Masterarbeit im Fach Informatik
Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 6
Prof. Dr.-Ing. H. Ney

Unsupervised Learning of Neural Network Lexicon and Cross-lingual Word Embedding

20. September 2018

vorgelegt von:
Autor Jiahui Geng
Matrikelnummer 365655

Gutachter:
Prof. Dr.-Ing. H. Ney
Prof. B. Leibe, Ph. D.

Betreuer:
M.Sc. Yunsu Kim

Erklärung

Geng, Jiahui
Name, Vorname

365655
Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/~~Bachelorarbeit~~/
Masterarbeit* mit dem Titel

Unsupervised Learning of Neural Network Lexicon and Cross-lingual Word Embedding

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 20. September 2018
Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, 20. September 2018
Ort. Datum

Unterschrift

Abstract

Cross-lingual representations of words enable us to explore the distribution properties of different languages in a shared space. Based on semantic similarities reflected by cross-lingual word embeddings, we can implement bilingual dictionary induction, information retrieval and knowledge transfer between resource-rich and resource-lean languages.
In this thesis, I propose a novel training method of cross-lingual word embedding, which is distinct from other methods by its corpus-based instead of vocabulary-based training process. It integrates embedding mapping learning, lexicon induction and corpus translation. Language model (LM) is exploited make better use of contextual information to improve the corpus translation. The improvement of lexicon aids correspondingly cross-lingual word embedding. The performance of novel corpus-based approach is measured on a word retrieval. The experiments show that our method can achieve competitive results compared with the current supervised and unsupervised learning methods.
We further improve the unsupervised translation system with denoising neural network to handle reordering. The simple yet efficient translation system surpasses state-of-the-art unsupervised translation system without costly iteratively training. We verify the cross-lingual embedding on subword units performs poorly in translation and show that vocabulary cut-off helps for sentence translation. We also analyze the effect of different artificial noises to denoising model and propose a novel noise type.

Contents

Chapter 1

Introduction

Building a good machine translation system requires an enormous collection of parallel data. NMT systems often fail when the training data is not enough. However, parallel corpora are expensive because they require huge human labor, time and expertise of corresponding languages. Several approaches has been proposed to alleviate this issue, for instance, triangulation or semi-supervised learning techniques; these systems however still need a strong cross-lingual signal. Unsupervised machine translation uses only monolingual data of the source and target languages to train the model and in fact, monolingual corpora are readily available.
Recently, cross-lingual word embedding draws more and more attention since it helps to explore the multilingual semantic information. For instance, it enables computation of cross-lingual word similarities, which can be directly applied to bilingual dictionary induction or cross-lingual information retrieval. Several methods are proposed to learn the cross-lingual cross-lingual word embedding without any parallel data. This correspondingly inspires work on unsupervised machine translation.
In this thesis, I propose a novel corpus-based unsupervised training method, the experiments the method can achieve competitive results in comparison with the start-of-the-art methods. I further explore the application of the cross-lingual word embeddings in the machine translation task. I develop a fully unsupervised machine translation system starting from the simple word-by-word translation principle. LM is integrated to improve the translation. In order to boost the translation efficiency, I design a context-aware beam search to find the best translation candidate. To handle to reordering, I implement a denoising network with artificial noises, which are to mimic the true noises in the word-by-word translation, The results demonstrate that such simple but efficient translation system performs better even than most unsupervised neural machine translation system with costly iterative training.

1.1 Related Work

1.1.1 Word Embedding

Traditional language processing systems treat words as discrete atomic symbols, which assign for each word a specific ID number. Such encodings are arbitrary; it does not provide any information about the relations that may exist between individual symbols. In comparison, distributed representations of words in a high dimensional continuous space help learning algorithm to achieve better performance in natural language processing. According to the distributional hypothesis, similar words tend to occur with similar neighbors, and have similar word representations. At the beginning, in order to explore word similarity, word embedding training involves dense matrix multiplication and complex matrix analysis.
Mikolov et al. [2013a] propose two models for learning word embeddings with neural networks, namely skip-gram and continuous bag of words (CBOW) models. Neural architectures makes the training extremely efficient. In further work, Mikolov et al. [2013c] discuss about learning the embedding for phrases. Pennington et al. [2014] propose a regression model which combines the global matrix factorization and local context window methods, making the training process more interpretable. Bojanowski et al. [2016] represent each word as a bag of character n-grams, and assign for each character n-gram a distinct embedding. In this way, they can exploit subword information in word embedding training.
Mikolov et al. [2013b] again notice that continuous embedding spaces exhibit similar structures across languages, even when for distant language pairs, for example, English-Vietnamese. With separately learned monolingual embeddings they learn a linear mapping from source embedding space to target embedding space. To do this, they employ a parallel vocabulary of five thousand words as anchor points and evaluate their approach on a word translation task. Xing et al. [2015] show that results can be improved by enforcing an orthogonal constraint on the linear mapping.
Recently, Artetxe et al. [2017a] raise an iterative method that aligns the word embedding spaces gradually. In their work, the vocabulary size for learning can be reduced to 25 word pairs and even with Arabic numerals. The performance is comparable to the learning with large vocabulary dictionary **?** propose a distribution-based model, which is a modified CBOW model minimizing the distribution dissimilarity between source and target embeddings; here, distribution information refers the mean and variance. Zhang et al. [2017] put forward a method using adversarial training without any parallel data. The discriminator tries to distinguish if the embedding is from the source side, the generator aims to learn the mapping from source embedding space to the target one. These methods are fully unsupervised but the performances are much worse than the supervised training. Conneau et al.

[2017] simplify the adversarial training structure with different loss functions for the generator and discriminator, regularize the linear mapping to be orthogonal, the results get improved significantly. They also propose to use cross-domain similarity local scaling (CSLS) to handle hubness.

1.1.2 Unsupervised Machine Translation

As mentioned previously, the lack of parallel corpora motivates people to use monolingual data to improve the machine translation system. Some researchers use triangulation techniques (Cohn and Lapata [2007]) and semi-supervised approaches (Cheng et al. [2016]) to alleviate this issue. But these methods still require parallel corpora.
As to fully unsupervised machine translation, which only takes advantage of monolingual data, Ravi and Knight [2011] first consider it as a deciphering problem, where the source language is considered as ciphertext. They also propose iterative expectation-maximization (EM) method and Bayesian decipherment to train this unsupervised model. To solve the bottleneck of such model – mainly the huge memory required to store the candidates search space, – Nuhn et al. [2012] limit search candidates according to the word similarity. Nuhn and Ney [2014] limit the search space by using and preselection beam search. For the same reason, Kim et al. [2017] enforce the sparsity with a simple thresholding for the lexicon, also initializing the training with word classes to efficiently boosts the performance. Although initially not based on distributional semantics, recent studies show that the use of word embeddings can bring significant improvement in statistical decipherment Duong et al. [2016].
For neural machine translation, He et al. [2016] show a bidirectional, iterative way of effectively using monolingual data. More concretely, they train two agents to translate in opposite directions (e.g. French $\rightarrow$ English and English $\rightarrow$ French) and make them teach each other through a reinforcement learning process. While promising, this approach still requires a small parallel corpora for a warm start. Artetxe et al. [2017b] and Lample et al. [2017] propose two bi-directional unsupervised machine translation model which totally rely only on monolingual corpora in each language. Both models need to use cross-lingual word embedding to initialize the MT system and train the sequence-to-sequence system with denoising autoencoder. They turn the unsupervised training into a supervised one by introducing back-translation techniques. The most important principle for these two work is the shared representations. Sentences from different languages are encoded into a shared spaces and then translated into specific languages with different decoders.

1.2 Outline

The remainder of this thesis is structured as follows. In Section 1.3 we introduce the notations for this thesis. Chapter 2 introduces the development of machine translation systems from statistical models to neural models. Some basic techniques and principles for unsupervised machine translation are also explained. Chapter 3 gives a survey of training and applications details of cross-lingual word embedding. Chapter 4 discusses the unsupervised learning of cross-lingual word embedding and our efforts on data-driven training. We describe our unsupervised machine translation system in Chapter 5, which contains mainly context-aware search with the help of language model and denoising autoencoder method aimed at reordering. We demonstrate experimental results of cross-lingual word embedding model and unsupervised machine system in Chapter 6, comparing the performance with the state-of-the-art model. In Chapter 7, we summarize our work and draw conclusions.

1.3 Notation

In this thesis, we use the following notations:

- Source sentence: $f_1^J := f_1 \cdots f_j \cdots f_J$
- Target sentence: $e_1^I := e_1 \cdots , e_i \cdots e_I$
- Parallel sentences: (e_1^I, f_1^J)
- Probabilities of denoising autoencoders: $P_{f \to f}$, $P_{e \to e}$
- Translation models for both directions: $\Theta_{f \to e}$, $\Theta_{e \to f}$
- Probabilities of translation models for both directions: $P_{f \to e}$, $P_{e \to f}$
- Loss for denoising autoencoder: $\mathcal{L}^{\text{auto}}$
- Loss for back-translation: $\mathcal{L}^{\text{back}}$
- Single character for each language: f, e
- Source and target word embedding: $\boldsymbol{f}$, $\boldsymbol{e}$
- Seed bilingual dictionary: *dico*
- Corresponding embedding pair: $(\boldsymbol{f}, \boldsymbol{e})$
- Source and target embedding matrices: $\boldsymbol{F}$, $\boldsymbol{E}$
- Source and target Corpora: $\mathcal{F}, \mathcal{E}$

- Mapping default from source to target embedding space: W
- Mapping with specific direction: $W_{f \to e}$, $W_{e \to f}$
- Internal states of decoder: $\boldsymbol{s}$ and $\boldsymbol{s}_i$
- Internal states of encoder: $\boldsymbol{h}$ and $\boldsymbol{h}_i$
- Context vectors in attention mechanism: $\boldsymbol{c}$ and $\boldsymbol{c}_i$
- Cross-lingual regularization term $\Omega(\cdot)$
- Loss function for source and target embedding: $\mathcal{L}^f$, $\mathcal{L}^e$
- Temperature in inverted softmax: β
- General scaling parameter: λ
- Scaling parameters for lexicon and languagel model respectively: λ_{lex}, λ_{LM}
- Expectation: $\mathbb{E}$
- Dimension of embedding: d
- Frobenius norm: $\|\cdot\|_{\mathrm{F}}$
- Manifold of orthogonal matrices: $O_d(\mathbb{R})$
- Language model for source and target size: $\mathbf{LM}_f$, $\mathbf{LM}_e$
- Similarity matrices for source and target respectively: M_F,M_E
- Vocabulary size for most frequent words $|\tilde{V}|$

Chapter 2

Machine Translation

This chapter describes the classical or start-of-the-art machine translation (MT) models from statistical MT to neural MT. The related research work on unsupervised MT is also included.

2.1 Statistical Machine Translation

Statistical MT had achieved success in the early era of MT, until neural machine translation (NMT) emerged. The initial statistical models for MT are based on words as atomic units that may be translated, inserted, dropped and reordered. In statistical MT, we use both a translation model and a language model which encourages fluent output. Later, the statistical MT prefers to use translation of phrases as atomic units. These phrases are any contiguous sequences of words and not necessarily linguistic entities. In this approach, the input sentence is broken up into a sequence of phrases and are mapped one-to-one to output phrases, which may be reordered.

2.1.1 Word-Based Model

Noisy-channel model
Noisy-channel model is based on the notion of a noisy channel from Shannon's information theory, which had been applied to many language processing problems. Assuming that the source sentence is a distorted message omitted from the target sentence, we have a model on how the message is distorted (translation model $Pr(f_1^J|e_1^I)$) and also a model on which original messages are probable (language

model $Pr(e_1^I)$). The task is to find the best translation $\hat{e}_1^{\hat{I}}$ for an input foreign sentence.

$$\begin{aligned}\hat{e}_1^{\hat{I}} &= \underset{e_1^I}{\operatorname{argmax}}\{Pr(e_1^I|f_1^J)\} \\ &= \underset{e_1^I}{\operatorname{argmax}}\frac{Pr(f_1^J|e_1^I)Pr(e_1^I)}{Pr(f_1^J)} \\ &= \underset{e_1^I}{\operatorname{argmax}}\{Pr(f_1^J|e_1^I) \cdot Pr(e_1^I)\}\end{aligned}$$

IBM Models

Based on the noisy-channel model, IBM word-based translation makes the model more complicated by adding submodels. Starting from lexical translation, absolute alignment model, fertility etc. are added step by step.

