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

27. März 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

LastName, Firstname
Name, Vorname

Number
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, 27. März 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, 27. März 2018
Ort. Datum

Unterschrift

Abstract

neural machine translation systems(NMT) beyond traditional statistical machine translation(SMT) in

This thesis start from unsupervised word translation,

Contents

Chapter 1

Introduction

Different from image and audio processing, objective can be the compound of sub-layer information like pixel value or ... 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. Actually maybe 'dog' may have some common activities description as 'cat'. If we can not capture such relationship, we need to run the algorithm on much more data or need human efforts like labeling or grammar analysis since representing word as unique, discrete ids leads to data sparsity. 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.

Word embedding attached more for natural language processing, for example sentiment analysis and machine translation in recent years.

1.1 Related Work

1.1.1 Word Embedding

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. [2017] 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 Outline

Chapter 2

Machine Translation

2.1 Rule-based machine translation

Rule-based machine translation is machine translation system based on linguistic information. An RBMT system generates translation based on different levels of analysis, e.g. part of speech tagging (POS), morphological analysis, semantic analysis, constituent analysis, dependency analysis.

Rule-based machine translation has the following advantages: No bilingual texts are required. Also because RBMT are built on a source language analysis and the target language generator and the source analysis part and target generation part are seperate, so it can be shared between multiple translation system if we replace the part with an other part for a closed related language. However the method is still not so general, we need to build the dictionary and linguistic rule set manually, it is very expensive. also it is hard to deal with rule interactions in big systems, ambiguity and idiomatic expression.

2.2 Statistical machine translation

The initial models for machine translation are based on words as units (Word-based machine translation), that can be translated, inserted, dropped and reordered. Fertility is the notion that input words produce a specific number of output words in the output language.

Define the phrase-based statistical machine translation model mathematically. First apply the Bayes rule to invert the translation direction and integrate a language model p_{LM} so the best English translation for the input sentence f is defined as

The advantages of the phrase-based machine translation is :

1. many-to-many translation can handle non-compositional phrase
2. better utilization of local context in translation
3. the more data, the longer phrases can be learned

Probabilistic Model

Bayes rule
translation model $p(\boldsymbol{e}|\boldsymbol{f})$, language model $p_{LM}(\boldsymbol{e})$

Weighted Model

Describe standard model consists of three sub-models:

1. phrase translation model
2. reordering model d
3. language model $p_{LM}(e)$

Phrase translation model can be learned based on a word alignment, extraction of phrase pairs and scoring phrase pairs or using the EM algorithm to learn the phrase table Reordering model can be distance based reordering model or lexicalized reordering, the lexicalized reordering model predicts the orientation of a phrase: either monotone, discontinuous or swap word alignment, phrase extraction, phrase scoring align phrase pairs directly with EM algorithm Add weights: λ_Φ, λ_d , λ_{LM}

Log-linear Model Combination

$$p(e, a|f) = exp[\lambda \sum_{i=1}^{I} \log \Phi(f_i|)]$$

Reordering is handled by a distance-based reordering model.

Different mode components in the phrase model are combined in a log-linear model, in which each component is a factor which maybe be weighted. The model cab be further extended by components like: bidirectional translation probabilities, lexical weighting, word penalty and phrase

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

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

extended language model for target word sequence

$$p(e_t|e_{t-1}h_{t-1}, f_1^J) = p(e_t|e_{t-1}, h_{t-1}, c_t)$$
$$\tilde{\boldsymbol{h_t}} = tanh(\boldsymbol{W_c}[\boldsymbol{c_t}; \boldsymbol{h_t}])$$
$$p(y_t|y_{<t}, x) = softmax(\boldsymbol{W_s}\tilde{\boldsymbol{h_t}})$$

2.4 Unsupervised Machine Translation

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.

