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

The unsupervised learning of cross-lingual word embedding offers elegant word matches between languages, but there are fundamental limitations in sentence translation. In this work, we propose a simple yet effective method to improve the word-by-word sentence translation based on cross-lingual word embedding, using only monolingual corpora but without any back-translation. We integrate a language model for context-aware beam search, we also combine with a novel denoising autoencoder to handle reordering. Our system surpasses state-of-the-art unsupervised machine translation system without costly iterative training. We also analyze the effects of parameters and artificial designed features in our model. We also propose a novel data-driven unsupervised learning of cross-lingual word embedding, which can be actually describe as an iterative process combing the translation process and SGD training of word embedding. Since the quality of cross-lingual word embedding effects the performance of our model. Our contributions to embedding learning and translation system can provide better understanding of learning the cross-lingual word embedding and its usage in translation.

Contents

Chapter 1

Introduction

Neural machine translation (NMT) has recently become the dominant method to machine translation task. In comparison to the traditional statistical machine translation (SMT), NMT systems are trained end-to-end, taking advantages of the continuous representation of the hidden states that greatly alleviate the sparsity problem and make better use of more contextual information. Building a good machine translation system requires a huge collections of parallel data. NMT systems often fail when the training data is not enough. The lack of parallel corpora is actually a problem. Parallel corpora are costly to build because they requires huge human labor and time and sometime expertise of corresponding languages. Conversely, the monolingual corpora is readily available.
Several approaches has been proposed to alleviate this issue, for instance, triangulation and semi-supervised learning techniques, how these systems still need a strong cross-lingual signal. Unsupervised machine translation uses only monolingual data of the source and target language to train the model.
Recently the work on cross-lingual word embedding can help to eliminate the need of cross-lingual information. Cross-lingual word embedding can be defined as the word embedding of multiple languages in a joint embedding space. In our work, we consider the learning of cross-lingual word embedding space as the task of learning the mapping from source embedding space to target embedding space.
In this work, we propose a simple yet efficient unsupervised machine translation system based on cross-lingual word embedding, we build the system by combining the context-aware beam search and using denoising autoencoder to handle reordering. Our system surpasses the state-of-the-art performance of unsupervised machine translation systems without iterative training. We also analyze the effect of detailed model parameters. We analyze the factors for cross-lingual word embedding training, propose a novel unsupervised training method using stochastic gradient descent instead of popular Procrustes training. We demonstrate that our algorithm can achieve the same performance with the state-of-the-art unsupervised methods.

For future work we hope to implement an iterative algorithm to polish our unsupervised machine translation model

1.1 Related Work

1.1.1 Word Embedding

Traditional language processing systems treat words as discrete atomic symbols, they assign for each word a specific id number. Such encoding are arbitrary, and it does not provide any information about the relations that may exist between individual symbols. In comparison, with learning algorithm each word can be represented in high dimensional continuous space. According to the distributional hypothesis, similar words tends to occurs with similar neighbors, so similar words have similar word word representation. Mikolov et al. [2013b] first noticed that continuous embedding spaces exhibit similar structures across languages, even when for distant language pairs for example English-Vietnamese.

There are many successful methods Mikolov et al. [2013a] Mikolov et al. [2013c]for continuous distributed representation of words. Exploiting word occurrence statistics, word vectors reflect the semantic similarities and dissimilarities. Similar words are close in the embedding space. Pennington et al. [2014]

Bojanowski et al. [2016]

Mikolov et al. [2013b] first noticed that continuous word embedding space exhibits similar structures according languages, even for distant language pairs. They proposed to learn similarity by learning a linear mapping from a source to a target embedding space. They employed a parallel vocabulary of five thousand words as anchor points to learn this mapping and evaluated their approach on a word translation task.

In practice, Mikolov found that the similarity cross languages can be represented by a linear transformation.Xing et al. [2015] showed that results can be improved by enforce an orthogonal constraint on the the linear mapping.

Recently, Artetxe et al. [2017a] proposed an iterative method that align the word embedding spaces gradually. He still need a parallel vocabulary like digits or to start the procedure.

Cao et al. [2016] Zhang et al. [2017] proposed a method using adversarial training without any parallel data. The discriminator tried to discriminate if the embedding from the source side and the generator aimed to learning the mapping from source embedding space to target one.

Conneau et al. [2017] viewed the word translation task as a word retrieval task and find the nearest neighbor searching suffers from the hubness problem. They further proposed to use cross-domain similarity local scaling (CSLS) penalize the hubs and a validation criterion for unsupervised model selection.

1.1.2 Unsupervised Machine Translation

As mentioned previously, the lack of parallel corpora motivate people use more information for example Some researchers tried to use triangulation techniques and semi-supervised approaches to address.
Cohn and Lapata [2007] Chen et al. [2017]
There are considerable works on statistical decipherment to induce a machine translation model from monolingual data. Ravi and Knight [2011] first proposed some idea about the unsupervised machine problem, he considered the unsupervised translation as a deciphering problem. The source language is considered as ciphertext He also proposed iterative EM method and Bayesian decipherment to learn the unsupervised model. To solve the memory bottleneck of such model, in further work Nuhn et al. [2012] tried to according to context similarity to limit search candidates Nuhn and Ney [2014], beam search and preselection search was combined to limit the search space. Kim et al. [2017] enforced the sparsity with a simple threshold for the lexicon. He also tried to initialize the lexicon training with word classes, which efficiently boosts the performance.
Although initially not based on distributional semantics, recent studies show that the use to word embeddings can bring significant improvement in statistical decipherment Duong et al. [2016]
He et al. [2016] made an important contribution to train neural model for unsupervised machine translation. More concretely, their method trains two agents to trainlate 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] Lample et al. [2017] proposed two bi-directional unsupervised machine translation model which totally rely only on monolingual corpora in each language. These two models both need to use cross-lingual word embedding to initialize the MT system and, train the sequence-to-sequence system as denoising autoencoder and turn the unsupervised training into a supervised one by introducing back-translation techniques. Lample et al. [2018] proposed two model variants, a neural and a phrase-based model. These models have fewer hyper-parameters. And these models performs well also on some distant and low-resource languages.

1.2 Outline

In summary, this work makes the following main contributions:
The remainder of this thesis is structured as follows. In Section 1.3 we introduce Chapter 2 introduces the development of machine translation systems from statistical models to to neural models. Some basic techniques and principles for machine translation are also included. Chapter 3 gives more information, I do a

survey of training and applications details of cross-lingual word embedding. We discuss the unsupervised learning of cross-lingual word embedding and our efforts on data-drive training. We describe our unsupervised machine translation model in chapter 5, which contains mainly context-aware part: with the help of language model and denoising autoencoder aimed at reordering. We demonstrate the results of cross-lingual word embedding model and unsupervised machine model. We will compare the performance with the state-of-the-art model. In Chapter 7 we summarize our work which helps better understanding of learning the corss-lingual word embedding and its usage in translation domain.