The advances of the five IBM models are:

- IBM Model 1: lexical translation;
- IBM Model 2: adds absolute alignment model;
- IBM Model 3: adds fertility model;
- IBM Model 4: adds relative alignment model;
- IBM Model 5: fixes deficiency.

Alignment Model
This is an explicit model for reordering words in a sentence. More often than not, words that follow each other in one language have translations that follow each other in the output language. In detail, the position j in the source sentence is aligned with the position i in the target sentence when translating, denoted as $i = a_j$. Alignment model is a global reordering model. For whole sentence, we denote the alignment as:

$$a_1^J := a_1 \cdots a_J$$

There is not probabilistic model for alignment in IBM Model 1, it treats probability of all alignments equally. IBM Model 2 addresses the issue of alignment with an explicit model for alignment based on the positions of the input output words and sentence lengths $a(i|j, I, J)$.

Fertility
It models the specific number of output words in the output language. Generally one

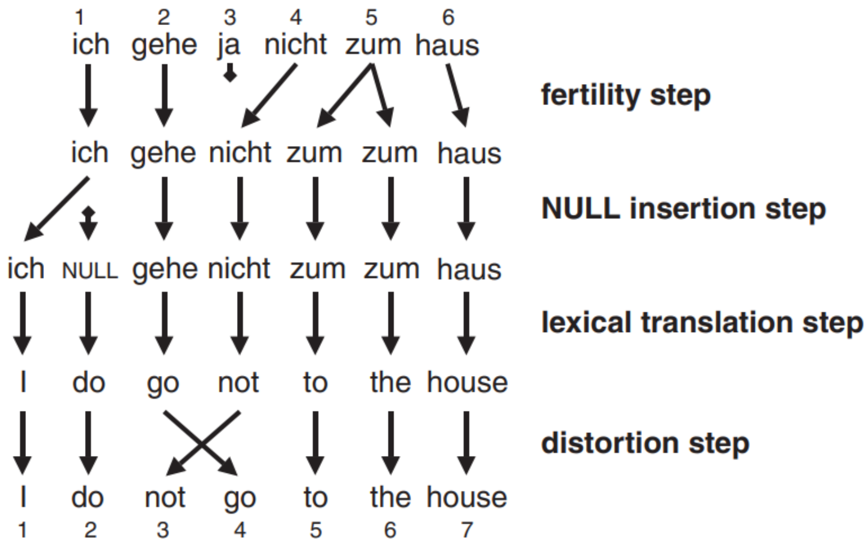

Figure 2.1: Illustration of a translation process using IBM-3 model (Koehn [2009])

word in one language is just translated into one single word in the other language. But some words produce multiple words or get dropped. We denote the fertility, as $\phi(e)$. As in all IBM models, it introduces the NULL token to account for source words that have no correspondent in target side. NULL insertion in added after the fertility step. The word-based translation process can be described as the Figure 2.1 Koehn [2009]

Weighted Model

In order to optimize the sub-models with appropriate weights, scaling exponents λ_m are introduced as in speech recognition. For example,

$$P(f_1^J, e_1^I; a_1^J) = P(J|I)^{\lambda_1} \cdot \prod_{i=1}^{I} P(e_i|e_1^{i-1})^{\lambda_2} \cdot \prod_{j=1}^{J} [P(a_j|a_{j-1}, I, J)^{\lambda_3} \cdot P(f_j|e_{a_j})^{\lambda_4}]$$

where the four probabilities corresponds to length model, language model, alignment model and lexical model respectively.

2.1.2 Phrase-Based Model

Actually when translating, words may not be the best candidates for the smallest units for translation; sometimes one word in a foreign language should be translated into two English words, or vice versa. Word-based models often break down

in these cases.
Phrase-based models typically do not strictly follow the noisy-channel approach proposed for word-based models, but use a log-linear framework to allow s straightforward integration of additional features.

Log-linear Model Combination
Consider arbitrary models (”feature functions”):

$$q_m(f_1^J, e_1^I; a_1^J) > 0 \quad m = 1, \cdots, M$$

$$\begin{aligned} Q(e_1^I, f_1^J; a_1^J) &= \prod_{m=1}^{M} q_m(f_1^J, e_1^I; a_1^J)^{\lambda_m} \\ &= \exp(\sum_{m=1}^{M} \lambda_m \log q_m(f_1^J, e_1^I; a_1^J)) \end{aligned}$$

In this frame work, we view each data point as a vector of features and the model as a set of corresponding feature functions, these functions are trained separately and combined assuming that they are independent of each other.

Components for log-linear model can be such as language model, phrase translation model, reordering model are used as feature functions with appropriate weights.

- Phrase translation model can be learned from a word-aligned parallel corpus, alternatively we also can use expectation maximization algorithm to directly find phrase alignment for sentence pairs.
- Reordering model for phrase case is typically modeled by a distance based reordering cost that discourage reordering in general. Lexicalized reordering model can be introduced for specific phrase pair.

The model can be further extended by components like: bidirectional translation probabilities, lexical weighting, word penalty and phrase penalty.

2.1.3 Decipherment

Ravi and Knight [2011] frame the MT problem as a decipherment task, treating the foreign text as a cipher. They also propose iterative EM method and Bayesian method to train this unsupervised model. Many concepts in deciperment are the same as the SMT.

IBM model tries to maximize the probability with hidden alignment model

$$\underset{\theta}{\operatorname{argmax}} \prod_{(e_1^I, f_1^J)} P_\theta(f_1^J | e_1^I) = \underset{\theta}{\operatorname{argmax}} \prod_{(e_1^I, f_1^J)} \sum_a P_\theta(f_1^J, a | e_1^I)$$

While for unsupervised case, we train

$$\underset{\theta}{\operatorname{argmax}} \prod_{f_1^J} P_\theta(f_1^J) = \underset{\theta}{\operatorname{argmax}} \prod_{f_1^J} \sum_{e_1^I} P(e_1^I) \cdot P_\theta(f_1^J | e_1^I)$$

for hidden alignments:

$$\underset{\theta}{\operatorname{argmax}} \prod_{f_1^I} \sum_{e_1^I} P(e_1^I) \cdot \sum_a P_\theta(f_1^J, a | e_1^I)$$

Since the model is very complicated, more assumptions are added to the model. The model accounts for word substitutions, insertions, deletions and local reordering during the translation process but does not incorporate fertilities or global re-ordering.

2.2 Neural Machine Translation

NMT has recently become the dominant method for MT task. In comparison to the traditional SMT, NMT systems are trained end-to-end, taking advantage of continuous representation of the hidden states that greatly alleviate the sparsity problem and make better use of more contextual information.Traditional phrase-based MT, which consists of several models that are tuned separately. neural MT can directly output translations given input, it contains only one single model. Also NMT models are trained with only one single training criterion.
Sutskever et al. [2014] first use a multi-layer Long Short-Term Memory (LSTM) to model a MT system. Attention mechanism has lately been used to improve neural MT by selectively focusing on parts of the source sentence during translation (Bahdanau et al. [2014], Luong et al. [2015a]). Here, inherently sequential nature precludes parallelization within training examples. Convolutional (Gehring et al. [2017]) and self-attentional (Vaswani et al. [2017]) architectures are brought forward for significantly more parallelization during training process in order to better exploit GPU hardwares and can reach a new state-of-the-art in translation quality.

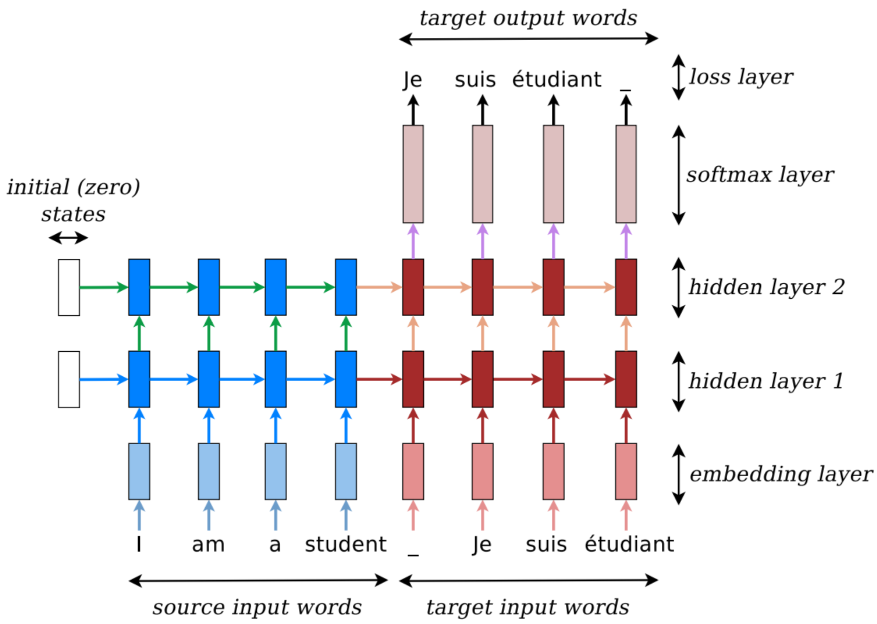

Figure 2.2: English→ French translation– example of a deep NMT (Luong et al. [2015a]).

2.2.1 Encoder-Decoder Framework

The most common NMT structure is the encoder-decoder framework, in which two neural networks work together to transform one sequence to another. An encoder condenses an input sequence into a vector $\boldsymbol{c}$, and a decoder unfolds that vector into a new sequence. The model can be expressed as:

$$p(e_1^I|f_1^J) = \prod_i p(e_i|e_0^{i-1}, f_1^J) = \prod_i p(e_i|e_0^{i-1}, \boldsymbol{c})$$

The decoder predicts the next word on the basis of the internal states and the previous target words. In more detail, one can parameterize the probability of decoding each word e_i as:

$$p(e_i|e_0^{i-1}, \boldsymbol{c}) = \mathrm{softmax}(g(\boldsymbol{s_i}))$$

with g being the transformation function that outputs a vocabulary-size vector. Here $\boldsymbol{s}_i$ is the decoder hidden state, updated as:

$$\boldsymbol{s}_i = f(\boldsymbol{s}_{i-1}, \boldsymbol{c})$$

It is common in neural MT systems to use beam search to sample the probabilities for the words in the sequence output by the model. The wider beam width leads to more exhaustive search but generally better translation quality.

There are obvious drawbacks in this model that will affect the performance of translation. The model compresses all information from the input sentence into a dense vector , while ignores the length of input sentence. When the length of input sentence get very long–even longer than the training sentences, it becomes harder to extract specific information for predicting target word for each different position. And it is not suitable to assign the same weight to all input words; one target word corresponds usually to one or several words in the input sentence. Treating all words equally does not distinguish the source information and influence the performance badly.

