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 — 365655

Name, Vorname — 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

In recent years, cross-lingual representation of words enable us to explore the structures of different languages in a shared space. Based on word similarity in a cross-lingual space. we can realize 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 exploiting an extra language model and its corpus-based instead of vocabulary-based training process. We can make better use of contextual information. We further combine the translation system with denoising neural network to handle reordering and build a simple yet efficient unsupervised machine translation system. The performance of cross-lingual word embedding is measured on a word retrieval task released byConneau et al. [2017]. The results are comparable with the current supervised and unsupervised methods. And 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 (byte pair encoding, BPE) 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 a enormous collection of parallel data. NMT systems often fail when the training data is not enough. However parallel corpora are costly 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 such monolingual corpora are readily available.
Recently cross-lingual word embedding draws more and more attention since it helps to exploit 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 without any parallel data. This correspondingly inspires work on unsupervised machine translation.
In this thesis, I first make a survey of the current training algorithm of cross-lingual word embedding, find the significant training skills among them and their respective advantages and disadvantages. Then I propose a novel data-based unsupervised training method, the experiments the method can achieve considerable results in comparison with the start-of-the-art method. I further explore the application of the cross-lingual word embeddings in the machine translation domain. I develop a total unsupervised machine translation system starting from the simple word-by-word translation principle, to over the shortcomings to such more, I design the 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 noise in the word-by-word translation, The results demonstrate that such simple but efficient translation system can surpass the 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 occurs with semantic neighbors, having 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 word 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 Cao et al. [2016] 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 totally unsupervised but the performances are far worse than the supervised training. Conneau et al. [2017] simplifies the adversarial training structure with different loss

functions for the generator and discriminator, improving the performance significantly, They also propose to use cross-domain similarity local scaling (CSLS) to handle hubness and a validation criterion for unsupervised model selection.

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: $\mathbf{LM}_f$, $\mathbf{LM}_e$

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

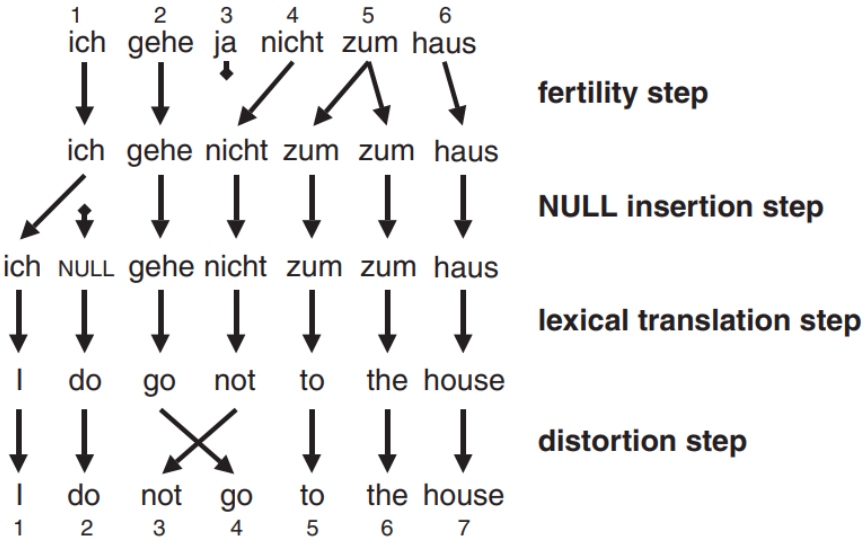

Figure 2.1: Illustration of a translation process using IBM-3 model (Koehn [2009])

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.