1.3 Notation

Normally we denote

- source sentence $f_1^J := f_1 \cdots f_j \cdots f_J$
- target sentence $e_1^I := e_1 \cdots , e_i \cdots e_I$
- single character e, f separately
- sentences e_1^N, f_1^N when word-by-word translating
- $\boldsymbol{e}$ and $\boldsymbol{f}$ the source and target word embedding
- $\boldsymbol{E}$, $\boldsymbol{F}$ are the corresponding embedding matrices
- source and target Corpora $\mathcal{S}, \mathcal{T}$
- model probability with subscripts s, t such as:
  $P_{s \to s}(\cdots)$, $P_{t \to t}(\cdots)$ for denoising autoencoder, $P_{s \to t}(\cdots)$, $P_{t \to s}(\cdots)$ for translation model

Chapter 2

Machine Translation

This chapter describes the classical or start-of-the-art machine translation models from statistical machine translation model to neural machine model. The relative research works on unsupervised machine translation will also be included.

2.1 Statistical Machine Translation

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

2.1.1 Word-Based Model

Noisy-channel model: Noisy channel model based on the notion of a noisy channel from Shannon's information theory has been applied to many language processing problems. Assume 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 $Pr(e_1^I)$), our task is to find the best translation e_1^I for an input foreign sentence.

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

The basic model for word-based model is noisy-channel model. The task of word alignment is an artifact of word-based machine translation. Alignment models target at the reordering problem for word-based translation.
Alignment model: 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

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

Zero-Order Models: IBM-1 & IBM-2 Models

We reformulate the translation model with alignment:

$$Pr(f_1^J | e_1^I) = \sum_{a_1^J} Pr(f_1^J, a_1^J | e_1^I)$$

Further decomposed into a length model, alignment model and a lexicon model.

$$\begin{aligned} Pr(f_1^J, a_1^J | e_1^I) =& Pr(J|e_1^I) \cdot Pr(f_1^J, a_1^J | J, e_1^I) \\ &= Pr(J|e_1^I) \cdot Pr(a_1^J | J, e_1^J) \cdot Pr(f_1^J | a_1^J, J, e_1^I) \end{aligned}$$

Assumptions for IBM1 and IBM2 models:

- length model dependent only on length of target sentence:

$$Pr(J|e_1^I) = P(J|I)$$

- alignment model dependent only on absolute position and sentence length:

$$Pr(a_1^J | J, e_1^J) = \prod_{j=1}^{J} p(a_j | j, I, J)$$

- lexicon model dependent on aligned words only:

$$Pr(f_1^J | a_1^J, J, e_1^I) = p(f_j | e_{a_j})$$

Difference by alignment model:

- IBM-1 model: uniform probabilities: $p(i|j, I, J) = \frac{1}{I+1}$
- IBM-2 model: $p(i|j, I, J)$ is a large table which can be initialized by IBM-1 model

IBM-1 model consider all possible reordering as the same possibility. More often than not that, the word translation for input source word also follows the translation of the predecessor of that word, or more generally in limited distance. In IBM-2 model, we add an explicit model for alignment.

First-Order Model: HMM

Assume a_j depends on immediate predecessor position a_{j-1} as in speech recognition:

$$Pr(a_1^J|J, e_1^J) = \prod_j^J p(a_j|a_{j-1}, I, J)$$

Weighted Model

To overcome shortcomings in modeling, introduce scaling exponents λ_m as in speech recognition:

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

where e_h denotes the history words in N-gram model

Fertility: IBM-3 Model

Mostly, one word in the source language corresponding to just one single word in the target sentence. But fertility models that each source word can be translated to zero or more than one words.
Fertility models $\varphi(e)$ the specific number of target words given an source word e. As consequence, length J depends on fertilities:

$$J = \sum_{i=0}^{I} \varphi(e_i)$$

Since fertility models the one-to-many translation or dropping input words. Similarly, we need to model adding words. IBM models these added words are generated by the special NULL token. We could model the fertility of the NULL token in the same way. After fertility step, we insert one NULL token with a specific probability p after each generated word.

IBM-3 model contains four steps:

- fertility step
- NULL insertion
- lexical translation step
- distortion step for reordering

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 languages 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 This model allow s straightforward integration if 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.

2.2 Neural Machine Translation

Unlike traditional phrase-based machine translation, which consists of several models that are tuned separately, neural machine translation tries to build a more general neural network model which can directly output translations given input, it

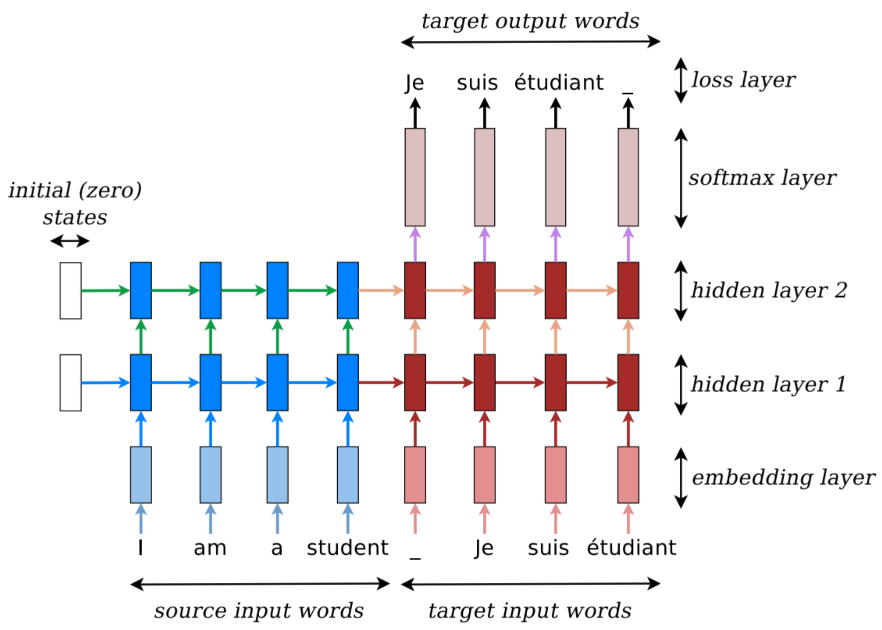

Figure 2.1: Neural machine translation – example of a deep recurrent architecture (Luong et al. [2015])

contains only one single model, and only one single training criterion.
The first successful neural machine translation , attention mechanism has lately been used to improve neural machine translation by selectively focusing on parts of the source sentence during translation. The inherently sequential nature precludes parallelization within training examples.