2.2.2 Attention Mechanism

Alignment is the problem in MT that identifies which parts of the input sequence are relevant to each word in the output, whereas translation is the process of using the relevant information to select the appropriate output. Bahdanau et al. [2014] introduce an extension to the encoder-decoder model which learns to alignment and translate jointly. Each time the model predicts the next target word, it softly searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.
Described in formula, attention mechanism derives a context vector $\boldsymbol{c}_i$ that capture the input information to help to predict the target word at the position i. Given the target hidden state $\boldsymbol{s}_i$ and the source-side context vector $\boldsymbol{c}_i$, we can compute the hidden state $\tilde{\boldsymbol{s}}_i$ by combining the current hidden state $\boldsymbol{s}_i$ and the context vector $\boldsymbol{c}_i$:

$$\tilde{\boldsymbol{s}}_i = tanh(W_c[\boldsymbol{c}_i; \boldsymbol{s}_i])$$

Then the target word is correspondingly predicted by softmax function:

$$p(e_i|e_0^{i-1}, f_1^J) = softmax(W_s\tilde{\boldsymbol{s}}_i)$$

Attention mechanism attends all the input words, weighted average of source hidden states (word representations):

$$\boldsymbol{c}_i = \sum_j \alpha_i(j) \cdot \boldsymbol{h}_j$$

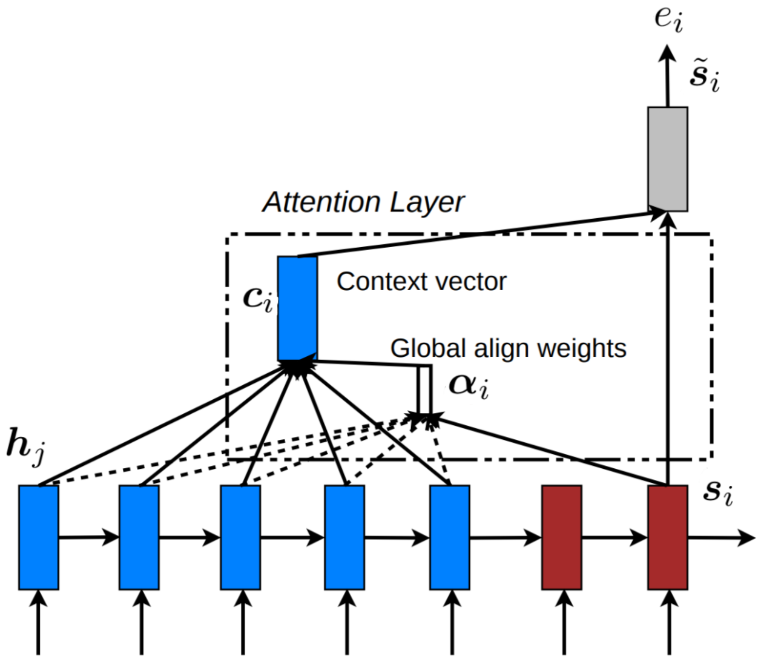

Figure 2.3: Global attention (Luong et al. [2015a])

where $\boldsymbol{\alpha}_i(j)$ is the normalized attention weight that output at position i is aligned with input position j, and it can be calculated as the following softmax-like function:

$$\alpha_i(j) = \text{align}(\boldsymbol{s}_i, \boldsymbol{h}_j) \tag{2.1}$$

$$= \frac{\exp\ (\text{score}(\boldsymbol{s}_i, \boldsymbol{h}_j))}{\sum_{\boldsymbol{j'}} \exp\ (\text{score}(\boldsymbol{s}_i, \boldsymbol{h}'_j))} \tag{2.2}$$

Since score function is referred as a content based function, different forms can be considered:

$$\text{score}(\boldsymbol{s_i}, \boldsymbol{h}_j) = \begin{cases} \boldsymbol{s_i}^T \boldsymbol{h}_j & \textit{dot} \\ \boldsymbol{s_i}^T W_a \boldsymbol{h}_j & \textit{general} \\ \boldsymbol{v_i}^T tanh(W_a[\boldsymbol{s_i}; \boldsymbol{h}_j]) & \textit{concat} \end{cases} \tag{2.3}$$

2.2.3 Transformer

The RNN encoder-decoder models have achieved state of the art in sequence modeling and MT problem. However such RNN models also have some disadvantages, because of their inherently sequential computation which prevents parallelization across elements of the input sequence. That means it is more difficult to fully take

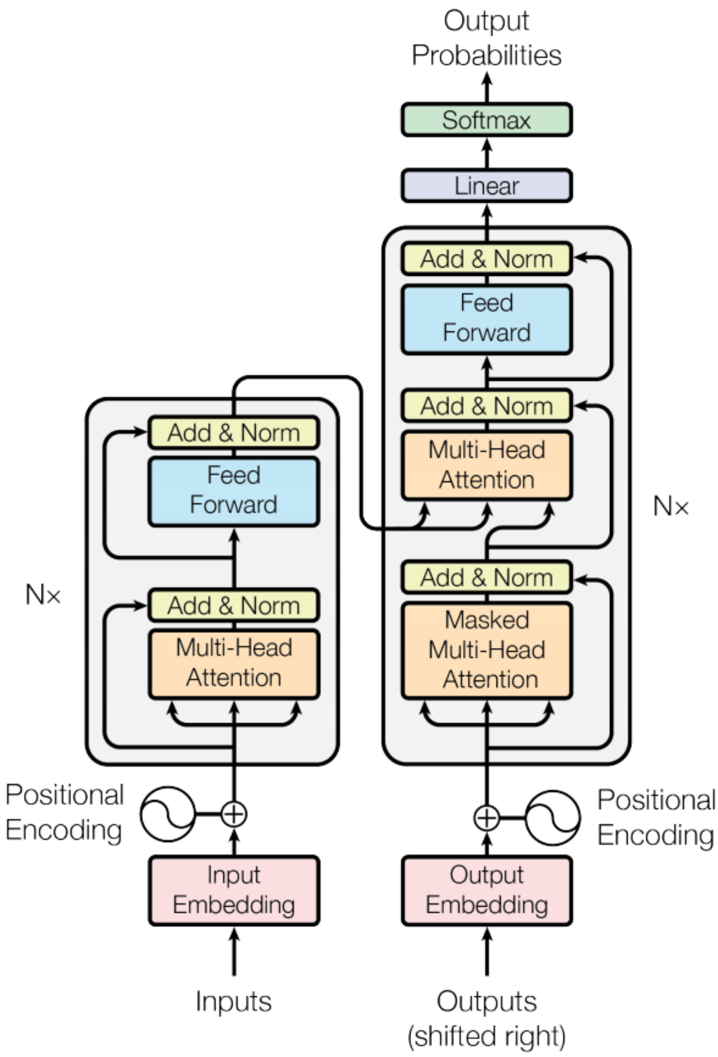

Figure 2.4: The Transformer model architecture (Vaswani et al. [2017])

advantage of the modern computing devices like GPU and TPU. Convolutional Gehring et al. [2017] and fully-attentional feed-forward architectures like Transformer Vaswani et al. [2017] moAnd please review your text again and find points to extend.dels are proposed as alternatives for RNNs.

Self-Attention
self-attention sublayers employs h attention heads. To form the sublayer output,

results from each head are concatenated and a parameterized linear transformation is applied.
Each attention head operates on an sequence of hidden states. For example in the encoder, given input $\boldsymbol{h} := \boldsymbol{h}_1, \cdots \boldsymbol{h}_J$ of J, attention head computes a new sequence $\boldsymbol{h}' := \boldsymbol{h}'_1, \cdots, \boldsymbol{h}'_J$ of the same length.
Each output element, $\boldsymbol{h}'_{j'}$ is computed as a weighted sum of a linearly transformed input elements:

$$\boldsymbol{h}'_{j'} = \sum_{j'=1}^{J} \alpha_{j'j} \cdot \boldsymbol{h}_j$$

Each weight coefficient $\alpha_{j'j}$ is computed using a softmax function:

$$\alpha_{j'j} = \frac{\exp r_{j'j}}{\sum_{\tilde{j}=1}^{J} \exp r_{j'\tilde{j}}}$$

And $r_{j'j}$ is computed using a compatibility function:

$$r_{j'j} = \frac{(\boldsymbol{h}_{j'})^\top (\boldsymbol{h}_j)}{\sqrt{d}}$$

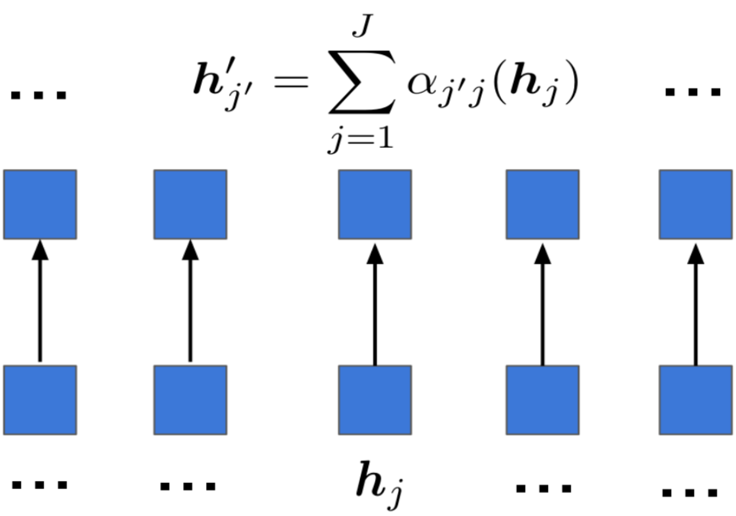

Figure 2.5: Self-attention structure

In the self-attention structure, the total computational complexity per layer will be greatly reduced, amount of computations can be paralellized. It performs better for long-range dependencies than traditional RNN structure.

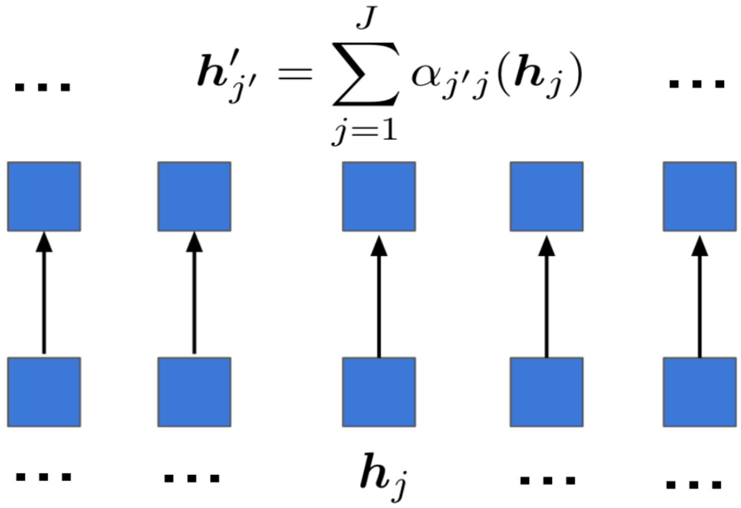

Figure 2.6: Self-attention structure

2.2.4 Unsupervised Neural Machine Translation

Although the NMT systems have achieved near human-level performance on some languages, the lack of large parallel corpora poses a major challenge for many low-resource languages. Techniques are raised to alleviate this issue with like triangulation and semi-supervised learning techniques. Monolingual data are leveraged to improve the quality of translation system.
Recently, several neural algorithms with fully unsupervised settings are proposed. The core idea is very similar to the dual learning in MT (He et al. [2016]). The systems all contain bidirectional models as sub-systems, where pseudo parallel data are created with back-translation for each iteration in both translation directions. It turns the unsupervised problem into a supervised translation setting. With better translation , both sub-systems get improved synchronously.
Though the dual unsupervised translation structure has achieved a relative good results, the efficiency of training such system are costly, especially because the back-translation element.