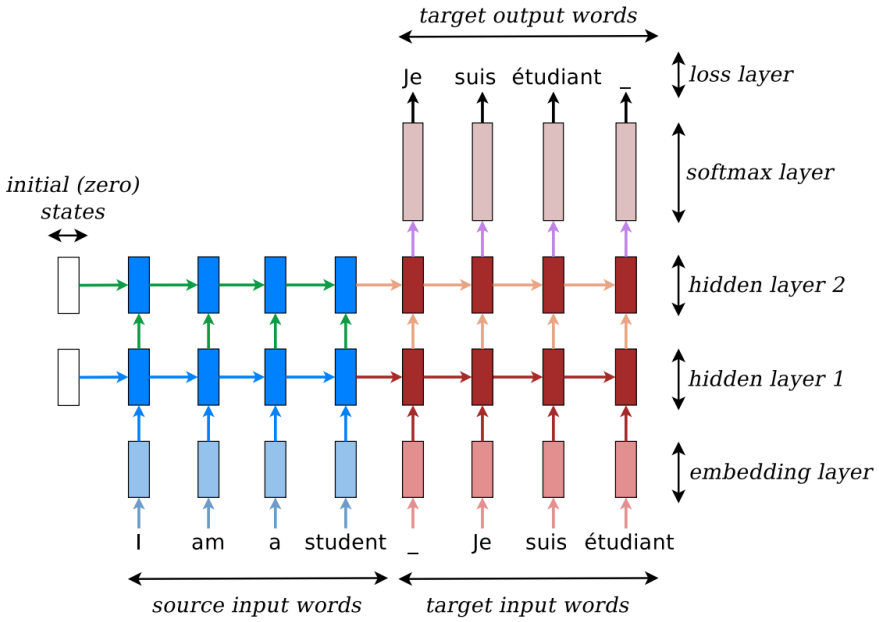

Figure 2.2: English→ French translation– example of a deep NMT (Luong et al. [2015a]).

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:

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

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):

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

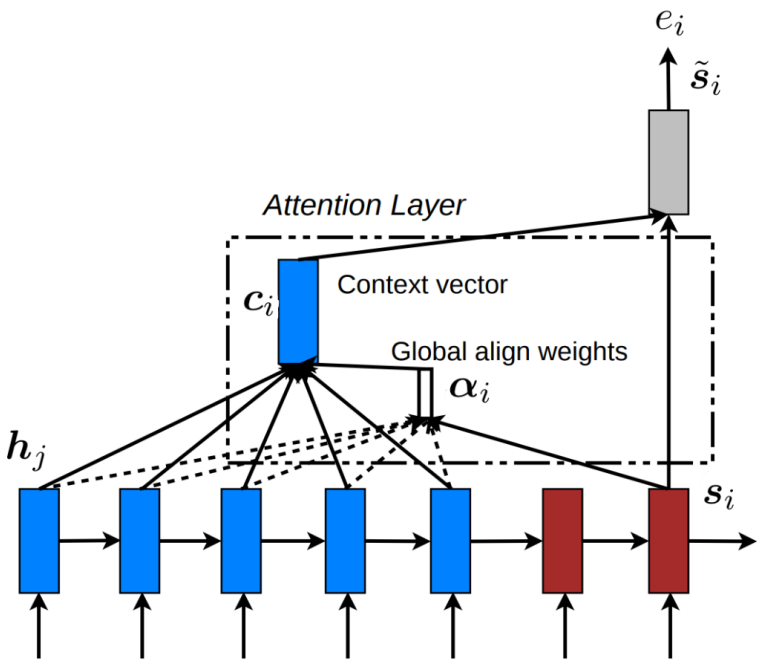

Figure 2.3: Global attention (Luong et al. [2015a])

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:

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

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

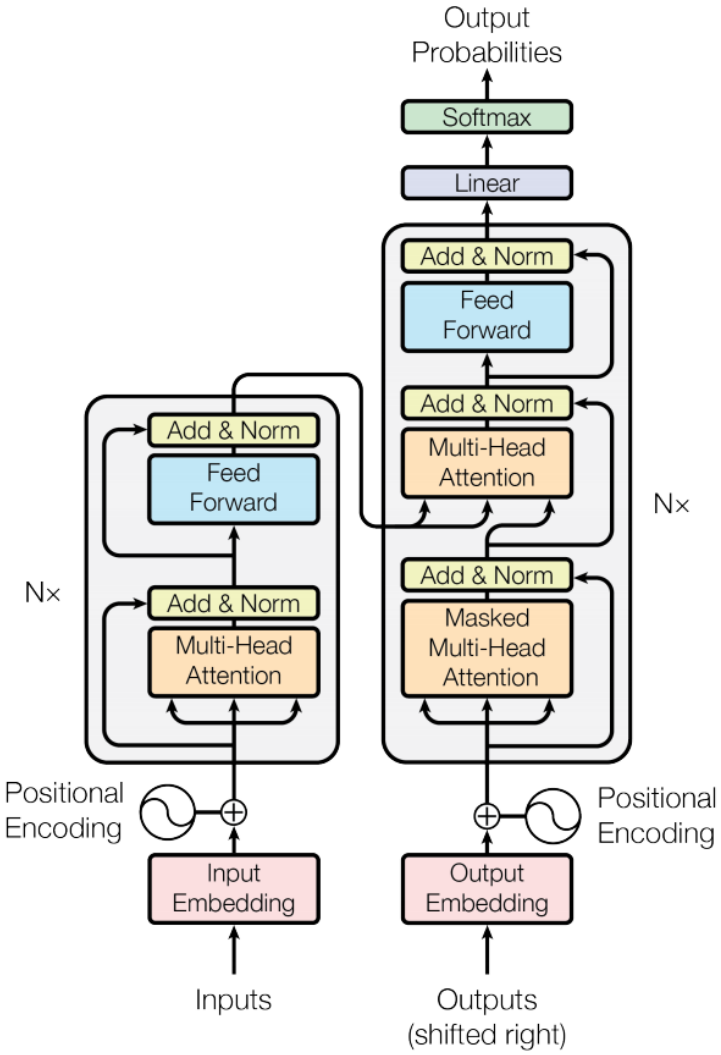

Figure 2.4: The Transformer model architecture (Vaswani et al. [2017])