2.4.1 Language Model

2.4.2 Back Translation

2.4.3 Shared Latent Representations

Unsupervised MT Learn language models P_s and P_t over source and target languages

Algorithm 1 How to write algorithms

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: $P_{s\to t}^{(0)}$ and $P_{t\to s}^{(0)}$

for $k = 1$ *to* N **do**

- **Backtranslation:** Generate source and target sentences using the current translation models $P_{t\to s}^{(k-1)}$ and $P_{s\to t}^{(k-1)}$, factoring in language models, P_s and P_t;
- **Train new translation models $P_{s\to t}^{(k)}$ and $P_{t\to s}^{(k)}$:** Use the generated sentences and leveraging P_s and P_t

end

Initial translation models:

Backtranslation: Generate source and target sentences using the current translation models $P_{t\to s}^{(k-1)}$ and $P_{s\to t}^{(k-1)}$, factoring in language models, P_s and P_t; Train new translation models $P_{s\to t}^{(k)}$ and $P_{t\to s}^{(k)}$ using the generated sentences and leveraging P_s and P_t

Chapter 3

Word Embedding

3.1 Monolingual Embedding

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 embeddings 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:

$$p(w_t|w_s) = \text{softmax}(s(w_t, w_s)) \tag{3.1}$$

$$= \frac{\exp\{s(w_t, w_s)\}}{\sum_{w' \in W} \exp\{s(w', w_s)\}} \tag{3.2}$$

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:

$$J_{ML} = \log p(w_t|w_s) \tag{3.3}$$

$$= s(w_t, w_s) - \log(\sum_{w' \in W} \exp\{s(w', w_s)\}) \tag{3.4}$$

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

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 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:

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

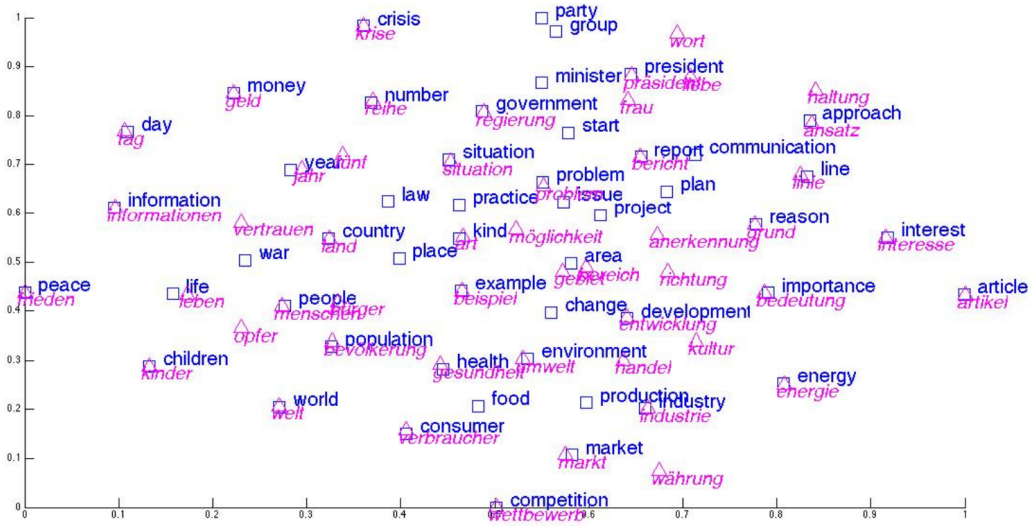


Figure 3.1: A cross-lingual embedding space between German and English

3.2 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.

3.2.1 Supervised Learning

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 Unsupervised Learning

However, large dictionary is also not readily available for many language pairs.
Several unsupervised learning algorithms are studies
Self-learning framework

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.

Unsupervised Initialization

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.