Dual Structure
Dual structure is inspired by the observation: any MT has a dual task: e.g. German-to-English translation (primal) and English-to-German translation (dual). Dual tasks can form a closed loop and generate informative feedback signals to for both translation models. Based on the feedback signals generated during this process, and leverage language model, we can minimize the reconstruction error of the original sentences. We can iteratively update the two models until convergence. He

et al. [2016] train two agents to translate in opposite directions and make them teach each other through a reinforcement learning process. While promising, this approach still requires a parallel corpus for a warm start. Lample et al. [2018] simplify this structure; the gradients will not be back-propagated through the reverse model, but only make use of the back-translation. The training process is depicted in Algorithm 1 Lample et al. [2018]

Algorithm 1 Unsupervised Machine Translation

Language models: $\mathbf{LM}_f$, $\mathbf{LM}_e$ over source and target languages;
Initial translation model: Leveraging $\mathbf{LM}_f$, $\mathbf{LM}_e$, lean two initial translation models in each direction: $\Theta^{(0)}_{f\to e}$, $\Theta^{(0)}_{e\to f}$;
for k = *1 to N* **do**
    Back-translation: Generate source and target sentences using the current translation models $\Theta^{(k-1)}_{f\to e}$ and $\Theta^{(k-1)}_{e\to f}$, leverading $\mathbf{LM}_f$, $\mathbf{LM}_e$;
    Retrain: Train new translation models $\Theta^{(k)}_{e\to f}$ and $\Theta^{(k)}_{f\to e}$ using the generated sentences and leverading $\mathbf{LM}_f$, $\mathbf{LM}_e$;
end

Initialization
Without enough information to start the dual MT system, it will be hard for the system to catch meaningful signals and then it will take much more iterations. One option is to initialize the system by a naive a word-by-word translation of the sentence, where the bilingual lexicon are derived from the same monolingual data. Though such initial word-by-word translation may be poor for languages or corpora that are not closely related, it still preserves some original semantics.

Shared Latent Representation
A shared encoder representation acts like an interlingua, which is translated into corresponding language regardless of the input source language. Since the supervision information only comes from the monolingual data, the model learn to translate by learning to reconstruct in both language from this shared space.
In order to share the encoder representations, we share all encoder and decoder parameters including the embedding matrices since we perform joint tokenization across the two languages. While sharing the encoder is critical to get the model to work, sharing the decoder simply induces useful regularization.

Optimization
When minimizing the loss function, gradients will not be back propagated through

the reverse model which generate the data. Instead the objective function minimized at every iteration is the sum of L^{auto} and L^{back}

- Denoising autoencoder loss

$$\mathcal{L}^{auto} = \mathbb{E}_{e_1^I \sim \mathcal{E}}[-\log P_{t \to t}(e_1^I | \text{noise}(e_1^I))] + \mathbb{E}_{f_1^J \sim \mathcal{F}}[-\log P_{f \to e}(f_1^J | \text{noise}(f_1^J))]$$

  where s $P_{f \to f}$ and $P_{e \to e}$ are the probability of encoder and decoder both operating in the source and target sides, respectively

- Back-translation loss

$$\mathcal{L}^{back} = \mathbb{E}_{e_1^I \sim \mathcal{E}}[-\log P_{f \to e}(e_1^I | u(e_1^I))] + \mathbb{E}_{f_1^J \sim \mathcal{F}}[-\log P_{e \to f}(f_1^J | v(f_1^J))]$$

  we denote the sentence that translated by intermediate target-to-source translation model as $u(e_1^I)$, similarly denote the sentence translated by source-to-target model as $v(f_1^J)$, so $u(y)$ should in source language and $v(x)$ should in target language. The pairs $(f_1^J, v(f_1^J))$, $(u(e_1^I), e_1^I)$ constitute synthetic parallel sentences.

Chapter 3

Word Embedding

Researchers have worked on a number of methods for using word embedding in MT tasks. It has been proven that word embedding helps to improve the translation quality and handle the Out-Of-Vocabulary (OOV) problems (Neishi et al. [2017], Qi et al. [2018]). Recently, cross-lingual word embedding is quickly gaining in popularity. They play a crucial role in transferring knowledge from one language to another. Cross-lingual word embeddings have been used either in standard machine translation system (Lample et al. [2017]) or as a method for learning translation lexicons in an entirely unsupervised manner (Xing et al. [2015], Lample et al. [2018]). Before we discuss the cross-lingual word embedding in this chapter, I will first introduce the learning algorithm for monolingual word embedding.Then I will explain the training and the lexical translation inference methods.

3.1 Monolingual Embedding

Word embedding is distributed representation of word in high dimensional space. It can capture the contextual or co-occurrence information, so that similar words have similar distribution in the embedding space. Several methods have been proven to be success like word2vec (Mikolov et al. [2013a]) GloVe (Pennington et al. [2014])

3.1.1 CBOW and Skip-gram

CBOW and skip-gram are current mainstream neural network model to learn word embedding. Algorithmically, CBOW tries to predict the current word based on the context while skip-gram model is to find better representations that are useful for predict Qi et al. [2018]ing the context words. Using skip-gram model as example, given a large training corpus represented as a sequence of words $e_1^N := e_1, \cdots e_I$, the objective of the model is to maximize the following log-likelihood:

$$\mathcal{L}^e_{\text{skip-gram}} = -\frac{1}{I}\sum_{i=1}^{I}\sum_{-c\leq d\leq c, d\neq 0} \log\ p(e_{i+d}|e_i)$$

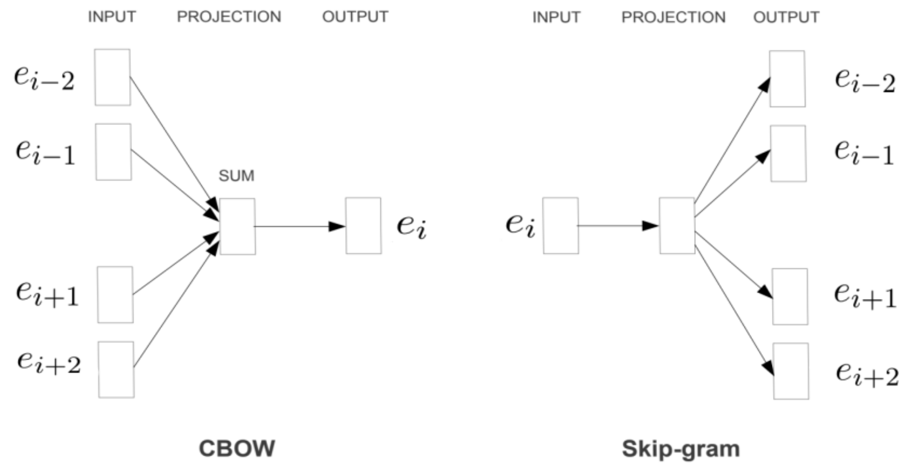

Figure 3.1: Global attention model (Mikolov et al. [2013a])

where e_i is the current word and c is the size of training context.

Similarly, we define the loss for CBOW model as

$$\mathcal{L}^{e}_{\text{CBOW}} = -\frac{1}{I}\sum_{i=1}^{I}\sum_{-c\leq d\leq c, d\neq 0} \log\ p(e_i|e_{i+d})$$

Negtive Sampling
The prediction probability $p(e_{i+d}|e_i)$ is defined as a softmax function:

$$p(e_{i+d}|e_i) = \frac{\exp\{\text{score}(e_{i+d}, e_i)\}}{\sum_{e'\in V_e}\exp\{\text{score}(e', e)\}}$$

where V_e is the entire vocabulary. The normalization over V_e is very expensive since it is calculate for all words at every training steo. Negative sampling is proposed to approximate the sofrmax to make it computationally more efficient. It trains to model to distinguish a target word from negative samples drawn from a noise distribution noise.

$$p(e_{i+d}|e_i) = \log p(e_{i+d} = 1|e_i) + \frac{1}{k}\sum_{e'\in p_{\text{noise}}} \log p(e' = 0|e_i)$$

where $p(\cdot|e_i)$ is a model to predict if the word is in the context of e_i.
Subword Information
The training methods mentioned treat each word as a distinct word embedding,

however, intuitively we can obtain more information from the morphological information of words. A subword model is proposed to capture the subword informaion. The training network is still CBOW or skip-gram. Each word is represented as a bag of character-level n-grams and assign a distinct embedding for each character n-gram. Score function in the network is calculated using the sum of inner products of the subword embeddings.

3.1.2 GloVe

Global vectors(GloVe) allows us to learn word representations via matrix factorization. GloVe minimizes the difference between the dot product of the embeddings of word and its context and the logarithm of their number of co-occurrences within a certain window size:

$$\mathcal{L}^{e}_{GloVe} = \sum_{i,j=1}^{|V_e|} f(C_{ij})(\boldsymbol{e}_i{}^{\top}\boldsymbol{e}_j - \log C_{ij})$$

where the matrix of word-word co-occurrence counts be denoted as C, whose entries C_{ij} tabulate the number of times word e_j occurs in the context of word e_j and $f(\cdot)$ is a weighting function that assigns relatively lower weight to rare and frequent co-occurrences. V_e is the source vocabulary size.

3.2 Cross-lingual Word Embedding

Mikolov et al. [2013b] observe that word embeddings trained separately on monolingual corpora exhibit isomorphic structure across languages, as illustrated in Figure 3.2. That means we can create a connection between source embedding and target embedding even with simple linear mapping. This has far-reaching implication on low-resource scenarios (Adams et al. [2017]), because word embedding training requires only plain text to train, which is the most abundant form of linguistic resource.

3.2.1 Supervised Training

Before we go on to the main training setting of this thesis (unsupervised), I will introduce the training methods under supervision. The supervision comes from seed lexicons, parallel data or document-aligned data.
Supervised training methods of cross-lingual word embedding can be divided into joint training approaches and mapping-based approaches. The difference is that joint methods train directly the cross-lingual word embeddings, while the mapping-based approaches first train monolingual word representations on corresponding

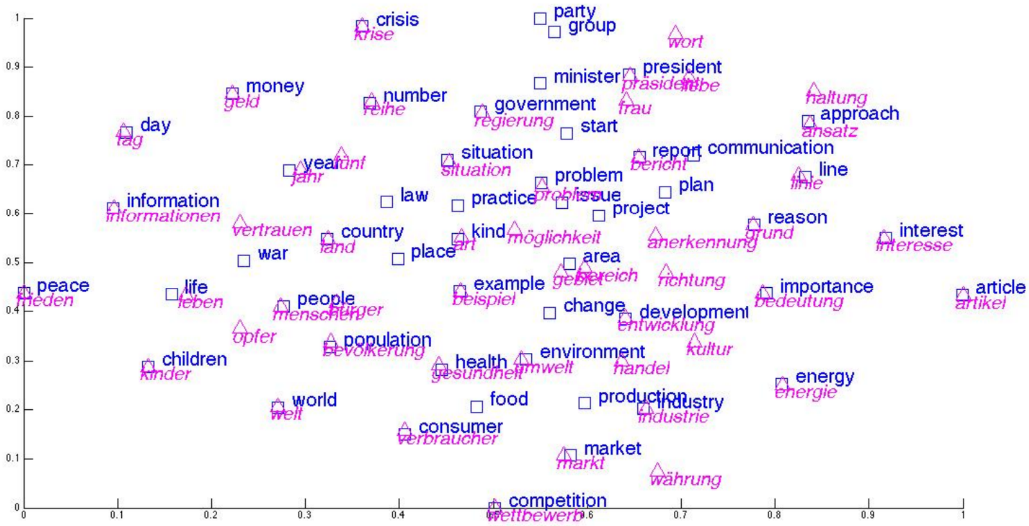

Figure 3.2: A cross-lingual embedding space between German and English (Ruder et al. [2017])

corpus separately. Then mapping will be learned to project different embeddings into a shared space according to supervision signals.

Joint Training Approaches

Here the joint training means to optimize source and target embedding objectives $\mathcal{L}(\cdot)$ together with the cross-lingual regularization term $\Omega(\cdot)$.

$$\mathcal{L} = \mathcal{L}^e_{\text{mono}} + \mathcal{L}^f_{\text{mono}} + \cdot\Omega(\cdot)$$

where the first two losses are exact the same as those in monolingual embedding training. $\Omega(\cdot)$ encourages representations to be similar for words that are related across different languages.

Different cross-lingual regularization terms are proposed to optimize the cross-lingual embeddings (Coulmance et al. [2016], Luong et al. [2015b], Gouws et al. [2015]). For example, Gouws et al. [2015] models it as:

$$\Omega_{\text{BilBOWA}} = \sum_{(f_1^J, e_1^I)} \|\frac{1}{I} \sum_{e_i \in e_1^I} \boldsymbol{e}_i - \frac{1}{J} \sum_{f_j \in f_1^J} \boldsymbol{f}_j\|^2$$

where $\boldsymbol{f}_j$ and $\boldsymbol{e}_i$ are the word embeddings for words in parallel source and target sentences (f_1^J, e_1^I). Instead of relying on expensive word alignments which minimize the distance that aligned to each other, they minimize the distance between the mean of the word representations in the aligned sentences.