2.2.1 Seq2seq Model

The most common network structure is the encoder-decoder framework and

$$p(e_1^I|f_1^J) = \prod_t p(e_i|e_0^{i-1}, f_1^J) = \prod_t p(e_i|e_{i-1}, h_{i-1}, f_1^J)$$

A basic form of NMT consists of two components: an encoder which computes a representation $\boldsymbol{s}$ for the whole source sentence and a decoder which generates one target word a a time and hence decomposes the conditional probability as:

$$p(e_1^I|f_1^J) = \prod_{i=1}^{I} p(e_i|e_{i-1}, \boldsymbol{s})$$

In more detail, the probability of decoding each word e_i as:

$$p(e_i|e_{i-1}, \boldsymbol{s}) = softmax(g(\boldsymbol{h_i}))$$

g is trainable function which gives the probability of all target words in the vocabulary. $\boldsymbol{h}_i$ is the RNN hidden unit, abstractly computed as:

$$\boldsymbol{h_i} = f(\boldsymbol{h}_{i-1}, \boldsymbol{s})$$

where f defines the transformation connecting hidden states.
Drawbacks of the such seq2seq model:

1. The model compresses all information from the input sentence into a hidden vector 1, 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 the target word, the performance will get worse.
2. It's 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

To solve the problem mentioned above, attention mechanism was proposed to derive a context vector $\boldsymbol{c_t}$ that capture the input information to help to predict the target word at time t. The basic idea is: given the target hidden state $\boldsymbol{h_t}$ and the source-side context vector $\boldsymbol{c_t}$, we can compute the hidden state $\tilde{\boldsymbol{h}}_i$ by combining the current hidden state $\boldsymbol{h_t}$ and the context vector $\boldsymbol{c_t}$:

$$\tilde{\boldsymbol{h}}_t = tanh(W_c[\boldsymbol{c_t}; \boldsymbol{h_t}])$$

Then the target word can be predicted by softmax function:

$$p(e_i|y_{\leq i,x}) = softmax(W_s\tilde{\boldsymbol{h}}_t)$$

The concept of attention mechanism comes first from the computer vision domain Xu et al. [2015], when generating image captions, The model learns to restrict attention to particular objects in the image.
Bahdanau et al. [2014] **Global attention**
The global attention attend all the input words, weighted sum of

$$a_i(\boldsymbol{s}) = \text{align}(\boldsymbol{h_i}, \bar{\boldsymbol{h}}_s) \tag{2.1}$$

$$= \frac{\exp(\text{score}(h_i, \bar{\boldsymbol{h}}_s))}{\sum_{\boldsymbol{s}'} \exp(\text{score}(h_i, \bar{h}'_s))} \tag{2.2}$$

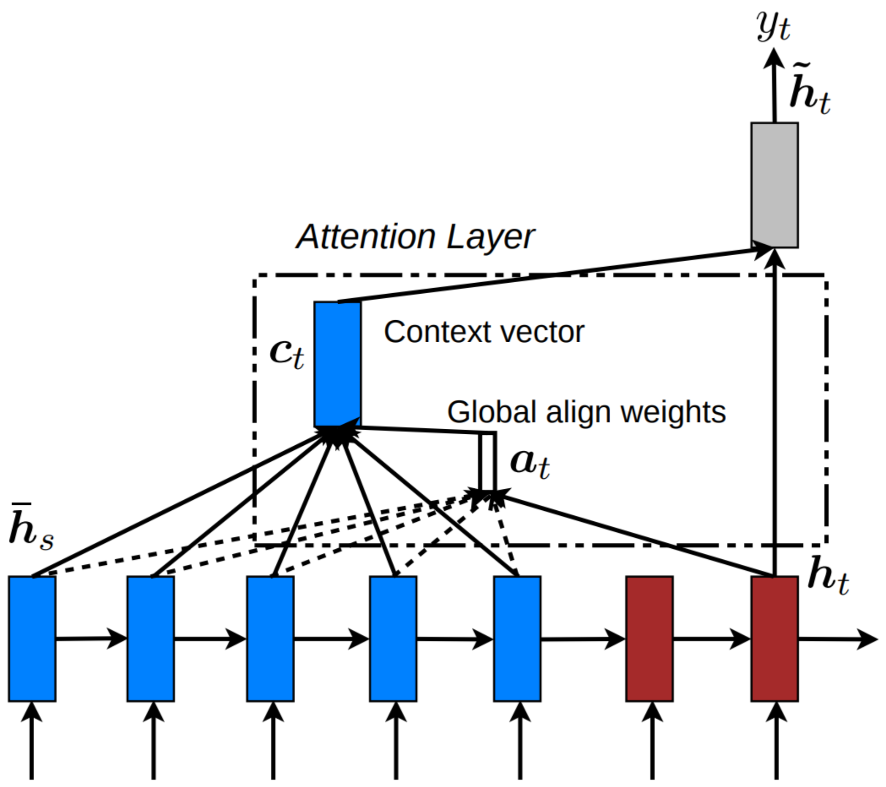

Figure 2.2: Global attention model (Luong et al. [2015])

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

with soft attention you need to calculate the attention over all features, however with hard attention, it is a deterministic methods so that Hard attention
Soft attention means when computing the distribution of the alignment, for each word in the input sentence, the model will give a probability.
In comparison with the soft attention, the hard attention will determine a specific word from the input sentence as the alignment and force the alignment probability as 0. Hard attention mechanism works in image processing however it performs worse in text processing. Because such one-to-one alignment will produce bad translation once a mis-alignment occurs. Luong et al. [2015]
Local attention
Since for global attention, for each target word we need to attend the whole in-

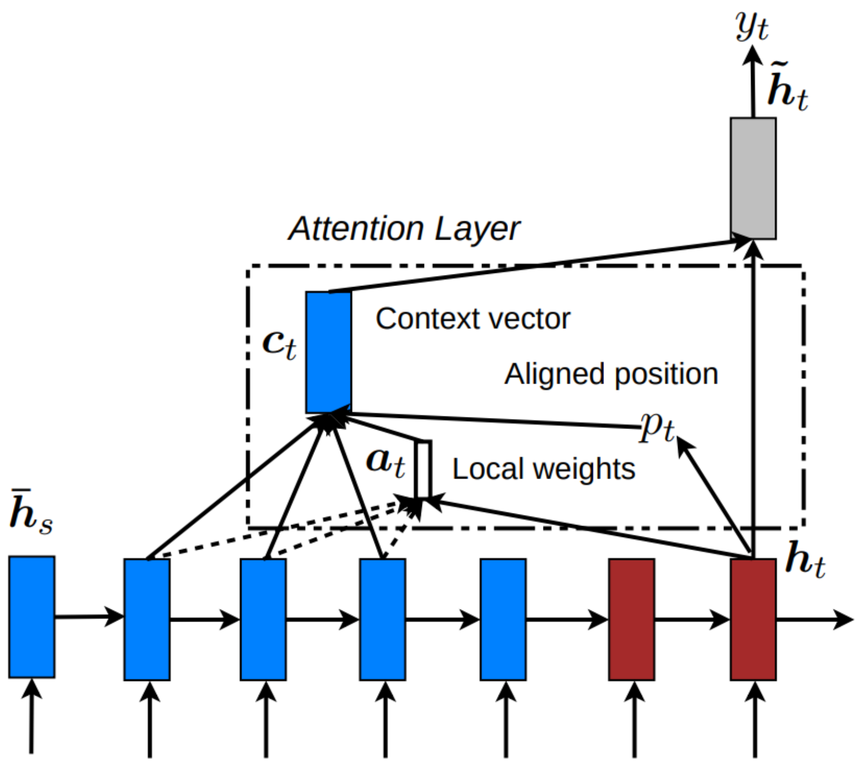

Figure 2.3: Local attention model (Luong et al. [2015])

put sentence, it is very expensive and impractical to translate longer sentences. Proposed the local attention, the local attention is actually the tradeoff between soft and hard attention, Soft attention tries to place attention over the whole image "softly" while select one patch of the image to attend. The hard attention need more complicated techniques like variance reduction or reinforcement learning though need less computation when inference.

