it can be modified to incorporate parallel data when available, resulting in increased accuracy.

Acknowledgments

We are grateful to the anonymous reviewers for suggesting useful additions. This research was supported by a Google Faculty award and NSF grant 1449828.

References

Ammar, Waleed, Chris Dyer, and Noah A. Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Advances in Neural Information Processing Systems (NIPS-27)*, pages 3311–3319, Montreal.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver.

Berg-Kirkpatrick, Taylor, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of the 2010 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-10)*, pages 582–590, Los Angeles, CA.

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

Brown, Peter F., Jennifer C. Lai, and Robert L. Mercer. 1991. Aligning sentences in

parallel corpora. In *Proceedings of the 29th Annual Conference of the Association for Computational Linguistics (ACL-91)*, pages 169–176, Berkeley, CA.

Church, Kenneth Ward. 1993. Char align: A program for aligning parallel texts at the character level. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Columbus, OH.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

Dou, Qing and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL-12)*, pages 266–275, Jeju Island.

Dou, Qing, Ashish Vaswani, and Kevin Knight. 2014. Beyond parallel data: Joint word alignment and decipherment improves machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, pages 557–565, Doha.

Dyer, Chris, Jonathan Clark, Alon Lavie, and Noah A. Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*, pages 409–419, Portland, OR.

Haghighi, Aria, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 771–779, Columbus, OH.

Haghighi, Aria and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the 2006 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-06)*, pages 320–327, New York, NY.

Hinton, Geoffrey. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Kneser, Reinhard and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 181–184, Detroit, MI.

Knight, Kevin and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Knight, Kevin and Kenji Yamada. 1999. A computational approach to deciphering unknown scripts. In *ACL Workshop on Unsupervised Learning in Natural Language Processing*, volume 1, pages 37–44, College Park, MD.

Koehn, Philipp 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 (AAAI-00)*, pages 711–715, Austin, TX.

Nuhn, Malte and Hermann Ney. 2014. EM decipherment for large vocabularies. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 759–764, Baltimore, MD.

Nuhn, Malte, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 1568–1576, Sofia.

Nuhn, Malte, Julian Schamper, and Hermann Ney. 2015. Unravel–a decipherment toolkit. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 549–553, Beijing.

Quattoni, Ariadna, Michael Collins, and Trevor Darrell. 2004. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems (NIPS-17)*, pages 1097–1104, Vancouver.

Rapp, Reinhard. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-93)*, pages 320–322, Cambridge, MA.

Ravi, Sujith. 2013. Scalable decipherment for machine translation via hash sampling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 362–371, Sofia.

Ravi, Sujith and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*, pages 12–21, Portland, OR.

Snyder, Benjamin, 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 (ACL-10)*, pages 1048–1057, Uppsala.

Tiedemann, Jörg. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Recent*

Advances in Natural Language Processing (RANLP), pages 237–248, Borovets.

Zhang, Meng, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver.

Zipf, George K. 1949. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.

Zoph, Barret, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 1568–1575, Austin, TX.