Mapping-based Approaches
Another approach is to train monolingual word representations separately on monolingual corpora and then learn a transformation matrix mapping that maps representations in one language to the representations of other language. This approach can be further divided into learning separate mappings for each language or learning mapping only for source language to project source embeddings into target embedding space.
learn the transformation for each language, both monolingual embeddings are mapped into a shared embedding space. The typical model is CCA-based approaches (Faruqui and Dyer [2014], Dhillon et al. [2011] and Lu et al. [2015]). The other method is to learn the mapping from the source embedding space to the target embedding space. The criterion is to minimize the mean squared error between the transformed source embeddings and the target embeddings. The advantage of this method is that it is really fast to learn the embeddings and embedding alignments. People criticize whether linear or nonlinear can capture the relationship between all words in different languages.

- Mapping matrix for each language
  The typical method is canonical correlation analysis(CCA). let F and E be the embedding matrices comprised of seed word translation pairs. The W and V are the projection matrices which mapped the original embedding matrices into F^* and E^*, which are in the joint space, d is the embedding dimension.

  $$F^* = WF \qquad E^* = VE$$

  The objective of CCA is to find the two matrices such that the projections of the parallel matrices are maximally correlated:

  $$W^*, V^* = \underset{W,V \in \mathbb{R}^{d \times d}}{\operatorname{argmax}} \operatorname{corr}(WF, VE) \tag{3.1}$$

  $$= \underset{W,V \in \mathbb{R}^{d \times d}}{\operatorname{argmax}} \frac{\mathbb{E}[(WF)(VE)]}{\sqrt{\mathbb{E}[(WF)^2]}\sqrt{\mathbb{E}[(VE)^2]}} \tag{3.2}$$

  Once we get the two projection matrices, we can calculate the cross-lingual word embedding for the entire source and target vocabulary.

  A linear feature mapping is often not sufficiently powerful to faithfully capture the hidden, non-linear relations within the data. Lu et al. [2015] propose a non-linear extension of CCA using deep neural networks to maximize the correlation between the projections of both monolingual embedding spaces

  $$f_1^*, f_2^* = \underset{f_1, f_2}{\operatorname{argmax}} \operatorname{corr}(f_1(F), f_2(E))$$

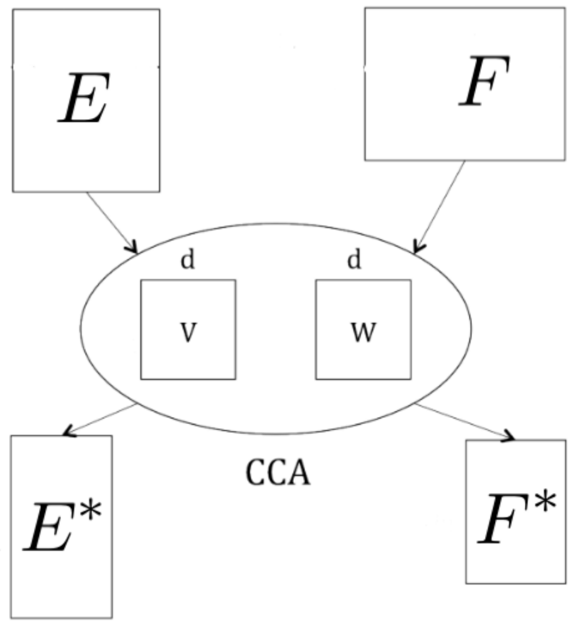

Figure 3.3: Cross-lingual projection with CCA (Faruqui and Dyer [2014])

where f_1 and f_2 are non-linear transformation implemented by neural networks.

- Mapping matrix for only source language Mikolov et al. [2013b] notice that the geometric relation that hold between words are similar across languages. This suggest that, we can learn the transformation directly from source embedding space to target space. Given a seed word translation pair set *dico*, the objective is to learn W that minimizes the distance between mapped source word embedding $W\boldsymbol{f}$ and corresponding target embedding $\boldsymbol{e}$:

$$\underset{W}{\operatorname{argmin}} \sum_{(\boldsymbol{f},\boldsymbol{e})\in dico} \|W\boldsymbol{f} - \boldsymbol{e}\|_2^2$$

This can also be expressed in matrix form as minimizing the squared Frobenius norm of the residual matrix F, E:

$$\underset{W}{\operatorname{argmin}} \|WF - E\|_{\mathrm{F}}^2$$

Xing et al. [2015] showed that the results are improved when we use l_2 normalized embeddings and constrain the W to be an orthogonal matrix. In this case we can use Procrustes analysis which advantageously offers a closed form solution obtained from the singular value decomposition

$$W^* = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WF - E\|_{\mathrm{F}}^2 = UV^T, \quad U\Sigma V^T = \mathrm{SVD}(EF^T)$$

where $O_d(\mathbb{R})$ denotes manifold of orthogonal matrices

3.2.2 Biligual Lexicon Induction

When we use nearest neighbors search to infer the word translation, we suffer from the so-called "hubness Problem": points are tending to be nearest neighbors of many points in high-dimensional space. Those points (hubs) will harm the search accuracy.
Now this limitation is addressed by apply a applying a corrective metric at inference time, such as inverted softmax (ISF)(Smith et al. [2017]) or cross-domain similarity scaling (CSLS)(Conneau et al. [2017])

- Inverted Softmax (ISF)
  The confidence of choosing a target word as translation of a source word can be considered as softmax-like normalized probability

  $$p(e|f) = \frac{\exp\Big(\beta \cdot \mathrm{score}(e, f)\Big)}{\sum_{e'} \exp\Big(\beta \cdot \mathrm{score}(e', f)\Big)}$$

  where the score$(\cdot)$ is the score function we can define ourselves. we learn the "temperature" β by maximizing the log probability over the training dictionary.

  $$\underset{\beta}{\operatorname{argmax}} \sum_{e,f} \ln p(e|f)$$

  The author observed that, if we invert the softmax and normalizing the probability over all the source words rather than target words, the hubness problem could be mitigated:

  $$p(e|f) = \frac{\exp\Big(\beta \cdot \mathrm{score}(e, f)\Big)}{\alpha \, \sum_{f'} \exp\Big(\beta \cdot \mathrm{score}(e, f')\Big)}$$

- Cross-domain Similarity Local Scaling (CSLS)
  We denote $\mathcal{N}_e(\boldsymbol{f})$ the set of k nearest neighbors of points of the mapped $\boldsymbol{f}$ in the target embeddiWe use the phrase embedding also in the machine translationng space, and $\mathcal{N}_f(\boldsymbol{e})$ the nearest neighbors of mapped $\boldsymbol{e}$ in the source embedding space. We consider the mean cosine similarity as hub-ness:

$$r(\boldsymbol{f}) = \frac{1}{K} \sum_{\boldsymbol{e}' \in \mathcal{N}_e(\boldsymbol{f})} \cos(\boldsymbol{e}', W\boldsymbol{f})$$

  So in this way we penalize the hub points:

$$CSLS(\boldsymbol{e}, \boldsymbol{f}) = 2\cos(\boldsymbol{e}, \boldsymbol{f}) - \frac{1}{k} \sum_{\boldsymbol{e}' \in \mathcal{N}_e(\boldsymbol{f})} \cos(\boldsymbol{f}, \boldsymbol{e}') - \frac{1}{k} \sum_{\boldsymbol{f}' \in \mathcal{N}_f(\boldsymbol{e})} cos(\boldsymbol{f}', \boldsymbol{e})$$

3.3 Phrase Embedding

Different methods are proposed to learn phrase embedding, which can be mainly divided into two categories. The first is based on the distributional hypothesis; it treats a phrase as fundamental unit as that in phrase-based MT. Phrases will be considered as non-divisible and phrase embedding algorithm is similar to word embedding. However, distributional methods cannot make use of the information embedded in component words and they also face data spareness problem. The second is based on the principle of compositionality. It infers phrase embedding according to the embeddings of its component words. However, many phrases have a meaning that is not simple composition of the meaning of its individual words.
In this paper, we follow first idea, we train the phrase embeddings using the data-driven method proposed by Mikolov et al. [2013c]. We also propose a novel phrase detection method. We find that our method is easy to implement and to tune the parameter. In experiments, our method works obviously better than the other one with the same phrase skip-gram training.

Phrase Detection

- Phrases formed based on the unigram and bigram counts (Mikolov et al. [2013c])

$$\text{score}(e', e) = \frac{\text{count}(e', e) - \delta}{\text{count}(e') * \text{count}(e)}$$

  where δ is a discounting coefficient and prevents too many phrases consisting of very infrequent words to be formed. The bigrams with score above the chosen threshold are then used as phrases.

- Process sentences with most common phrases in training corpus
  We count the most frequent bigrams in training corpus, using the counts directly as phrase score: $\text{score}(e', e) = \text{count}(e', e)$. We maximize the sum of phrase scores in given sentences in order to detect the phrases with dynamic programming. In our approach we consider the top-k as the only parameter.

With the same training algorithm as word embedding, we can obtain the phrase embeddings. We use the phrase embeddings also in the machine translation system, which will be explained in Chapter 5, and find phrase embeddings performs better than word embedding iunder some specific settings. Experiment results are demonstrated in Chapter 6.4.4.

Chapter 4

Unsupervised Learning of Cross-lingual Word Embedding

Chapter 3 introduces the supervised learning of cross-lingual word embedding, they employ parallel data like bilingual dictionary or parallel corpora and have shown strong results on a word retrieval task. However, labeling the lexicon information still costs lots of labor. This motivates people to explore fully unsupervised method. In this chapter, several unsupervised approaches will be explained including a novel corpus-based approach, which is distinct from other methods by exploiting an extra language model and corpus-based training process. All these approaches contains an iterative training procedure.

4.1 Initialization

Pre-trained monolingual embeddings capture distribution information for each language. These embeddings are typically not expected to be aligned between languages A good initialization is a key to the iterative training framework. A practical approach for reducing the need of bilingual supervision is to design heuristics to build the seed dictionary. Hauer et al. [2017] propose to use shared words and cognates to extract the seed dictionary. This method has a flaw that it depends on the writing system. Artetxe et al. [2018] raise a method that using similarity matrix to capture rough cross-lingual signals.

4.1.1 Heuristics

The seed is constructed by identifying words with similar spelling (cognates). A relative-frequency constraint is imposed to filter out word pairs that have fully different meaning; generate the list of words by frequency first, for each source word, if there exists target word that the normalized edit distance (NED) and absolute difference between the respective frequency ranks is within a specific threshold, these two word will be added into a seed dictionary. This noisy seed lexicon will be considered as initial signals. There is no one-to-one constraint, so both source and target words may appear multiple times in the seed.

4.1.2 Similarity Matrix

Similarity matrix is based on a simple assumption: while the axes of the original $|\tilde{V}|$ most frequent embeddings $F \in \mathbb{R}^{d\times|\tilde{V}|}$ and $E \in \mathbb{R}^{d\times|\tilde{V}|}$ are different, but $M_F = F^\top F$, $M_E = E^\top E$ capture the similarity inner respective languages. The similarity matrices M_F and M_E would be equivalent up to a permutation of their rows and columns. For example, in one language, the similarity between a, b, c and i, j, k is as the left graph in Figure 4.1, suppose ideal the their corresponding translations are a', b', c' and i', j', k', but the order differs as the middle graph. If we first sort the values in each row, it will be much easier to find the relationship between a and a', etc.. They find $\sqrt{M_F}$ and $\sqrt{M_E}$ work better in practice.

	i	j	k
a	1	2	3
b	4	5	6
c	7	8	9

	k'	j'	i'
b'	6	5	4
c'	9	8	7
a'	3	2	1

b'	4	5	6
c'	7	8	9
a'	1	2	3

Figure 4.1: Illustration of similarity matrix method

While the latter is far from the accurate alignment, it works well as the initialization for the iterative methods described next.