Local attention: the model first generate an aligned position p_i for word at the position i. Then the context vector $\boldsymbol{c}_i$ is a weighted sum within the window $[p_i - D, p_i + D]$, D is selected empirically. The model predict the aligned position p_i as followed:

$$p_i = S \cdot \text{sigmoid}(\boldsymbol{v}_p^T \tanh(W_p \boldsymbol{h}_i))$$

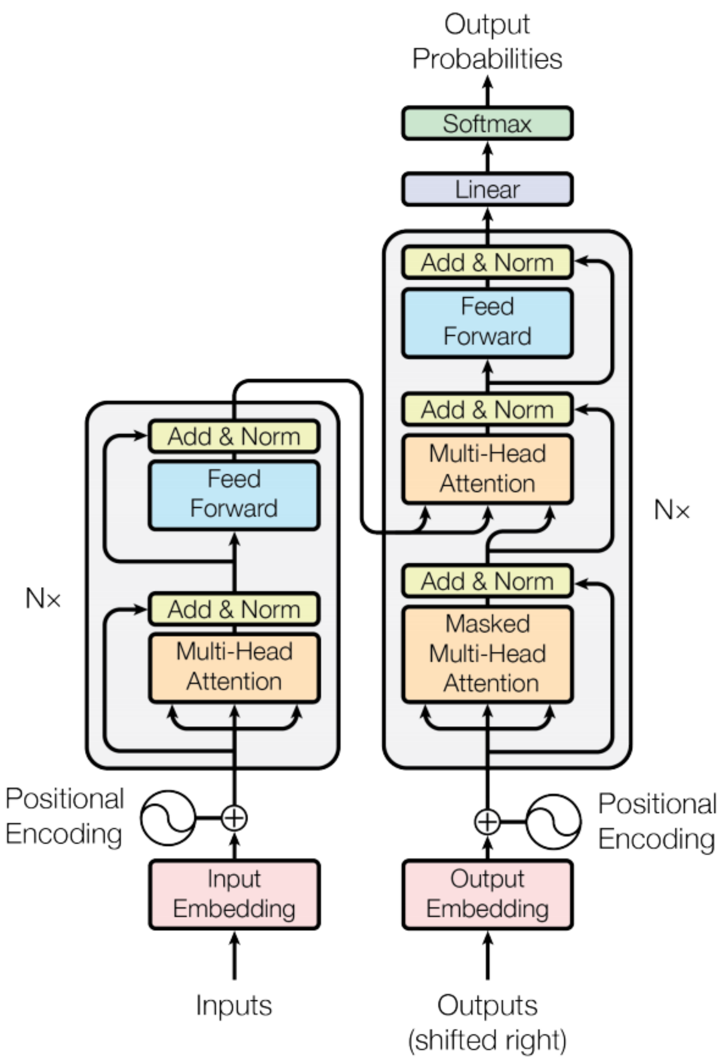

Figure 2.4: (The Transformer model architectureVaswani et al. [2017])

W_p and $\boldsymbol{v}_p$ are the model parameters which will be learned to predict positions. To favor the words near position p_i, we place a Gaussian distribution which centered at p_i.

$$\boldsymbol{a}_i(s) = \text{align}(\boldsymbol{h}_i, \bar{\boldsymbol{h}}_s)\text{exp}\Big(- \frac{(s-p_i)^2}{2\sigma^2}\Big)$$

2.2.3 Transformer

As the illustration above, the long-short term memory (LSTM) and gated recurrent units (GRU) have achieved state of the art in sequence modling and machine translation problem. However the RNN models also have some disadvantages, because of its sequential nature, it is more difficult to fully take advantage of the modern computing devices uch GPU and TPU, which excel at parallel computation not sequential processing.
Dot-Product Attention a query q and a set of key-value (k-v) pairs to an output, the weight of the each value is the inner product of the query and corresponding key. Queries and keys are of the same dimension d_k, values are of dimension d_v, than the attention of query on all (k-v) pairs are:

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j} v_i}$$

When we stack the queries q to Q:

$$A(Q, K, V) = softmax(QK^T)V$$

As d_k get larger, the variance of $q^T k$ get larger. The softmax become very peaked and the gradient get smaller. To counteract this effect, we scaled the dot products by $\frac{1}{\sqrt{d_k}}$.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

multi-head attention: first map Q,K,V into h many lower dimension spaces via linear mapping matrix. Then apply attention and concatenate the outputs. Suppose the original dimensions of queries, keys, values are d_model. We the number of heads is h, then $d_k = d_v = d_{model}/h$. With $i \in 1 \cdots h$ different matrix $W_i^Q \in R^{d_{model} \times d_k}$ $W_i^K \in R^{d_{model} \times d_k}$ $W_i^V \in R^{d_{model} \times d_k}$ and $W^O \in R^{d_{model} \times d_k}$.

$$MultiHead(Q, K, V) = Concat(head_1, \cdots, head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Transformer reduce the training complexity to a constant $O(1)$ number of operations.
The encoder-decoder attention: the queries come from the previous decoder layer, the keys and values are just the output of decoder. This works as the attention mechanism in the seq2seq model, it allow the decoder to align all positions in the input sequence with different weights
encoder self-attention: all queries, keys, values come from the encoder layer, each position allows to attend to all positions before that position.

decoder self-attention: similar to encoder attention, each position is allowed to attend to position before and including that position.
Positional Encoding Since there is no RNN or CNN structures in transformer model, we need also need to make use of sequence information for seq2seq learning. We need inject position information into the embedding.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

where pos is the position and i is the dimension. That is, each dimension of

2.3 Unsupervised Machine Translation

2.3.1 Decipherment

$$\underset{\theta}{\operatorname{argmax}} \prod_{f} \sum_{e} P(e) \cdot P_\theta(f|e)$$

for hidden alignments:

$$\underset{\theta}{\operatorname{argmax}} \prod_{f} \sum_{e} P(e) \cdot \sum_{a} P_\theta(f, a|e)$$

Propose a simple generative story for MT without parallel data. The model accounts for word substitutions, insertions, deletions and local reordering during the translation process but does not incorporate fertilities or global re-ordering.
The generative process:

1. Generate an English sentence e_1^N with probability $P(e)$.
2. Insert a NULL word at any position in the English sentence with uniform probability.
3. For each English word token e_i (including NULLs), choose a foreign word translation f_i, with probability $P_\theta(f_i|e_i)$, the foreign word may be NULL.
4. Swap any pair of adjacent foreign words f_{i-1}, f_i, with probability $P_\theta(swap)$.
5. Output the foreign string f_1^M, skipping over NULLs.

use the Expectation Maximization (EM) algorithm to estimate θ in order to maximize likelihood of the foreign corpus. Finally use the Viterbi algorithm to decode the foreign corpus. and Produce an English translation e_1^N that maximizes

$P(e_1^N) \cdot P_\theta(f_1^M|e_1^N)$ but EM training face some problems, EM cannot scale to such large vocabulary sizes, need to instantiate the entire channel and resulting derivation lattice before we can run EM, this is too big to be stored in memory. Introduce iterative EM: identify the top K frequent word types in both the English and foreign data. Replace all the other word tokens with 'UNK', use EM algorithm to train this model to maximize likelihood of cipher data. Then extend the vocabulary size from K to 2K and so on. Iterative EM Nuhn et al. [2012] Altogether only look at $B_{LM} + B_{lex}$ many successor states. Nuhn and Ney [2014]
Gehring et al. [2017]

2.3.2 Neural Unsupervised Machine Translation

```
Result: Unsupervised Machine Translation
Input: Language models $LM_s$ $LM_t$ over the source and target languages
Initialize translation models: Leveraging $P_s$ and $P_t$, learn two initial translation
  models, one in each direction: $P^{(0)}_{s\rightarrow t}$ and $P^{(0)}_{t\rightarrow s}$
for $k = 1$ to $N$ do
    • Backtranslation: Generate source and target sentences using the current
      translation models $P^{(k-1)}_{t\rightarrow s}$ and $P^{(k-1)}_{s\rightarrow t}$, factoring in language models, $P_s$ and
      $P_t$;
    • Train new translation models $P^{(k)}_{s\rightarrow t}$ and $P^{(k)}_{t\rightarrow s}$: Use the generated sen-
      tences and leveraging $P_s$ and $P_t$
end
```

The key idea is to build a common latent space between the two languages (or domains) and to learn to translate by reconstructing in both domains according to two principles.
They initialize the model with an inferred bilingual dictionary. They leverage strong language model: denoising autoencoder, third, they implemented the back translation: the key idea is to train two translation models which translate in contrary directions at the same time. The last property is that the models constrain the latent representations produced by the encoder to be shared between the two languages. The encoders will encoder the input into a common latent representation space independent of the language. The decoder plays the role of translator and will try to learn to improve the translation quality with the help of back translation mechanism.

Dual Structure

made an important contribution. They train two agents to translate in opposite directions (e.g. French → English and English → French), and make them teach each other through a reinforcement learning process. While promising, this approach still requires a parallel corpus for a warm start.

Initialization

Denoising autoencoder Initialize the algorithm by using a naive unsupervised translation model based on a word-by-word translation of the sentence with a bilingual lexicon derived from the same monolingual data. While such initial "word-by-word" translation maybe poor if languages or corpora are not closely related, it still preserves some of the original semantics

Optimization

Denoising autoencoder
by minimization

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

where s $P_{s \to s}$ and $P_{t \to t}$ are the composition of encoder and decoder both operating in the source and target sides, respectively. **Back-Translation**
Since we have dual structure for bi-directional translation, we denote the sentence that translated by intermediate target-to-source translation model as $u(y)$, similarly denote the sentence translated by source-to-target model as $v(x)$, so $u(y)$ should in source language and $v(x)$ should in target language. The pairs $(x, v(x))$, $(u(y), y)$ constitute synthetic parallel sentences. They can be used to train the two machine translation models by minimizing the back-translation loss:

$$L^{back} = \mathbb{E}_{y \sim T}[-\log P_{s \to t}(y|u(y))] + \mathbb{E}_{x \sim S}[-\log P_{t \to s}(x|v(x))]$$

The objective functipn minimized at every iteration of stochastic gradient descent is simply the sum of L^{auto} and L^{back}. **Shared Latent Representation** A shared encoder representation actis like an interlingua, which is translated in the decoder corresponding language regardless less of the input source language.

- Shared encoder The system use only one encoder that is shared by both languages involved. The universal encoder is aimed to produce a language independent representation of the input language but the decoder should separately translate then into corresponding language.

- Adversarial training Train discriminator to classify between the encoding of source and target sentences. The discriminator operates on the output of the encoder, the encoder is trained instead to fool the discriminator.

Chapter 3

Word Embedding

Xing et al. [2015]

3.1 Monolingual Embedding

3.1.1 CBOW and Skip-gram Model

Word embeddings is distributed representation of words in a vector space. With the learning algorithm it can capture the contextual or co-occurrence information. The word embedding has an interesting and important property: similar words will have similar distribution in the embedding space, with that property, we can find meaningful near-synonyms or Some successful methods for learning word embedding like word2vec Mikolov et al. [2013c] Continuous Bag-of-Words model(CBOW) and Skip-Gram model CBOW model and Skip-Gram model are currently the common structures to learn the word embedding. Algorithmically, CBOW tries to predict the current word based on the context while Skip-Gram model tries to maximize classification of a word based on another word in the same sentence. The neural probability language model defines the prediction probability using the softmax function:

$$
\begin{aligned}
p(w_t|w_s) &= \text{softmax}(s(w_t, w_s)) && (3.1)\\
&= \frac{\exp\{s(w_t, w_s)\}}{\sum_{w' \in W} \exp\{s(w', w_s)\}} && (3.2)
\end{aligned}
$$

where w_t is target word)label word), w_s is the source word(input word), for Skip-Gram model, the target word refers to the context words, the source word refers to the current word, for CBOW model is simply inverted. W denotes the whole vocabulary. Then the training objective of the model is to maximize the log-likelihood on the training dataset, i.e. by maximizing:

$$
\begin{aligned}
J_{ML} &= \log p(w_t|w_s) && (3.3)\\
&= s(w_t, w_s) - \log(\sum_{w' \in W} \exp\{s(w', w_s)\}) && (3.4)
\end{aligned}
$$

However the normalization on the whole vocabulary is very expensive because it is conducted for all words at every training step. The problem of predicting words can be considered as an independent binary classification task. For example in the Skip-Gram model, we consider all the context words as positive samples and the words randomly sampled from the dictionary as the negative ones. Then the training objective is

$$J_{NEG} = \log Q_\theta(D = 1|w', w_s) + \sum_{w' \sim W} \log Q_\theta(D = 0|w', w_s)$$

where $Q_\theta(D = 1|w'w_s)$ is the binary logistic regression probability. In practice, we draw k contrastive words from the noise distribution. Since we only calculate the loss function for k samples instead the whole vocabulary, it becomes much faster to train.