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] models 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, $\boldsymbol{h}'_i$ is computed as a weighted sum of a linearly transformed input elements:

$$\boldsymbol{h}'_i = \sum_{j=1}^{J} \alpha_{ij}(\boldsymbol{h}_j W^V)$$

Each weight coefficient α_{ij} is computed using a softmax function:

$$\alpha_{ij} = \frac{\exp r_{ij}}{\sum_{k=1}^{J} \exp r_{ik}}$$

And r_{ij} is computed using a compatibility function:

$$r_{ij} = \frac{(\boldsymbol{h}_i W^Q)^\top (\boldsymbol{h}_j W^K)}{\sqrt{d}}$$

where $W^K, W^Q, W^V \in \mathbb{R}^{d \times (d/h)}$ are parameter matrices that are unique per layer and attention head.

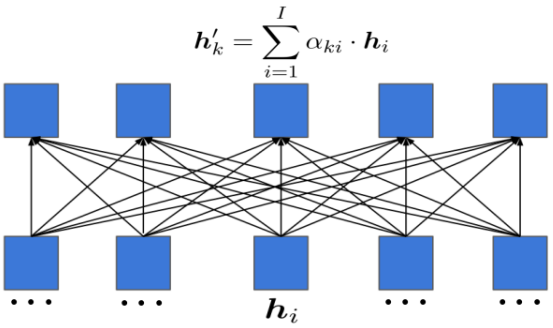

Figure 2.5: Self-attention structure

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.

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:

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

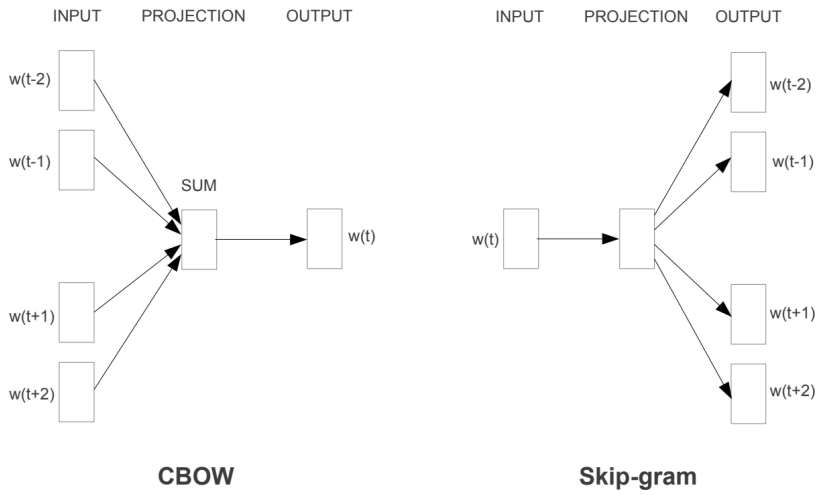

Figure 3.1: Global attention model (Mikolov et al. [2013a])

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

Similarly, we define the loss for CBOW model as

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

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.

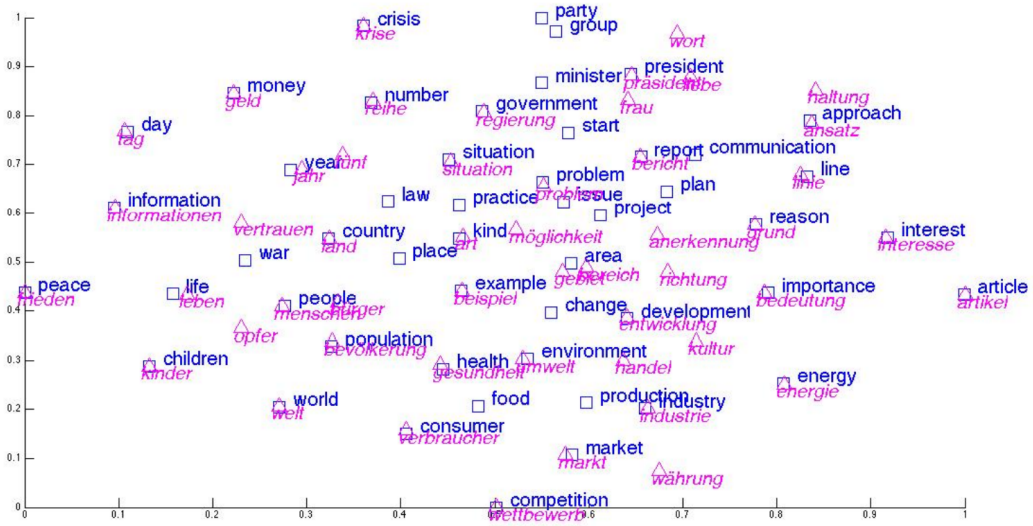

Figure 3.2: A cross-lingual embedding space between German and English (Ruder et al. [2017])