4.2 Vocabulary-based Learning

The vocabulary-based approach depends on high-quality lexical information to learn the mapping between embedding distributions. We can use Figure 4.2 to illustrate the vocabulary-based learning process. As in graph (A) there are two distributions of word embeddings, source words in red denoted by F and target words in bleu denoted by E, which we want to align. Each dot represents a word in shared space. Using initialization, we learn a rotate matrix W which roughly aligns the two distributions. According to dictionary induction, we can find some reliable points in graph (B). We can futher refine the mapping W, using frequent words aligned by the previous step as anchor points, and minimizes a loss function measured by mapping distance as in graph (C). Repeat steps in (B) and (C), we hope to find a liner mapping that can be generalized to all words in the dictionary as in graph (D).

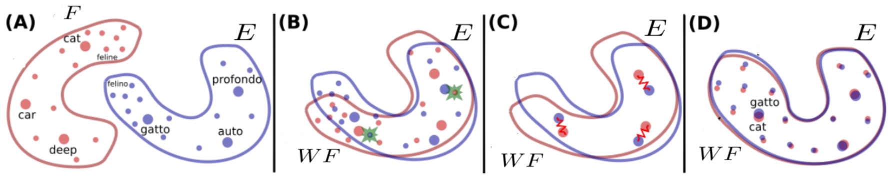

Figure 4.2: Illustration of vocabulary-based learning (Conneau et al. [2017])

4.2.1 Iterative Procrustes Analysis

After initialization, the words in the seed dictionary can be considered as temporary anchor points for Procrustes analysis, which has been introduced in Section 3.2.1. Apply the Procrustes solution on the current seed dictionary, use the learned mapping matrix W and dictionary induction technique to construct a better seed dictionary again. Repeat such process we suppose to obtain a hopefully better mapping and dictionary until some convergence criterion is met. The training will be stopped when the improvement on the average dot product for the induced dictionary falls below a give threshold from one iteration to the next. In practice, the threshold value is set at $1e-6$. To prevent the mapping and dictionary from getting stuck in local optima, Mutual nearest neighbors and thresholding are used to ensure a high-quality dictionary. In the experiments we find that intersection of two unidirectional dictionaries work better than union or a unidirectional dictionary, that is to say, the quality of dictionary plays more important role than the quantity.

Algorithm 2 Iterative training procedure

Input: F (source embeddings)
Input: E (target embeddings)
Input: *dico* (seed dictionary)
Result: W (embedding mapping)
while *not converge* **do**
  $W \leftarrow LEARN\_MAPPING(F, E, dico)$
  $dico \leftarrow LEARN\_DICTIONARY(W)$
end

4.2.2 Adversarial Training (GANs)

The adversarial training was for cross-lingual embedding mapping is first proposed by Barone [2016], who combines an encoder that maps source word embeddings into the target embedding space, a decoder that reconstructs the source embeddings from the mapped embeddings and a discriminator that discriminates between embeddings. Zhang et al. [2017] incorporate additional techniques like noise injections to aid the training. Conneau et al. [2017] finally achieve good results by dropping the reconstruction component regularizing the mapping to be orthogonal
Let $= \{\boldsymbol{f}_1, \cdots, \boldsymbol{f}_{|\tilde{V}|}\}$ and $= \{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{|\tilde{V}|}\}$ be the most $|\tilde{V}|$ frequent word embeddings from source and target languages respectively. We refer the discriminator parameters as θ_D.The discriminator is a multi-layer neural network trained to discriminate the transformed source word embedding from the target word embedding, while the mapping W, as simple as a linear transformation, is trained to fooling discriminator. In the two-player game, mapping from source embedding space to the target space is supposed to be learned.

Generator loss

$$\mathcal{L}_{\mathcal{G}}(W|\theta_D) = -\frac{1}{|\tilde{V}|}\sum_{n=1}^{\tilde{V}} \log P_{\theta_D}(source = 0|W\boldsymbol{f}_n) - \frac{1}{\tilde{V}}\sum_{n=1}^{|\tilde{V}|} \log P_{\theta_D}(source = 1|\boldsymbol{e}_n)$$

Discriminator loss

$$\mathcal{L}_{\mathcal{D}}(\theta_D|W) = -\frac{1}{|\tilde{V}|}\sum_{n=1}^{|\tilde{V}|} \log P_{\theta_D}(source = 1|W\boldsymbol{f}_n) - \frac{1}{|\tilde{V}|}\sum_{n=1}^{|\tilde{V}|} \log P_{\theta_D}(source = 0|\boldsymbol{e}_n)$$

4.3 Corpus-based Learning

4.3.1 Motivation

The vocabulary-based training approach makes good use of the embedding distribution similarity. However, it just looks at the mutual nearest neighbors, not considering polysemy of the words. It does not take the semantic or contextual information into consideration when training the cross-lingual mapping, which would be a flaw especially in machine translation task. We also argue that word frequency should also be considered when training. For example, though words are synonyms, some of them are more used than others. When inducing the lexicon, we should assign priority to frequent words.
In if we translate corpus, we can consider those various translations to make the mapping better The intuition of corpus-based approach is that, if we integrate

the translation task into the cross-lingual mapping learning process, the factors mentioned above will be taken into account; with information provided from LM, different translation candidates will be chosen. Also LM prefers the common words.

4.3.2 Framework

We propose a novel corpus-based learning approach by integrating corpus translation task. We build a word-based machine translation, in this thesis simplified as a word-by-word translation system, which exploits dictionary induction as lexical model. The intuition is that if the translation system gets improved better translation, we can further learn a better mapping based on the word translation pairs. LM is used to improve the translation. In the closed-loop learning process by alternating corpus translation and embedding mapping learning, we are hopefully learn good mapping matrix at last. Algorithm 3 summarize such iterative framework I propose. We use neural network with only one layer to represent the mapping matrix W, train successively with stochastic gradient updates to minimize the distance measurement.

Algorithm 3 Iterative learning of corpus-based approach

Input: F (source embeddings)
Input: E (target embeddings)
Input: $\mathbf{LM}_e$ (language model)
Input: $\mathcal{F}$ (source corpus)
Result: W (embedding mapping)
while *not converge* **do**
  $\mathcal{E} \leftarrow TRANSLATE(\mathcal{F}, F, E, \mathbf{LM}_e)$
  $W \leftarrow LEARN\_MAPPING(\mathcal{F}, \mathcal{E})$
end

This corpus-based approach combines mapping learning, dictionary induction and corpus translation. The efficiency turns out to be critical. Actually, the learning time depends mainly on the corpus size. It will be very costly if we translate the entire source corpus especially at the beginning stage, when the improvement of the mapping is relative slow and more training iterations are required. In order to solve this, I propose an online learning method, which can greatly improve the training efficiency.

4.3.3 Online Training

Algorithm 4 Online learning for corpus-based approach

Input: F (source embeddings)
Input: E (target embeddings)
Input: $\mathbf{LM}_e$ (language model)
Input: $\mathcal{F}$ (source corpus)
Result: W (embedding mapping)
while *not converge* **do**
    Generate batch of source sentences $\{f_1^J\}$ from $\mathcal{F}$
    $\{e_1^I\} \leftarrow TRANSLATE(\{f_1^J\}, F, E, \mathbf{LM}_e)$
    $W \leftarrow LEARN\_MAPPING(\{f_1^J\}, \{e_1^I\})$
end

Instead of translate the entire corpus, each time I iterate over the corpus to generate specific number of sentences as a sentence batch. After translation is done, we extract the word translation pairs as seed dictionary. For more complicated word-based machine translation system, extraction will requires the alignment information. In my word-by-word translation system, I just use the word translation pairs at each position. We could improve the quality of seed dictionary by filtering the pairs, which could be noise in the dictionary according to statistical information. We train the mapping using neural network trained with SGD. Mapping learned on current sentence batch is used to induce the lexicon for translation of next sentence batch. The efficiency of this online algorithm mainly depends on the corpus size and batch size. We can either extend the corpus size or train over the same corpus more than once since this approach is data-driven method.

4.3.4 Training Details

- Embedding Normalization
  We start with a pre-processing that normalize the embeddings, mean the centers of each dimension and then normalize again. We take three steps in order to obtain normalized embeddings with unit length and centering near zero. As in both CBOW and skip-gram model, the prediction probability is calculated with the inner product of embeddings. By normalization, the inner product falls back to the cosine distance, and cosine distance the criterion we used for nearest neighbors search. In addition to this, the orthogonal constraint preserves the length normalization and cosine similarity after the projection. Hence normalization helps to keep the consistence between embedding learn-

ing and dictionary induction (Xing et al. [2015]). After normalization, all embeddings are located on a hypersphere. Dimension-wise mean centering ensures that the expected inner product of two random embeddings consequently, the cosine similarity, is zero (Artetxe et al. [2016]).

- Dictionary Induction
  - CSLS retrieval With cross-lingual word embeddings, we can directly find the word translation using nearest neighbors search. The nearest neighbor suffers from the hubness problem. As mentioned in Section 3.4, we adopt the Cross-domain Similarity Local Scaling (CSLS) from Conneau et al. [2017]. Following the authors, we set $k = 10$.
  - Frequency-based vocabulary cutoff
    The complexity for dictionary induction grows with respect to target vocabulary size. This increases not only computation cost, but also the number of possible translations. In the experiments we find out that those less frequent words can be noise since less words are less trained. These noise word translations would have global negative influence of the sentence translation. We propose to restrict the vocabulary size to the k-top frequent words in the source and target languages, where we find $k = 50000$ can balance the efficiency and accuracy.

- Corpus Translation
  Most words have multiple translations, some are more likely than others. Also the translation depends on the context. For example, 'home', 'house' are synonyms, but when translating 'zu Haus', we will choose 'home'. By integrating LM, we can choose better word translation as well as sentence translation according to contextual information. We exploit a 5-gram LM trained by KenLM (Heafield [2011]) and use a linear mapping to convert the cosine similarity to lexical probability. I improve the translation efficiency greatly by implementing a context-aware beam search with beam size 10.

$$\hat{e}_1^J = \operatorname*{argmax}_{e_1^J} \prod_{i=1}^{J} P^{\lambda_{LM}}(e_i|e_{i-4}^{i-1}) \cdot Q^{\lambda_{lex}}(f_i, e_i)$$

  where $P(\cdot)$ and $Q(\cdot)$ are scores predicted respectively by LM and cross-lingual word embeddings. Both scores are in range $[0, 1]$. More details will be explained in Chapter 5.

- Optimization Objective
  Suppose we have got translation, we can extract word translation pairs as seed dictionary *dico*. The goal of optimization is to find the best mapping so that

the mapping is more general to cover as much as possible lexicon with high accuracy. In this theis, linear mapping $W \in \mathbb{R}^{d \times d}$ will be trained to minimize a self-defined loss function between mapped source word embeddings target word embeddings.

$$W = \underset{W \in \mathbb{R}^{d \times d}}{\operatorname{argmin}} \frac{1}{|\tilde{V}|} \sum_{n=1}^{|\tilde{V}|} l(W \boldsymbol{f}_n, \boldsymbol{e}_n)$$

where l is a loss function, in this thesis I compare the results of Euclidean distance loss and cosine similarity loss and find the difference is small.
To train our model, we follow the standard training procedure of neural network. For every input sample, we minimize the loss function with stochastic gradient decent (SGD) method. Better optimizer could be used to prevent from getting stuck into local optima.

- Orthogonal Constraint
  As mentioned previously, the orthogonal preserves the quality of monolingual embeddings also dot product of vectors and l_2 distance. In our experiments, we find the orthogonal constraint make training procedure converge successfully. In this thesis, the orthogonal constraint is added during the training process; we use projected gradient descent to train the neural network by alternating the training and the orthogonal constraining:

  $$W \leftarrow (1 + \beta)W - \beta(WW^{\top})W$$

  where $\beta = 0.01$ is usually found to perform well. This method ensures that the matrix stays close to the manifold of orthogonal matrices after each update.