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c<j<c, j \neq 0} \log p(w_{t+j}|w_t)$$

where c is the size of training context, larger context size make the results more precise at the cost of training time. Suppose we are give a scoring function to evaluate the word pair(word, context), the Skip-Gram model

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c<j<c, j \neq 0} \log p(w_t|w_{t+j})$$

According to empirical results, CBOW works better on smaller datasets because CBOW smoothes over a lot of the distributional information while Skip-Gram model performs better when we have larger datasets
Noise-Contrastive Training

3.1.2 fastText

The training methods above treat each word as a distinct word embedding, however intuitively we can obtain more information from the morphological information of words. A subword model was proposed to try to fix such problem.The training network is similar, the model design a new presentation of the word: it adds speicial

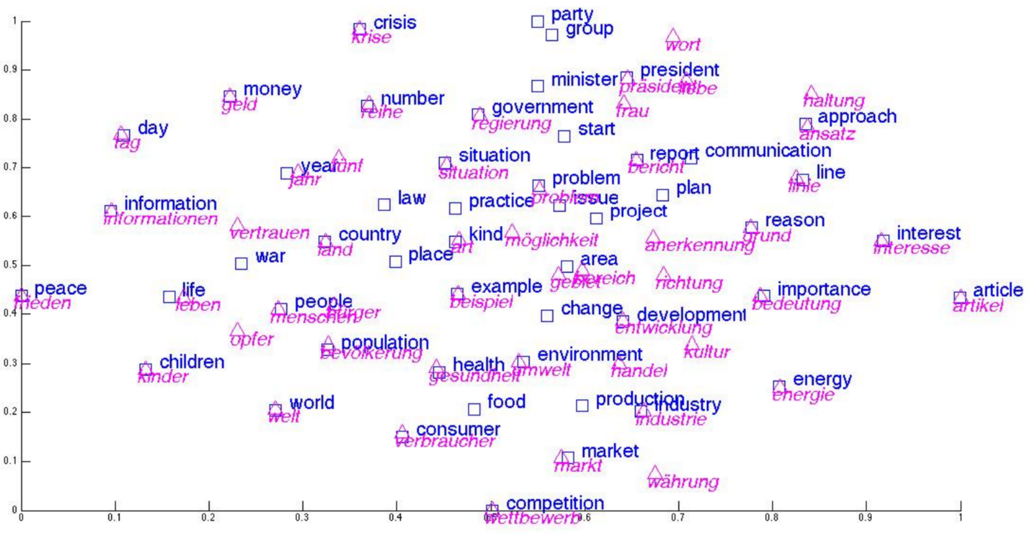

Figure 3.1: A cross-lingual embedding space between German and English (Ruder et al. [2017])

symbols <, > as boundary information at the beginning and the end of a word. Then a normal word is represented as a bag of character n-grams . For example the word "where" and n equals 3, the it can be represented as the following 5 tri-grams:

$$< wh, whe, her, ere, re >$$

Suppose in this way we denote a word w as G_w the set of character n-grams, we assign for each character n-gram g in G_w, we assign a distinct vector z_g, we will finally represent the embedding of word w as the sum of these vector and also for the scoring function:

$$s(w, w_s) = \sum_{g \in G_w} z_g^T w_s$$

3.2 Supervised Learning of Cross-lingual Word Embedding

Cross-lingual word embedding is defined as word embedding of multiple languages in a joint embedding space. Mikolov first notice that the embedding distributions exhibit similar structure across languages. They proposed to use a linear mapping from the source embedding to target embedding.

In the thesis, I assume there are two set of embeddings e, $f$trained separately on monolingual data. The propose of cross-lingual word embedding training is to learn such a mapping $W \in$ from source embedding space to target embedding space, so Wf_i, e_j in the same embedding space and for all corresponding word pairs, we need to optimize the mapping W, so that"

$$\arg \min_{W \in R^{d \times d}} \sum_i \|Wf_i - e_i\|$$

where d is the dimension of embeddings, and the distance $\|Wf_i - e_i\|$ can be different types. We prefer the Euclidean distance.
first observe that word embeddings trained separately on monolingual corpora exhibits isomorphic structure across languages, as illustrated in Figure . 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 , because word embedding requires only plain text to train, which is the most abundant form of linguistic resource.

3.2.1 ***

According to the training method we can divide the supervised method into three:

1. Mapping based approaches
   First train the monolingual word embedding separately and then seek the seed dictionary to learn the mapping.

2. Pseudo-multi-lingual corpora-based approaches
   Use the monolingual embedding training method on constructed corpora that contains both the source and the target language.

3. Joint methods
   Take the parallel text as input and minimize the source and target language losses jointly with the cross-lingual regularization term

A dictionary is necessary for learning the cross-lingual word embedding. minimizing the distance in a bilingual dictionary.

Xing showed that the results are improved when we constrain the W to be an orthogonal matrix. This constraint, the optimal transformation can be efficiently calculated in linear time with respect to the vocabulary size.
The problem then is simplified as the Procrustes problem and there exists a closed-form solution obtained from the SVD of EF^T

3.2.2 Orthogonal Constraints

Starting from Mikolov et al. [2013b], the mapping from source embedding space to target embedding space can be represented as a linear repression. The objective can be defined as:

$$\min_{W} \sum_{i} |Wf - e\|^2$$

Since we retrieve the word translation according to cosine similarity, it's better to solve the problem by redefine the optimization function using the cosine distance:

$$\operatorname*{argmax}_{W} \sum_{i} (Wf_i)^T e^i$$

. We consider the source and target embedding in the same space. In this case, the normalization constraint on word vectors can be satisfied by constraining W as an orthogonal matrix. This is equivalent to minimizing the (squared) Frobenius norm of the residual matrix:

$$W^* = \operatorname*{argmin}_{W} \|WF - E\|_F^2$$

The problem boils down to the Procrustes problem which has a closed form solution obtained from the singular value decomposition (SVD).

$$W* == UV^T, \quad U\Sigma V^T = SVD(EF^T)$$

3.2.3 CSLS Loss

Inspired from the work of Conneau et al. [2017], where the dictionary inducted from CSLS loss: $\boldsymbol{e}$

$$CSLS(\boldsymbol{e}, \boldsymbol{f}) = -2cos(\boldsymbol{e}, \boldsymbol{f}) + \frac{1}{k} \sum_{\boldsymbol{e}' \in N_{\boldsymbol{e}}(\boldsymbol{f})} cos(\boldsymbol{e}', \boldsymbol{f}) + \frac{1}{k} \sum_{\boldsymbol{f}' \in N_{\boldsymbol{f}}(\boldsymbol{e})} cos(\boldsymbol{f}', \boldsymbol{e})$$

since we have $cos(\boldsymbol{We}, \boldsymbol{f}) = \boldsymbol{e}^T \boldsymbol{W}^T \boldsymbol{f}$
The loss function can be rewritten as:

$$\min_{\boldsymbol{W} \in} = \frac{1}{n} \sum_{i=1}^{n}$$

Minimization of a non-smooth cost function over the manifold of orthogonal matrices . Instead of using manifold optimization tools, **?** proposed to derive convex relaxations that can lead to a simple and tractable minimization algorithm.