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 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

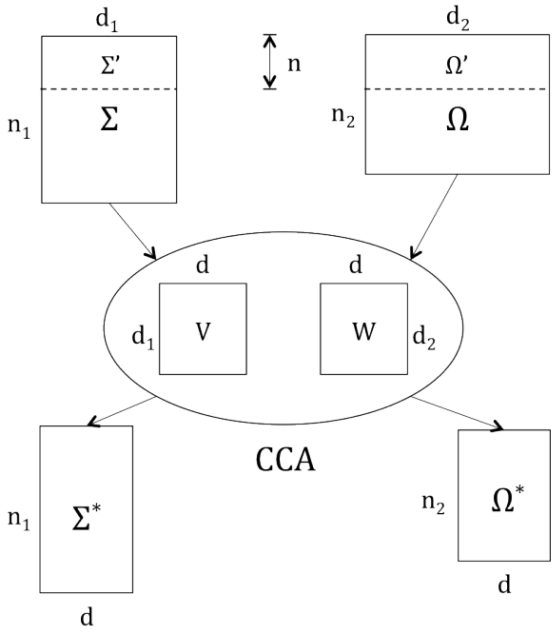

Figure 3.3: Cross-lingual projection with CCA (Faruqui and Dyer [2014])

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:

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

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

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

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

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

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

  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.

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

  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:

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

- 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 embedding 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:

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

Chapter 4

Unsupervised Learning of Cross-lingual Word Embedding

Chapter 3 introduce 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 cost lots of labor. This motivates people to explore fully unsupervised method. In this Chapter, several unsupervised approaches will be explained including a novel data-drive approach, which is distinct from other methods by exploiting an extra language model and corpus-based instead of vocabulary-based training process. These approaches contains always a iterative training procedure. Mapping will be refined until reach an optimum value.

4.1 Initialization

Pre-trained monolingual embeddings capture distribution statistics for each language. These embeddings are typically not aligned between languages and ince the alignment method employed is iterative, a good initialization is a key. A practical approach for reducing the need of bilingual supervision is to design heuristics to build the seed dictionary. propose to use shared words and cognates to extract the seed dictionary.This method has a deficit that it depends on the writing system. A recent line of using adversarial training to learn a mapping, which roughly align the two distributions of embeddings is proposed.

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. This noisy seed lexicon is considered as initial signal. Generate the list of words sorted by frequency first. For each of the most frequent source words, start from the top of the target word frequency list, if the normalized edit

distance (NED) and absolute difference between the respective frequency ranks is within a specific threshold value, then add word pair into the seed lexicon. There is no one-to-one constraint, so both source and target words may appear multiple times in the seed.

4.1.2 Adversarial Training (GANs)

The adversarial training was for cross-lingual embedding mapping is first proposed by , 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 discruminates between the mapped embeddings and the true target embeddings. Zhang et al. [2017] incorporate additional techniques like noise injections to aid the training. Conneau et al. [2017] drop the reconstruction component, regularize the mapping to be orthogonal.
Let $= \{\boldsymbol{f}_1, \cdots, \boldsymbol{f}_\}$ and $= \{\boldsymbol{e}_i, \cdots, \boldsymbol{e}_m\}$ be the two sets of n and m word embeddings from a source and a target language separately, 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, simply a linear transformation, is trained to fooling discriminator. In the two-player game, we are supposed to learn the mapping from source embedding space to the target space.
Generator loss

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

Discriminator loss

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

4.1.3 Similarity Matrix

4.2 Vocabulary-based Training

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 . Apply the Procrustes solution on the current seed dictionary, use the learned mapping matrix W and the nearest neighbor search to construct a better seed dictionary again. Repeat this process iteratively to obtain a hopefully better mapping and

dictionary each time until some convergence criterion is met. As for the convergence criterion, 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 $1e-6$.

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 Seed Dictionary Induction

Specifically, for each word in the most frequent words, we use nearest neighbors to find best candidates. 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 found that the number of word pairs in the dictionary is not so important as the dictionary quality. The bidirectional dictionary, which belongs to the intersection of two unidirectional dictionaries, work better than other set operation.