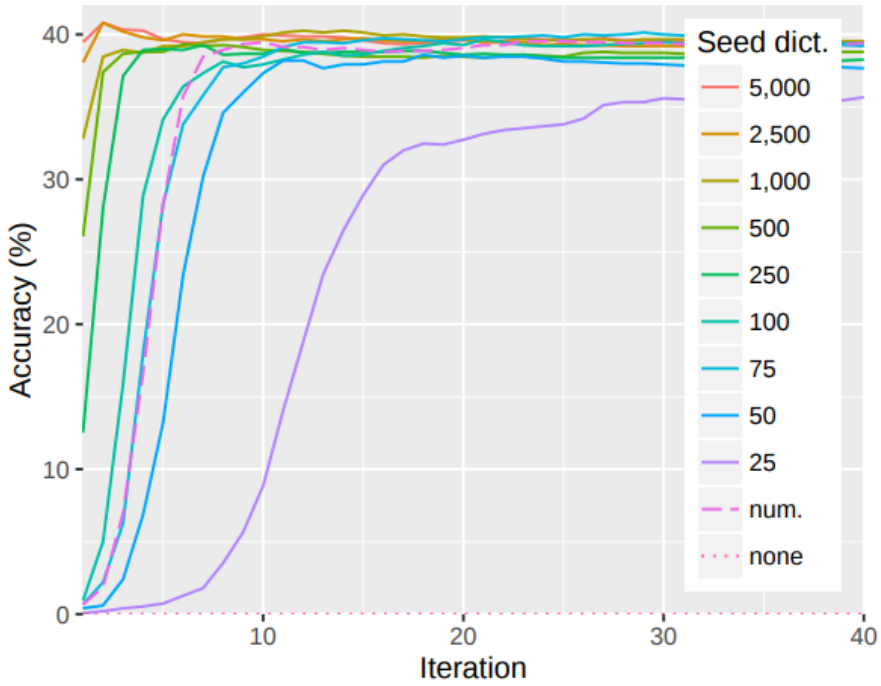


Figure 3.2: Accuracy on bilingual lexicon induction

Algorithm 2 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

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, 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

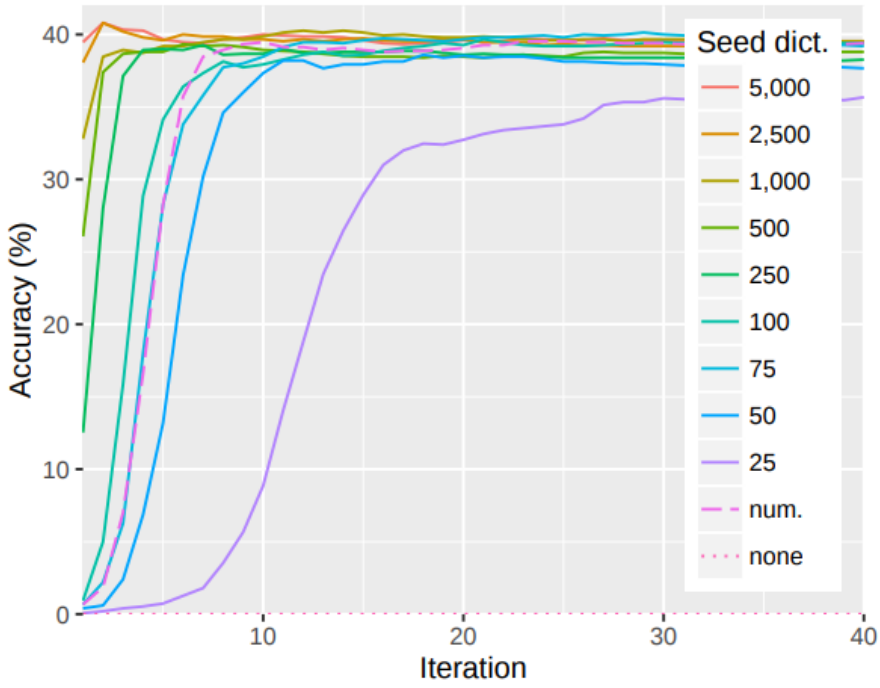


Figure 3.3: Accuracy on bilingual lexicon induction

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.

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:
$\sum_i \|e_i - W_{fe}f(e_i)\| + \sum_j \|f_j - W_{ef}e(f_j)\| + \lambda\sum_i\|e_i - W_{fe}W_{ef}e_i\| + \lambda\sum_j \|f_j - W_{ef}W_{fe}f_j\|$

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

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

3.2.3 Cross-domain Similarity Local Scaling (CSLS)

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.

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.

Chapter 4

Sentence Translation

4.1 Context-aware Beam Search

4.1.1 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.

4.1.2 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 $softmax$

4.2 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.

4.2.1 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.