- Spectral norm: replacing the set of orthogonal matrices 1 by its convex hull, that is the set of matrices with singular values smaller than 1, the unit ball of the spectral norm

- Frobenius norm: replacing the the set of orthogonal matrices 1 with ball of radium $\sqrt{d}$ in Frobenius norm

With such two relaxations, the CSLS loss is constrained to a convex function with respect to the mapping $\boldsymbol{W}$.
We train the linear mapping with in the spectral norm by projected gradient descent:2 For each iteration, train the mapping $\boldsymbol{W}$ with gradient descent, then constrain the mapping by projection of the set.

- Spectral norm
  take the SVD of the trained matrix, threshold the singular values to one

- Probenius norm
  divide the matrix by its Frobenius norm

Chapter 4

Cross-lingual Word Embedding without Parallel Data

Learning a good mapping still need seed word translation pairs as cross-lingual supervision signals. For example in the first work use five thousand seeds to train the linear mapping. Vulic and Korhonen [2016] shows that a seed dictionary of at least hundreds are need for the model to generalize. Zhang et al. [2017] proposed an adversarial autoencoder, using the generator G to implement the mapping so that the transformed source embedding similar to the corresponding target embedding, and the discriminator D strives to distinguish the target embedding from the transformed source embedding. Conneau et al. [2017] simplifies the structure However, large dictionary is also not readily available for many language pairs. Several unsupervised learning algorithms are studies

4.1 Initialization

Artetxe et al. [2018].
As it can be clearly observed that, the self learning algorithm can effectively learn the mapping between the source and target distribution even starts with a small dictionary size and the final accuracy when the algorithm converges are nearly the same level. However when start randomly without any support of dictionary, the framework does not work. In order to realize a fully unsupervised learning, we need to design heuristics to build the seed dictionary.
For total unsupervised learning, there is no direct alignment between the languages. The most challenge for unsupervised learning is the initialization. As proved in the experiment of , the self-learning framework can work even with a small dictionary set, however the model cannot work without such dictionary as hint. So it is important to find heuristic methods to initialize the model.
Several models are proposed: Based on matrices $M_E = EE^T$, $M_F = FF^T$, we can exploit latent information to reduce the mismatch. Assume that, the words in the source and target set are most common. For a specific word, the corresponding line in M_E demonstrates the distribution of similarity in the source vocabulary. M_E,

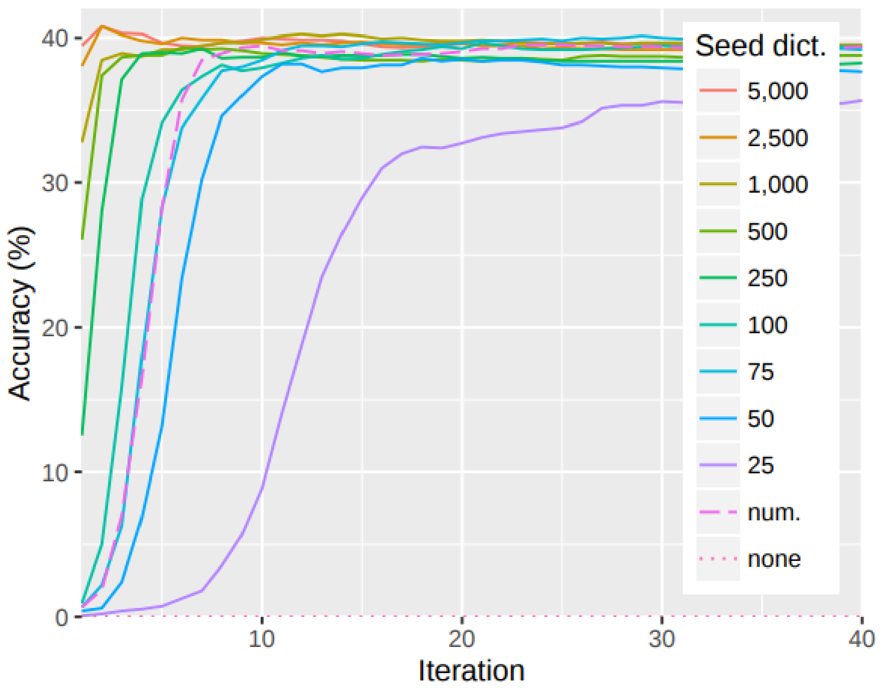

Figure 4.1: Accuracy on bilingual lexicon induction

M_F can be nearly equivalent up to a permutation of the words. Instead of explore all the probability of permutation, he first sorts the values in each row of M_F and M_E
Assume the SVD of $E = USV^T$, the similarity $M_E = US^2U^T$. In practice, he computed sorted 1, 2 yielding the matrix E' , F' and used to build the initial solution for self-learning.

4.1.1 Heuristics

Iterative Closest Point Method

Assume that many language pairs share same principle axes of variation. We do the approximate distribution alignment with PCA. For each language, we first select most frequent word vector,center the data and project it to the top p principle components. We denote the projected language representation as $P_l \in R^{N*p}$ Propose a method first learn T_{ef} and T_{fe} for $E \to F$ and $F \to E$, we include the cycle-constraints ensure that a word f transformed into joint embedding space and transformed back is unchanged.

1. For each e_i, find the nearest $W_{fe}f_j$. denote as $f(e_i)$
2. For each f_j, find the nearest $W_{ef}e_i$, denote as $e(f_j)$
3. For all mini-barches of e_i and f_j in epoch, iterative optimize T_{ef} and T_{fe} on: $\sum_i \|e_i - W_{fe}f(e_i)\| + \sum_j \|f_j - W_{ef}e(f_j)\| + \lambda\sum_i\|e_i - W_{fe}W_{ef}e_i\| + \lambda \sum_j \|f_j - W_{ef}W_{fe}f_j\|$

4.1.2 Adversarial autoencoder

The intuition can be formalized as the minimax game with the value function:

$$V(D, G) = E_{e\sim p_e}[\log D(e)] + E_{f\sim p_f}[\log(1 - D(G(f)))]$$

As a generic binary classifier, a feed-forward neural network with one hidden layer is used to parameterize the discriminator D, and its loss function is the usual cross-entropy loss.

$$L_D = -\log D(f) - \log(1 - D(G(e)))$$

After the generator G transforms a source word embedding e into a target language representation Gf, we should be able to reconstruct the source word embedding f by mapping back with G^T. We therefore introduce the reconstruction loss measured by cosine similarity:

$$L_R = -\cos(f, G^T G))$$

Note that this loss will be minimized if G is orthogonal. With this term included, the loss function for the generator becomes

$$L_G = -\log D(G(x)) - \lambda \cos(f, G^T Gf)$$

where λ is a hyper-parameter that balances the two terms.