4.3 Corpus-based Training

4.3.1 Motivation

The vocabulary-based training approach have maken good use to the similarity of embedding distributions. It dose not consider the semantic or contextual information when training the cross-lingual mapping. We propose to make use to corpus translation to learn the mapping, since we do not have parallel sentences and alignment information, we can build a word-based machine translation, which exploits the language model to improve the translation quality. The intuition is that if the sentence translation system, which contains lexicon model, get improved compared with the previous one, we can further learn a better mapping. Repeat this process the lexicon translation and embedding mapping will be trained alternatively, until some convergence criterion is met.

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: $W(embeddingmapping)$
while *not converge* **do**
    $\mathcal{E} \leftarrow TRANSLATE(\mathcal{F}, F, E, \mathbf{LM}_e)$
    $W \leftarrow LEARN\_MAPPING(\mathcal{F}, \mathcal{E})$
end

Algorithm 3 summarize this iterative framework I propose.
The approach combines embedding mapping learning, dictionary induction technique and word-based machine translation, in this thesis, the word-based translation is simplified to word-by-word translation. However, the efficiency turns out to be critical for this approach. By enclosing the learning logic in a loop, the training time depends mainly on the corpus size. It will be very costly if we translate the entire source corpus. Especially at the beginning stage, the improvement of the mapping is a relative slow process and need more training iteration. It takes too long to translation the whole corpus. I propose an online training method, which can greatly improve the training efficiency.

4.3.2 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: $W(embeddingmapping)$
while *not converge* **do**
    Generate batch of source sentences $\{f_1^J\}$ from $\mathcal{F}$
    $\{e_1^I\} \leftarrow TRANSLATE(\{e_1^I\}, F, E, \mathbf{LM}_e)$
    $W \leftarrow LEARN\_MAPPING(\{f_1^J\}, \{e_1^I\})$
end

Instead of translate the entire corpus, I iterate over the corpus to generate specific number of sentences as a batch for translation. After translating, we extract the

word translation pairs as seed dictionary. For more complicated word-based machine translation system, extraction will also requires the alignment information. In word-by-word translation system, we just use the translation pairs at each position. We could improve the quality of the seed dictionary by filtering pairs, which could be noise in the dictionary according to statistical information. Based on the seed dictionary we can learn the cross-lingual word embeddings using neural network trained with SGD. Further we use the mapping to infer the lexicon model for translation of next batch of sentences. In such close-loop process, mapping learning and sentence translation will be alternatively refined to an optimum state. In general, the efficiency of this online algorithm mainly depends on the corpus size and batch size of sentences, which decide directly the translation time. We can either extend the corpus size or loop over the same corpus more than once in order to enrich the data for this corpus-based approach.

4.3.3 Training Details

- Dictionary Induction

  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 size of the similarity matrix grows with respect to that of the target vocabulary. This does not increase the cost of computation, but it also makes the number of possible solutions grow rapidly. In the experiments we find out that those less frequent words can be noise for when inducing the word translation. And those noise translations would have global negative influence of the translation. We propose to restrict the vocabulary size to the k-top frequent words in the source and target languages, where we found $k = 50000$ 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, some words are used only in certain circumstances. For example, 'home', 'house' are synonyms, but when translating 'zu Haus', we will choose 'home'. With language model, we can choose better word translation candidates according to context words.

  - Language model score
    we use 5-gram language model trained by kenlm tool.

  - Lexicon score
    we use simple linear mapping $q(f,e) = \frac{d(f,e)+1}{2}$, where $d(f,e)$ is the cosine similarity between $\boldsymbol{e}$ and $\boldsymbol{f}$
  - Scaling parameters for sub-models

$$\hat{e}_1^J = \underset{e_1^J}{\operatorname{argmax}} \prod_{j=1}^{J} p^{\lambda_{LM}}(e_j|e_{j-4}^{j-1}) \cdot q^{\lambda_{lex}}(f_j, e_j)$$

    empirically set λ_{lex} as 1 and λ_{LM} as 0.1

  More details will be explained in Chapter 5.

- Optimization Objective
  Suppose we have got synthetic parallel sentences from translation and according to the alignment, we have extract word translation pairs as seed dictionary *dico*, the goal the optimization is to find the best mapping so that the lexicon induction can be generalized as much as possible. The linear mapping $W \in \mathbb{R}^{d \times d}$ will be trained on the pairs to minimize a measure of discrepancy between mapped source word embeddings and the target word embeddings.