- Learning Rate Scheduling
  Two different learning rate schedulers are used in this thesis. Once we get the batch sentence translation, we can either train the mapping with the learning rate inherited from last training, or train always with the initial learning rate and keep training and updating learning rate until the learning rate drops to a minimal learning rate we predefined. The first training scheduler fits the normal online algorithm principle best but in practice we find the second scheduler trains the cross-lingual embedding more efficiently.

- Stop Criterion
  Once the results of this approach have converged to a good point, we can break the training loop early. We have considered different factors such as the loss in the embedding training process or statistics on the translation. In practice, we find the model selection methods provided by Conneau et al. [2017] more suitable in our case. They use CSLS retrieval to generate the

translations of 10k most frequent source words. And then compute the average cosine similarity between these deemed translations and use this average as validation metric. Empirically, they find that this simple criterion is better correlated with the performance on the evaluation tasks than those distance based criterion.

Chapter 5

Sentence Translation

As mentioned previously, training a traditional machine translation system requires large parallel data. With cross-lingual word embedding we can already find ambiguous word translations, in this chapter I propose a simple yet effective method to improve quality of translation which starts from the word-by-word translation. We integrate additional models, such as language model of the target side and denoising neural network of the target side to produce meaningful sentence translation. Since all the our models are trained on monolingual corpora, This method is fully unsupervised. Such system surpasses state-of-the-art unsupervised NMT without costly iteratively training.

5.1 Context-aware Beam Search

5.1.1 Language Model

Language models are widely applied in NLP tasks, they assign a probability to a sequence of words so that they can improve the quality of systems outputs like machine translation, spell correction, speech recognition and question-answering, etc.,.According to the similarity of cross-lingual word embedding, we are able to find some meaningful for translation candidates for a given word. But there are also words that actually noise in the candidates or obviously incorrect because of grammar checking. With the support of language model, we can select the most probable words given previous word translation candidates.

N-gram language models use the Markov assumption to break the probability of a sentence into the product of the probability of each word given a limit history of preceding words.

$$p(e_1^I) = \prod_{i=1}^{I} p(e_i|e_1, \cdots e_{i-1}) = \prod_{i=1}^{I} p(e_i|e_{i-(I-1)}, \cdots, e_{i-1})$$

The conditional probability can be calculated from N-gram model frequent counts:

$$p(e_i|e_{i-(n-1)}, \cdots, e_{i-1}) = \frac{count(e_{i-(n-1)}, \cdots, e_i)}{count(e_{i-(n-1)}, \cdots, e_{i-1})}$$

Language model can handle sparse data problem. Some words or phrases have not been seen yet in the training corpus does not mean they are not impossible. Different smoothing techniques like back-off or interpolation are implemented to assign a probability mass to unseen cases.

5.1.2 Beam Search

When using LM, the complexity of a search graph is exponential to the length of the given source sentence. Beam search is a heuristic search algorithm that explores a graph by expanding the most promising nodes. At each step of the search process, it will evaluate all the candidates together with the reserved translation results from last step, it will only store a predetermined number (beam size) of translations for next step. The greater the beam size is, the fewer states will be pruned. Tt is suggested to prune poor word translation candidates as soon as possible to reduce the search space and speed up the translation.
In this thesis, translation system does not handle reordering, it combines the language model and lexicon model to construct a word-by-word framework, where the output length equals that of input sentence. Since the lexicon model actually gives a cosine similarity instead of a normalized probability, we introduce weights λ_{LM} and λ_{lex} to scale the contribution of each of the two components:

$$\hat{e}_1^J = \underset{e_1^J}{\operatorname{argmax}} \prod_{i=1}^{J} p^{\lambda_{LM}}(e_i|e_{i-(n-1)}^{i-1}) \cdot q^{\lambda_{lex}}(f_i, e_i)$$

where the lexicon score $q(f,e) \in [0,1]$ defined as:

$$q(f,e) = \frac{d(f,e)+1}{2}$$

$d(f,e) \in [-1,1]$ cosine similarity between $\boldsymbol{f}$ and $\boldsymbol{e}$
In experiments, we find linear scaling works better than others, e.g. sigmoid or softmax.

5.2 Denoising Neural Network

5.2.1 Denoising Auto-encoder (DAE)

With the help of language model, we improve the quality of word-by-word translation but the results are still far from acceptable. Because of the natural defect of word-by-word translation, word sequence in translation keeps the same as in input. This is contrary to our knowledge that different languages have different grammars also sequences. We implement the sequential DAE to improve the translation output.

Autoencoder is a type of neural network model used to learn efficient data coding, typically dimension reduction in an unsupervised manner. The DAE is to force the hidden layer to discover more robust features by training the autoencoder to reconstruct the input from a corrupted version. It does two things, try to encode the input and try to undo the effect of a corrupt process stochastically applied to the input. In our model, the corrupt input is the word-by-word translation and the output ought to be the standard translation with correct sequence. Since we do not have parallel data as the references. We need to model the denoising process with artificial parallel data. We design three types of noises including a novel noise type to mimic the corrupted sentence. The loss function for DAE is defined as:

$$\mathcal{L}^{auto} = \mathbb{E}_{e_1^I \sim \mathcal{E}}[-\log P(e_1^I|\text{noise}(e_1^I))]$$

where noise(e_1^I) is noisy target sentences where artificial noises are added.

5.2.2 Noise Model

We design three types of noise to handle the fertility and reordering problem, namely reordering noise, insertion noise and deletion noise.

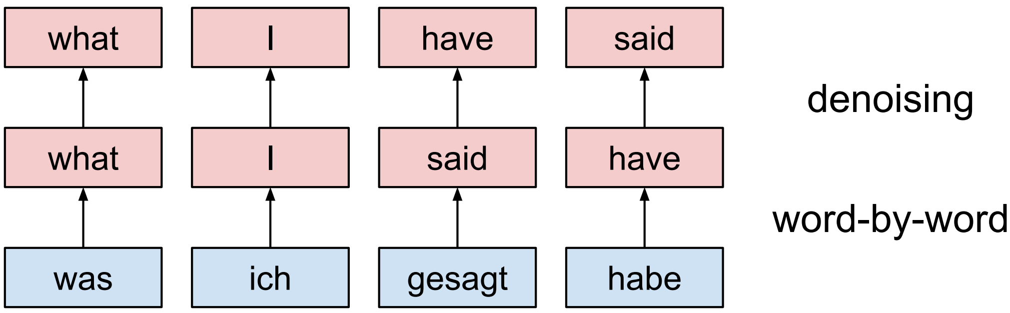

Figure 5.1: Reordering noise

Reordering Noise

The reordering problem is a common phenomenon in the word-by-word translation since the word order in source language is not the same in target language. For example, in the grammar of German, the verb is often placed at the end of the clause. “um etwas zu tun”. However in English, it is not the case; the corresponding translation sequence is “to do something”. The verb should always before the noun. In our beam search, LM only assists in choosing more suitable word from the translation candidates, and cannot reorder the word sequence at all.

For a clean sentence from the target monolingual corpora, we corrupt the word sequence by permutation operation. We limit the maximum distance between the original position and its new position.
The design of reordering noise is as followed:

1. For each position i, sample an integer δ_i from $[0, d_{per}]$
2. Add δ_i to index i and sort $i + \delta_i$
3. Rearrange the words to be in the new positions, to which where indices have been moved

Reordering actually depends on specific language pair. The reordering noise here just models the most general case.
Insertion Noise

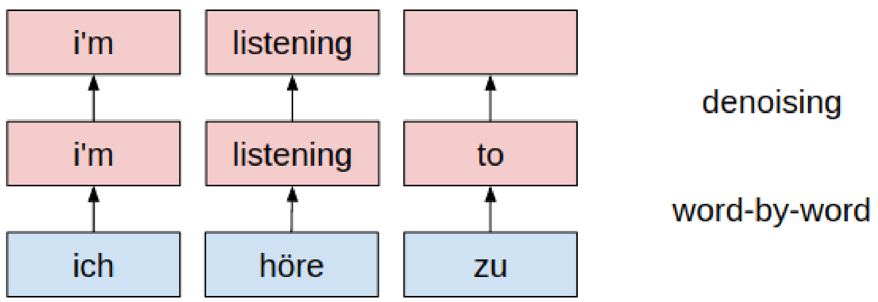

Figure 5.2: Insertion noise

The word-by-word translation system predict a target word at every source position of the sentence. However, the vocabularies of different languages are not symmetric. For example, in German, there are more compound words than that in English. So when translating between languages, there are a plenty of cases that a single word will be translated to multiple words and multiple words correspond to a single word conversely. For example: from a German sentence: “ich höre zu” to “i’m listening”. A very frequent word “zu” which corresponds to “to” in English, is dropped from the sentence. The design of reordering noise is as followed:

1. For each position i, sample a probability $p_i \sim \text{Uniform}(0, 1)$
2. If $p_i < p_{ins}$, sample a word e from the most frequent V_{ins} target words and insert it before the position$i$

We limit the insertion word in a set consisting of the top frequent word in the target language V_{ins}

Deletion Noise
The deletion noise is just a contrary case of insertion noise. Because we are limited to generate only one word per source word, it is also possible that a target word in the reference is not related to any source word. For example for "eine der besten" the corresponding translation is "one of the best". We need to add an extra preposition in the target sentence. To simulate such situation, we drop some words randomly from a clean target sentence.

1. For each position i, sample a probability $p_i \sim \text{Uniform}(0, 1)$
2. If $p_i < p_{del}$, drop the word in the position i

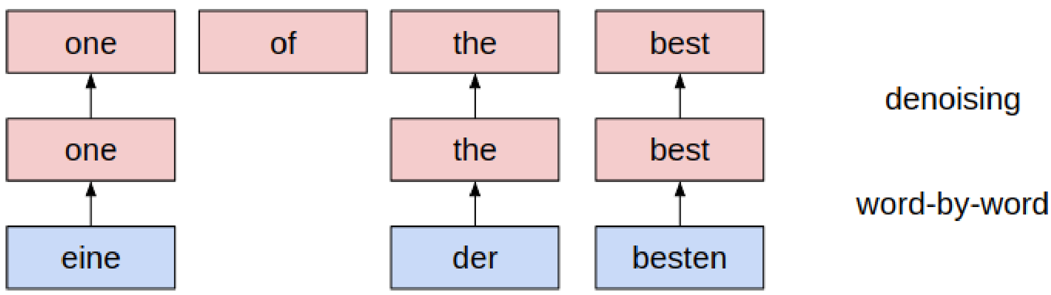

Figure 5.3: Deletion noise

Chapter 6

Experiments

In this chapter, we present experimental results of our proposed methods: corpus-based unsupervised learning algorithm for cross-lingual word embedding and unsupervised MT system integrating LM and DAE. We also empirically analyze the effect of different factors in these methods: corpus size, embedding unit, vocabulary size and artificial noises. Before that, we describe the parameter settings, evaluation methodology, and datasets for our experiments.

6.1 Corpus Statistics

The training corpora for embedding and LM training are the same, containing 100M sentences sampled from News Crawl 2014-2017 monolingual corpora. The data was lowercased and filtered to have a maximum sentence length 100. The test datasets are for MT evaluation, from WMT 2016 German↔English task and WMT 2014 French↔English task. We restrict MT vocabulary size to 50k. OOV rates reflect that the vocabulary size we set is reasonable. Each LM perplexity is in the proper range. Detailed statistics are in Table 6.1, 6.2.