4.1.3 Adversarial Training

Discriminator is trained to discriminate between elements randomly sampled from Wf_i and e_j and generator W is trained to prevent the discriminator from making accurate prediction

The training algorithm follows the standard procedure of deep adversarial networks(GAN) of Goodfellow **?**: the discriminator and generator are trained iteratively with the stochastic gradient descent to minimize the L_D and L_w
Let = $\{x_1, \cdots, x_n\}$ and = $\{y_i, \cdots, y_m\}$ 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. Discriminator objective

$$L_D(\theta_D|W) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(source = 1|Wf_i) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(source = 0|e_i)$$

Mapping objective

$$L_W(W|\theta_D) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(source = 0|Wf_i) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(source = 1|e_i)$$

Model Selection

Since the cross-lingual embedding training is under the unsupervised setting, We do not know the word translation accuracy, otherwise if we have the validation data, that means we will have parallel data, against the unsupervised idea. To address this issue, we must select from the property of data or the loss of the neural network as the unsupervised criterion. However in the experiments we find that the accuracy of the discriminator always stays at a high level no matter how is the word translation accuracy.
All these methods can be use to find meaningful word pairs in both languages. For further refinement we can use the induced dictionary to start the iterative self-learning algorithm.

4.2 Iterative Training

4.2.1 Self-learning with Procrustes

Self-learning framework

4.2.2 Data-driven Method

Training procedure:

1. translate corpus according to current mapping, get the word pairs D
2. train the network with D to minimize the mapping distance
3. Repeat $1, 2$ until the algorithm converges

Algorithm 1 Self-learning framework

```
Input: $\mathcal{F}$ (source embeddings)
Input: $\mathcal{E}$ (target embeddings)
Input: $\mathcal{D}$ (seed dictionary)
Result: $\mathcal{W}$ (embedding mapping)
while not converge do
    $\mathcal{W} \leftarrow LEARN\_MAPPING(\mathcal{F}, \mathcal{E}, \mathcal{D})$
    $\mathcal{D} \leftarrow LEARN\_DICTIONARY$
end
```

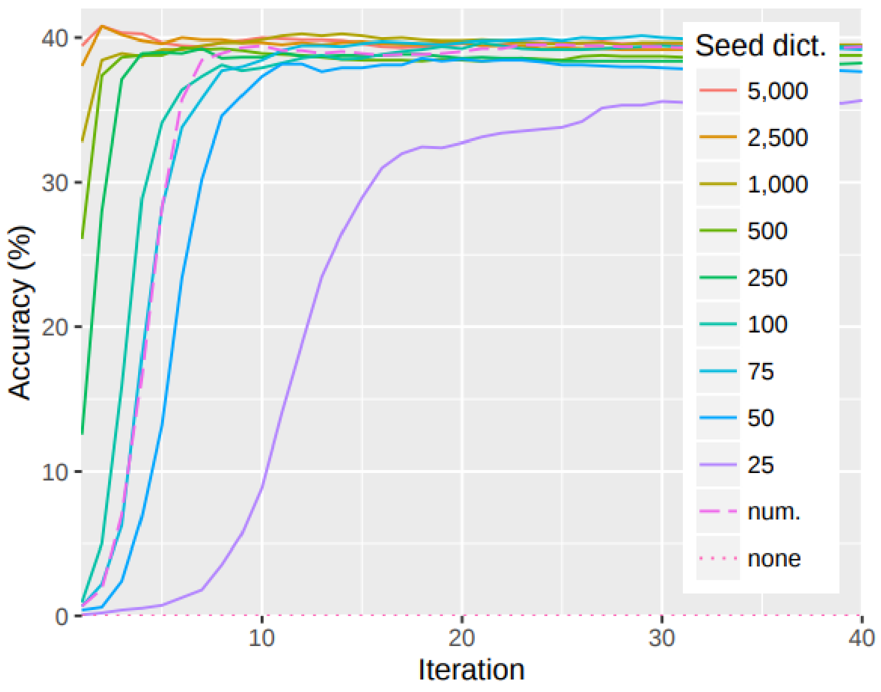

Figure 4.2: Accuracy on bilingual lexicon induction

$$\left.\begin{array}{ll}(f_1, & e_1)\\(f_2, & e_2)\\ & \vdots\\(f_N, & e_N)\end{array}\right\} \Rightarrow D$$

$$\mathcal{L} = \sum_{(f,e)\in D} \|Wf - e\|^2$$

4.3 Word Translation Induction

The CSLS as described by Conneau et al. [2017], can be written as:

$$CSLS(\boldsymbol{e}, \boldsymbol{f}) = 2cos(\boldsymbol{e}, \boldsymbol{y}) - \frac{1}{K} \sum_{\boldsymbol{e'} \in N(\boldsymbol{e})} cos(\boldsymbol{f}, \boldsymbol{e'}) - \frac{1}{K} \sum_{\boldsymbol{f'} \in N(\boldsymbol{e})} cos(\boldsymbol{f'}, \boldsymbol{e})$$

Since the embedding space is of high dimension and the nearest neighbour search is performed here, so that a few embeddings embedding will become the nearest neighbour of many data pointsto The obtained is known to suffer from the hubness problem, We denote $N_Y(x)$ the set of K nearest neighbors of points in the target embedding space, and $N_X(y)$ the nearest neighbors of y in the source embedding space. So in this way we penalize the hub points A good initialization is import for ICP methods, so begin with the projected data, and initialize transformation T_{ef} and T_{fe} with the identity matrix. After several iterations of, the estimated transformation becomes quite reliable. Then use this transformation to find the matches.

4.4 Non-linear Mapping

Chapter 5

Sentence Translation

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:

$$L(e; f, h) = \lambda_{emb} q(f, e) + \lambda_{LM} p(e|h)$$

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. $sigmoid$ for $softmax$

5.2 Denoising Autoencoder

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.

5.2.1 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.

5.2.2 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

5.2.3 Deletion Noise

The deletion noise is just a contrary case of insertion noise. Because different languages treat the prepositions or the articles differently. 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. 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 $p_i < p_{del}$, drop the word in the position i

Chapter 6

Experiments

6.1 Corpus Statistics

6.2 title

6.3 title

		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)

Table 6.1: Translation results on German↔English `newstest2016` and French↔English `newstest2014`.

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**
Lampe et al. [2017]	**13.3**	**9.6**	**14.3**	**15.1**
Artetxe et al. [2017b]	**-**	**-**	**15.6**	**15.1**

Table 6.2: 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		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**

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

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 6.2: 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

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.

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.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. A teacher-student framework for zero-resource neural machine translation. *arXiv preprint arXiv:1705.00753*, 2017.

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.

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

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

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.

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.

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

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.

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.

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.

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.

Ivan Vulic and Anna-Leena Korhonen. On the role of seed lexicons in learning bilingual word embeddings. 2016.

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.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 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.