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

  where l is a loss function, in this thesis, I compare the results of Euclidean distance loss and cosine similarity loss

- Orthogonal Constraint
  As mentioned previously, the orthogonal preserves the quality of monolingual embeddings. It also preserves the the dot product of vectors and the l_2 distance. In our experiments, we find the orthogonal constraint make training procedure converge successfully, specially at the beginning stage. 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
  There are two different learning rate schedules for our method. Once we get the batch sentences translation, we can either train the mapping with the

learning rate inheriting from last training and then decrease the learning rate, or train always with the initial learning rate and keep training and update learning rate until the learning rate has dropped to the minimal learning rate we preset. The first training schedule fits the normal online algorithm principle best but in practice we find the second schedule trains the cross-lingual embedding more efficiently which may due to the translation task differs for different batches.

- Stop Criterion
  Once the results of this approach have converged to a good point, we can break the training loop early. Several different factors can be selected as the stop criterion such as the loss in the embedding training process or statistic on the translation. In practice, we find the model selection methods provided by Conneau et al. [2017] more suitable in our case. In more detail, they use CSLS retrieval to generate the translations of 10k most frequent source words. They then compute the average cosine similarity between these deemed translations and use this average as validation metric. Empirically, they found that this simple criterion is better correlated with the performance on the evaluation tasks than those distance based criterion. This metric can also be used for hyper-parameter selection.

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 found ambiguous word translation, in this chapter we propose a simple yet effective method to improve quality of translation which starts from the word-by-word translation. We integrate monolingual model like language model and denoising neural network of the target side to produce meaningful sentence translation. Since all the sub-models are trained on monolingual corpora, our proposed method is total unsupervised. Our system surpasses state-of-the-art unsupervised translation without costly iteratively training.

5.1 Context-aware Beam Search

5.1.1 Language Model

Language models are widely applied in natural language processing, especially machine translation which need frequent queries.
n-gram models
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.

$$p(w_1^N) = \prod_{i=1}^{N} p(w_i|w_1, \cdots w_{i-1}) = \prod_{i=1}^{N} p(w_i|w_{i-(n-1)}, \cdots, w_{i-1})$$

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

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

Language model tries to handling sparse data problem because 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

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 stores a predetermined number (beam width) of translations for next step. The greater the beam width is, the fewer states will be pruned. So it is suggested to prune these word translation candidates as soon as possible to reduce the search space and speed up the translation. 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 from previous word translation candidates.
Given a history h of target word before e, the score of e to be the translation of f is defined as:

$$\hat{e}_1^N = \underset{e_1^N}{\operatorname{argmax}} \prod_{n=1}^{N} p^{\lambda_{LM}}(e_n|e_{n-4}^{n-1}) \cdot q^{\lambda_{emb}}(f_n, e_n)$$

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 f and e
In experiments, we find such lexicon score works better than others, e.g. sigmoid or softmax

5.2 Denoising Neural Network

5.2.1 Denoising Auto-encoder

With the help of language model we have actually improved the quality of word-by-word translation but the results are still far from a acceptable one because of the drawback of the word-by-word mechanism, maybe to some degree we can infer the meaning of sentence, but the sequence of sentences depends on the specific language. We implement the sequential denoising autoencoder to improve the translation output.
An autoencoder is a neural network that is trained to copy its input to its output, autoencoders minimize the loss function like:

$$L(\boldsymbol{x}, g(f(\boldsymbol{x})))$$

where L penalizing the difference between the input and output. While a denoising autoencoder (DAE) instead minimizes

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}})))$$

where $\tilde{\boldsymbol{x}}$ is a noise transformation of $\boldsymbol{x}$ and denoising autoencoder will try to learn to ignore the noise in $\boldsymbol{x}$ reconstruct the correct one. Sequential denoising autoencoder will find robust representation of sentences. In practice, denoising autoencoder consists of two parts, namely encoder and decoder. The encoder processes noised data and produces real-valued vectors as an encoding feature of the data. The computational graph of the denosing autoencoder, which attempt to reconstruct the normal input $\boldsymbol{x}$ from it corrupted version $\tilde{\boldsymbol{x}}$. The model is trained by minimize the loss.
For our sequential denoising model, the label sequences would be the monolingual data of the target language. However we do not have the noise input. In order to make the model run correctly, we should mimic the noise sentence of word-by-word translation on the target monolingual corpus.
We design different noise types w.r.t. the word-by-word translation. In the experiments, we inject the artificial noise into a clean sentence, the experiment results shows the noise is reasonable and suitable in this case.

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. In experiments, the noise model can improve the sentence translation, but since it actually starts from the word-by-word translation, it can only deal with reordering in limited distance, cannot work for global reordering.