		German	**English**	**French**
Train	**Sentences**	**100M**	**100M**	**100M**
	Running Words	**1880M**	**2360M**	**3017M**
	Vocabulary	**1254k**	**523k**	**660k**

Table 6.1: Training corpora for embeddings and LM

		newstest2016		newstest2014		
		German	**English**	**French**	**English**	
	Sentences	**2999**	**2999**	**3003**	**3003**	
Test	**Running Words**	**62506**	**64619**	**81165**	**71290**	newste
	Vocabulary Size	**11978**	**8645**	**10899**	**9200**	
	OOV Rates	**4116 (6.6%)**	**1643 (2.5%)**	**1731 (2.1%)**	**1299 (1.8%)**	
	LM perplexity	**211.0**	**109.6**	**51.2**	**84.6**	

Table 6.2: Evaluation corpora for MT

6.2 Experimental Setup

We train monoligual word embeddings with a dimensionality of 300 using FastText (Bojanowski et al. [2016]) for 5 epochs, only for words that have occurs at least 10 times. Our corpus-based approach is also implemented on MUSE framework (Conneau et al. [2017]). Vocabulary sizes for source and target side are limited to 50k. 5-gram LMs are trained separately using KenLM with Kneser–Ney smoothing (Heafield [2011]).
LM supported context-aware beam search is performed among top 100 word translation candidates for each position with beam size 10.
DAE is implemented in Sockeye (Hieber et al. [2017]), using RNN and Transformer architectures. RNN-based DAE has 2-layer encoder and 2-layer decoder, the hidden layer size is 512. In Transfomer based DAE, encoder and decoder both have 6 layers and hidden layer size is also 512. The multi-head attention mechanism uses 6 heads. DAEs are trained on smaller monolingual corpora containing 20M sentences.
We evaluate different cross-lingual embedding learning methods on cross-lingual word retrieval tasks. The datasets are released by Conneau et al. [2017], which contains dictionaries for 12 language pairs. Each test dictionary contains 1500 unique source words for evaluation.

6.3 Word Translation

We conduct the cross-lingual word retrieval task on English↔German pair. Empirically, we find the best sentence batch size is 1k and the best initial lr is 0.0001. Numbers in Table 6.3 are the accuracy (%) of results, with the same sentence batch size and initial lr. The last three rows are the baseline experiments: adversarial training, adversarial training with refinement, and supervised learning. Dictionary with 5000 unique source words are provided for supervised learning. We run the novel corpus-based approach on a relative small corpus with 50k sentences, just top

50k sentences from the training corpus using different lr schedulers. The first one is to start from initial lr, the second one is to inherit the lr after the training on previous batch sentences. The first lr scheduler always performs better than the second one. We demonstrate the results for German→ English and English → German with the first scheduler, also the performance of German→ English with second scheduler. In order to lift the performance of the second scheduler, we augment corpus by training over the 50k corpus for two epochs and extend the corpus size to 200k. The results get better but still not as good as first lr scheduler. Results are in Table 6.4. Figure 6.1 demonstrates retrieval accuracy curve w.r.t the number of training batches.

		De-En	En-De
Corpus-based	lr scheduler 1	60.37	60.57
	lr scheduler 2	51.50	\
Corpus-based without LM	lr scheduler 1	54.36	\
Adversarial		53.50	\
Adversarial + Refinement		64.92	\
Supervised		65.38	\

Table 6.3: Cross-lingual word retrieval accuracy

In general, the results for mappings from English to German and from German to English are near, this is as we expected. Without LM the corpus-based approach can still learn a mapping, while the accuracy increases very slowly and converges to a level much lower than the normal one. The second lr scheduler learns very slow at the beginning and then tends to a good point, it seems that it keeps learning.

Corpus	Accuracy (%)
50k	51.50
50k $\times 2$	55.51
200k	57.87

Table 6.4: Word retrieval accuracy on different training corpus

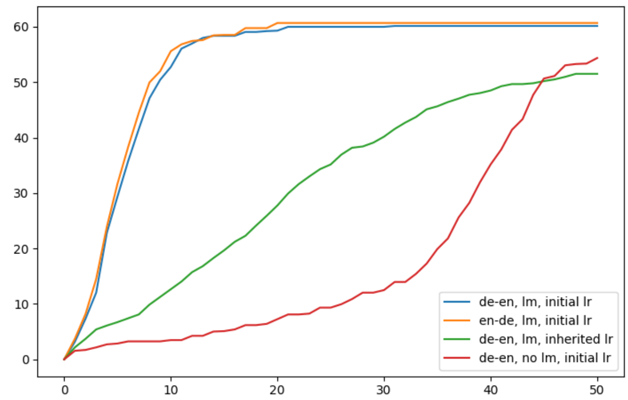

Figure 6.1: Comparison of different experiments

6.4 Sentence Translation

6.4.1 Overall Results

System	De-En BLEU [%]	En-De BLEU [%]	Fr-En BLEU [%]	En-Fr BLEU [%]
Word-by-Word	11.1	6.7	10.6	7.8
+ LM (5-gram) + tgt w/ high LM score for OOV	12.9	8.9	12.7	10.0
+ LM (5-gram) + copy from src for OOV	14.5	9.9	13.6	10.9
+ Denoising (RNN)	16.2	10.6	15.8	13.3
+ Denoising (Transformer)	17.2	11.0	16.5	13.9
Lample et al. [2017]	13.3	9.6	14.3	15.1
Artetxe et al. [2017b]	-	-	15.6	15.1

Table 6.5: Translation results on German↔English `newstest2016` and French↔English `newstest2014`

As demonstrated in Table LM improves wordby-word baselines consistently in all four tasks, giving at least +3% BLEU. When our denoising model is applied on top of it, we have additional gain around +3% BLEU. Note that our methods do not involve any decoding steps to generate pseudo-parallel training data, but still perform better than unsupervised MT systems that rely on repetitive back-translations by up to +3.9% BLEU.

Table 6.6: Word-by-word translation from German to English

	Accuracy [%]	Bleu [%]
5M	44.9	9.7
10M	51.6	10.1
50M	59.4	10.8
100M	61.2	11.2

As in Table 6.9, larger corpus improves the word translation accuracy as well as the word-by-word translation.

6.4.2 BPE vs Word

Byte pair encoding (BPE)(Sennrich et al. [2015]) is a simple data compression technique for word segmentation. It allows for the representation of an open vocabulary through a fixed-size vocabulary of variable-character sequences, making it very suitable word segmentation strategy for neural network models. It helps to reduce the vocabulary size and they eliminate the presence of unknown words in the output translation. We use BPE to represent the mapping between languages and compare with word-level cross-lingual embeddings. In BPE algorithm, the vocabulary is updated using an iterative greedy algorithm. Merges in Table 6.7 denotes the pre-defined number of merge operations.

Vocabulary		Bleu [%]
	Merges	
BPE	20k	10.4
	50k	12.5
	100k	13.0
	Cross-lingual training	
Word	20k	14.4
	50k	14.4
	100k	14.5
	200k	14.4

Table 6.7: Different embedding units and vocabulary size

We examine how the translation performance varies with different vocabularies of cross-lingual word embedding in Table 6.7. The first three show that BPE embeddings performs worse than word embeddings, especially with smaller vocabulary size. For small BPE tokens (1-3 characters), the context they meet during the embedding training is much more various than a complete word, and a direct translation of such small token to a BPE token of another language would be very ambiguous. For word level embeddings, we compared different vocabulary sizes used for training the cross-lingual mapping. Surprisingly, cross-lingual word embedding learned only on top 20k words is comparable to that of 200k words in the translation quality. This means that word-by-word translation with crosslingual embedding depends highly on the frequent word mappings, and learning the mapping between rare words does not have a positive effect.

6.4.3 Artificial Noise

d_{per}	p_{del}	p_{ins}	V_{ins}	Bleu [%]
2				14.7
3				14.9
5				14.9
3	0.1			15.7
	0.3			15.1
3	0.1	0.1	10	16.8
			50	17.2
			500	16.8
			5000	16.5

Table 6.8: Different artificial noises

To examine the effect of each noise type in denoising autoencoder, we tuned each parameter of the noise and combined them incrementally as in Table 6.8. Firstly, for permutations, a significant improvement is achieved from $d_{per} = 3$, since a local reordering usually involves a sequence of 3 to 4 words. With $d_{per} le5$, it shuffles too many consecutive words together, yielding no further improvement. This noise cannot handle long-range reordering, which is usually a swap of words that are far from each other, keeping the words in the middle as they are.
Secondly, we applied the deletion noise with different values of p_{del}. 0.1 gives +0.8% BLEU, but we immediately see a degradation with a larger value; it is hard to observe more than one one-tomany translations in each German→English sentence pair. Finally, we tested several sizes of V_{ins} for the insertion noise, fixing $p_{ins} = 0.1$. Increasing V_{ins} is generally not beneficial, since it provides too much variations in the inserted word, which might not be related to its neighboring words. Overall, we observe the best result (+1.5% BLEU) with $V_{max} = 50$.

6.4.4 Phrase Embedding

Vocabulary			No LM BLEU [%]	With LM BLEU [%]	Denoising BLEU [%]
Word			11.2	14.5	17.2
Mikolov et al. [2013c]	threshold	100	11.1	13.7	15.6
		500	11.0	13.7	16.2
		2000	10.7	14.0	16.5
Top frequent	count	50k	12.0	15.7	16.8

Table 6.9: Performance of phrase embedding in MT

With word2vec tool (Mikolov et al. [2013c]) we detect phrases in the training corpus. We set threshold value to 100, 500, 2000 to make the number of phrases differs. Fewer phrases will be detected as we lift the threshold value. As to our method, we extract only bigrams that in the top 50k frequent bigrams in the training corpus. We learn phrase embedding in an unsupervised manner as normal word embedding. As demonstrates in Table

Chapter 7

Conclusion

The main contributions of this thesis are 1) a novel corpus-based training approach for cross-lingual word embeddings, 2) integration of LM and DAE to improve the unsupervised translation performance.
The corpus-based approach combines the cross-lingual embedding learning, a word-by-word – a simplified word based machine translation. The improvements of sub-models will feed each other to make the entire system improved.
As to unsupervised MT, we achieved context-aware lexical choices using beam search and solved insertion/deletion/reordering problems using artificial noises. Actually insertion noise is a new noise type that first implemented in sequential denosing tasks and we observe up to 1.5 BLEU score improvement with that noise for translation from German to English. Our approach does no need expensive back-translation but still surpasses state-of-the-art unsupervised NMT systems.

Appendix A

Appendix

List of Figures

List of Tables

Bibliography

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 937–947, 2017.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, 2016.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 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)*, volume 1, pages 451–462, 2017a.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017b.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. *arXiv preprint arXiv:1805.06297*, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Antonio Valerio Miceli Barone. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. *arXiv preprint arXiv:1608.02996*, 2016.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*, 2016.

Trevor Cohn and Mirella Lapata. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 728–735, 2007.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. Trans-gram, fast cross-lingual word-embeddings. *arXiv preprint arXiv:1601.02502*, 2016.

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. Multi-view learning of word embeddings via cca. In *Advances in neural information processing systems*, pages 199–207, 2011.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. Learning crosslingual word embeddings without bilingual corpora. *arXiv preprint arXiv:1606.09403*, 2016.

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, 2014.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756, 2015.

Bradley Hauer, Garrett Nicolai, and Grzegorz Kondrak. Bootstrapping unsupervised bilingual lexicon induction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 619–624, 2017.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828, 2016.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics, 2011.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*, 2017.

Yunsu Kim, Julian Schamper, and Hermann Ney. Unsupervised training for large vocabulary translation using sparse lexicon and word classes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 650–656, 2017.

Philipp Koehn. *Statistical machine translation.* Cambridge University Press, 2009.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*, 2018.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256, 2015.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015a.

Thang Luong, Hieu Pham, and Christopher D Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, 2015b.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013b.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013c.

Masato Neishi, Jin Sakuma, Satoshi Tohda, Shonosuke Ishiwatari, Naoki Yoshinaga, and Masashi Toyoda. A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 99–109, 2017.

Malte Nuhn and Hermann Ney. Em decipherment for large vocabularies. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 759–764, 2014.

Malte Nuhn, Arne Mauser, and Hermann Ney. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 156–164. Association for Computational Linguistics, 2012.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *arXiv preprint arXiv:1804.06323*, 2018.

Sujith Ravi and Kevin Knight. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 12–21. Association for Computational Linguistics, 2011.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902*, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, 2015.

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