4.2.2 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

4.2.3 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

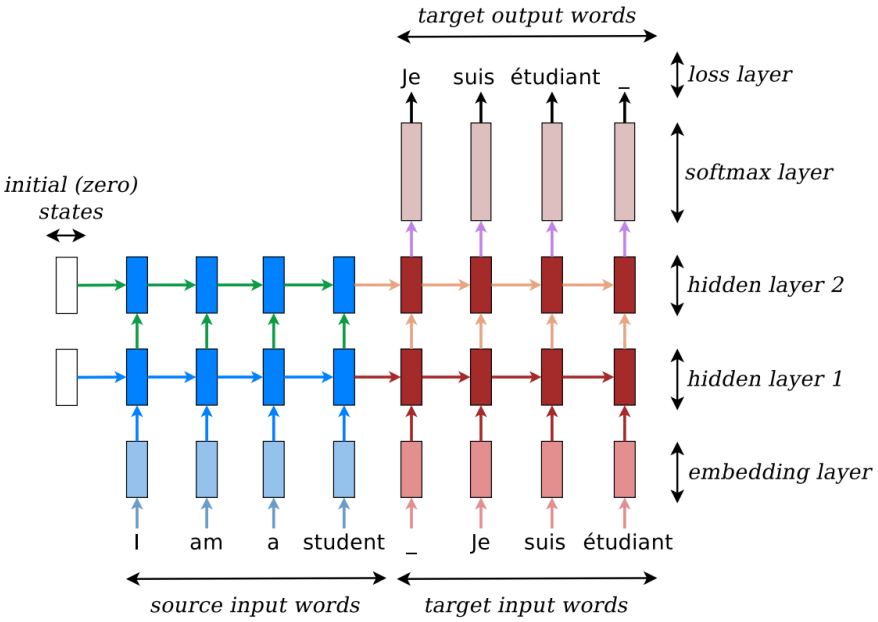
4.3 Neural Network

Seq2seq Model

Forget gate $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
Update gate
$z_t = \sigma(W_z x_t + U_z h_{t-1})$
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.

Attention Mechanism

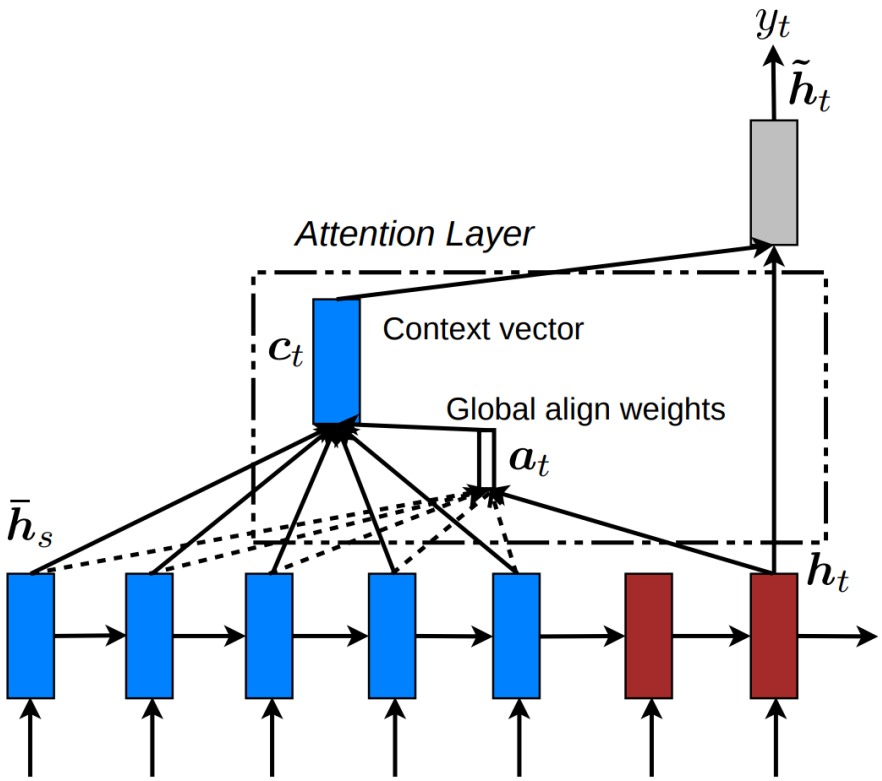
To solve the problem mentioned above, attention mechanism was proposed to derive a context vector c_t that capture the input information to help to predict the target word at time t. The basic idea is: given the target hidden state h_t and the source-side context vector c_t, we can compute the hidden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