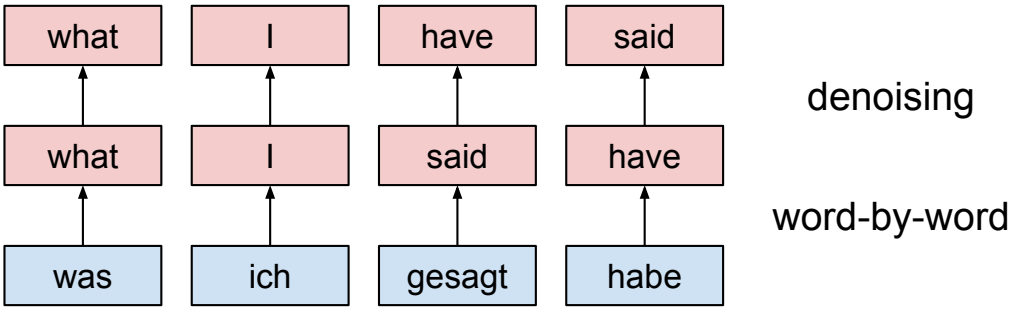

Figure 5.1: Reordering noise

Reordering Noise

The reordering problem is a common phenomenon in the word-by-word translation since the sequence in source language is not exact the sequence 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, language model only assisting in choosing more suitable word from the translation candidates, it 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 is actually depends on the specific language pair. However in the experiments we found the performance of the denoising network aimed at such noise is not obvious. The Bleu score before and after the process is close.

Insertion Noise

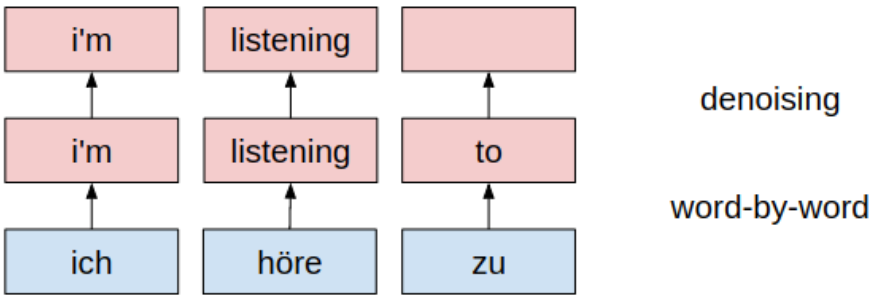

Figure 5.2: Insertion noise

The word-by-word translation system predict the source word at every position of the sentence. However the vocabularies of different systems are not symmetric, for example, in German there are more compound words than that in English. So when

translating cross languages, there are a plenty of cases that a single word will be translated to multiple words and multiple words correspond to a single conversely. We focus on such a case: 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 ne 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

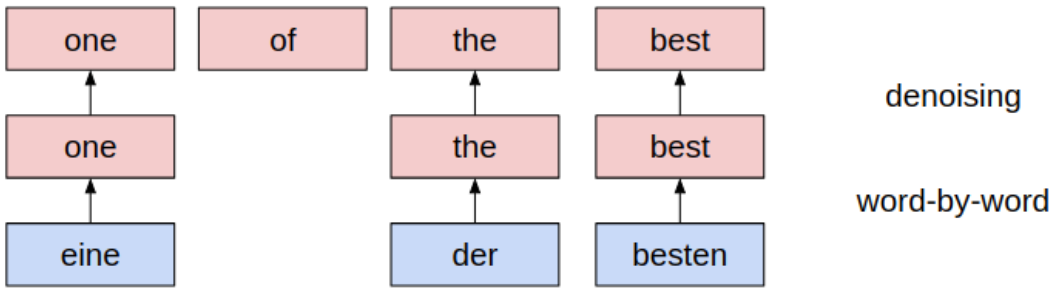

Figure 5.3: Deletion noise

Chapter 6

Experiments

6.1 Corpus Statistics

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

		newstest2016		newstest2014	
		German	English	French	English
Test	Sentences	2999	2999	3003	3003
	Running Words	62506	64619	81165	71290
	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

Search vocabulary in testing: 50k (src/tgt)

6.2 Word Translation

6.2.1 Baseline

Corpus Size for Monolingual Embedding

		De-En				En-De			
		5M	10M	50M	100M	5M	10M	50M	100M
Supervised	Procrustes-NN								
	Procrustes-ISF								
	Procrustes-CSLS								
Unsupervised	Adv-NN								
	Adv-CSLS								
	Adv-Refine-NN								
	Adv-Refine-CSLS								

Vocabulary Size for Word Translation

Accuracy(%)	50k	100k	150k	200k	No Cutoff
50k					
100k					
150k					
200k					
No Cutoff					

6.2.2 Data-driven Approach

Different Learning Rate Scheduler

Different Learning Rate Scheduler

- Start from Initial Learning Rate
  - German → English

Accuracy (%)	100	500	1k	2k	5k
0.01	1.00	6.01	24.21	28.37	42.71
0.005	0.85	5.47	7.71	17.12	38.78
0.001	1.23	7.32	14.88	15.34	27.45
0.0005	1.54	45.26	49.88	54.43	54.97
0.0001	36.23	58.09	60.14	60.37	58.67