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

Then the target word can be predicted by softmax function:

$$p(y_t|y_{\leq t,x}) = softmax(W_s\tilde{h_t})$$

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

$$a_t(s) = \text{align}(h_t, \bar{h}_s) \tag{4.1}$$

$$= \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} \exp(\text{score}(h_t, \bar{h}'_s))} \tag{4.2}$$

$$\text{score}(h_t, \bar{h}_s) = \begin{cases} {h_t}^T \bar{h}_s & dot \\ {h_t}^T W_a \bar{h}_s & general \\ {v_a}^T tanh(W_a[h_t; \bar{h}_s]) & concat \end{cases} \tag{4.3}$$

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 input 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_t for word at the time t. Then the context vector c_t is a weighted sum within the window $[p_t - D, p_t + D]$, D is selected empirically. The model predict the aligned position p_t as followed:

$$p_t = S \cdot \text{sigmoid}(v_p^T \tanh(W_p h_t))$$

W_p and v_p are the model parameters which will be learned to predict positions. To favor the words near position p_t, we place a Gaussian distribution which centered at p_t.

$$a_t(s) = \text{align}(h_t, \bar{h}_s)\text{exp}\Big(- \frac{(s - p_t)^2}{2\sigma^2}\Big)$$

Transformer

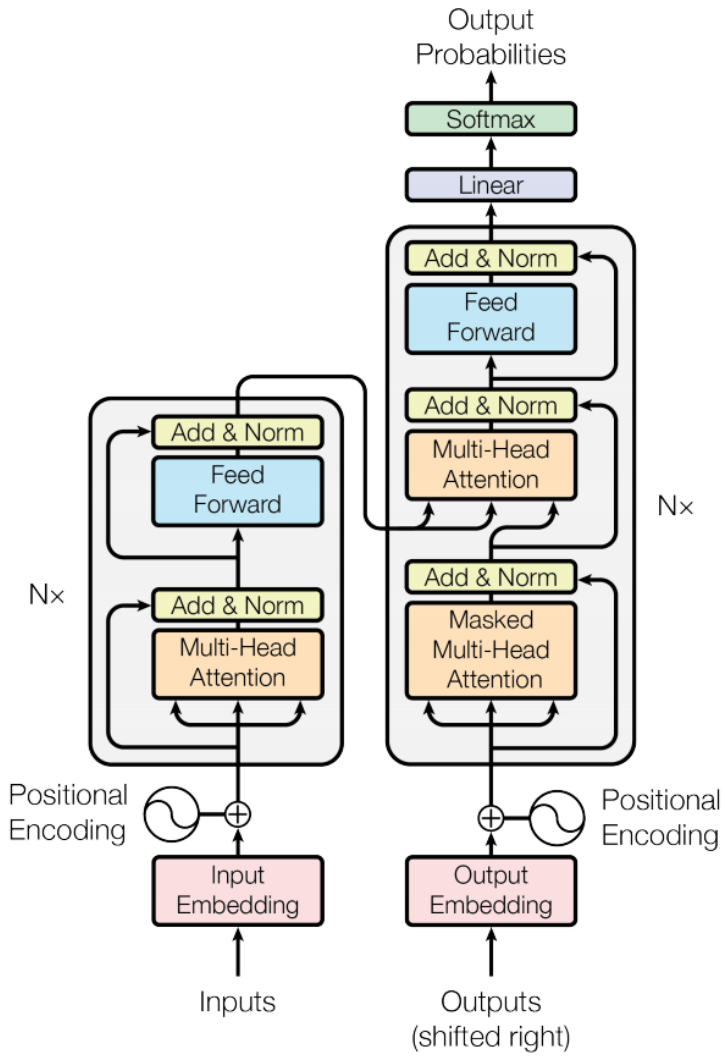
As the illustration above, the long-short term memory (LSTM) and gated recurrent units (GRU) have achieved state of the art in sequence modling and machine translation problem. However the RNN models also have some disadvantages, because of its sequential nature, it is more difficult to fully take advantage 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

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, 2017.

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.

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

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.

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.

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.