  - English → German

Accuracy (%)	1k	2k	5k
0.01	10.02	16.50	23.36
0.005	47.65	22.28	24.21
0.001	24.44	24.98	26.75
0.0005	54.82	55.67	48.44
0.00001	59.21	58.98	56.90

  - Learning Rate Decay

- Inherit Last Learning Rate
  - 50k

Accuracy (%)	1k	2k	5k
0.01	7.79	4.70	6.01
0.005	3.39	12.03	8.94
0.001	5.40	11.49	18.89
0.0005	5.78	8.33	10.25
0.00001	51.50	37.63	15.27

  - 50k 2 epochs

Accuracy (%)	0.5	0.7	0.9
0.01	3.78	1.39	1.31
0.005	4.39	3.31	14.73
0.001	5.63	10.56	13.72
0.0005	7.56	6.71	10.64
0.0001	55.51	56.36	57.59

  - 200k

Accuracy (%)	0.5	0.7	0.9
0.01	3.24	2.08	0.46
0.005	5.32	12.72	1.23
0.001	4.55	8.02	17.27
0.0005	6.71	6.94	6.79
0.0001	57.83	57.83	57.67

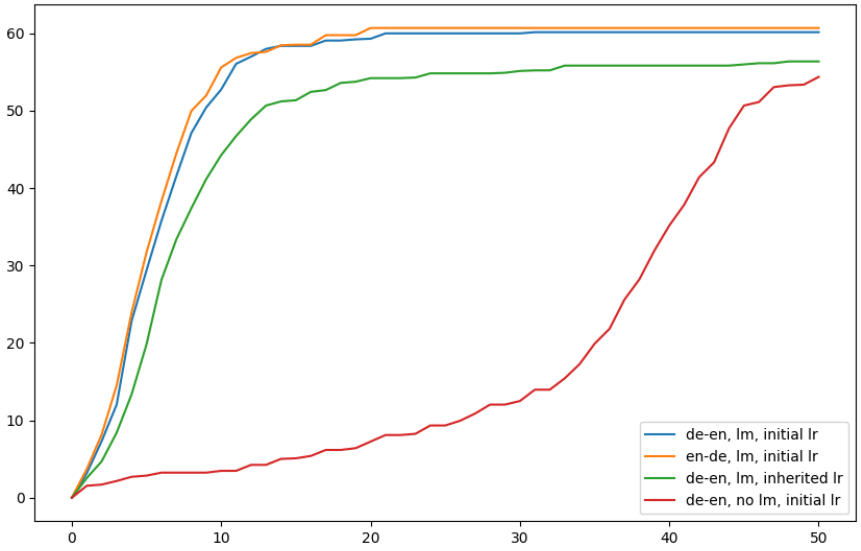

Figure 6.1: A cross-lingual embedding space between German and English (Ruder et al. [2017])

6.3 Sentence Translation

6.3.1 Comprehensive Table

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**

6.3.2 BPE vs Word

Byte pair encoding (BPE) is a simple data compression technique for word segmentation. It allows for the representation of an open vocabulary through a fixed-size

Table 6.1: 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**

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.

	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**

As the experiments results demonstrated above, the BPE performs worse than word in this unsupervised learning scenario. It might be difficult to learn the translation relationship between subword units.

6.3.3 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

6.3.4 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

6.3.5 Vocabulary Cutoff in Translation

Table 6.2: Word embedding vocabulary cut-off

Bleu [%]	20k	50k	100k
50k	11.1	11.3	11.2
100k	11.2	11.2	11.1
150k	10.9	10.9	-

Table 6.3: Phrase embedding vocabulary cut-off

Bleu [%]	50k	100k	150k
50k	11.3	-	-
100k	11.9	11.9	-
150k	12.0	11.9	11.9
200k	12.0	-	-

Chapter 7

Conclusion

For unsupervised machine translation system, the context-aware beam search language model help at the lexicon choice step.
The denoising networks which aimed at the insertion/deletion/reordering noise in the word-by-word translation sentence works well for fertility and localized reordering problem. Even though BPE is known to be an effective way to overcome the rare word problem in standard NMT, BPE embedding performs worse than word embedding in our case especially when vocabulary is small.
Word-by-word translation based on cross-lingual word embedding depends highly on the frequent word mappings. We found that phrase embedding only helps in word-by-word translation with context-aware beam search, it works not as good as that under the processing of denoising autoencoder.

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 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.

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

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

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. A distribution-based model to learn bilingual word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1818–1827, 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.

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.

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.

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.