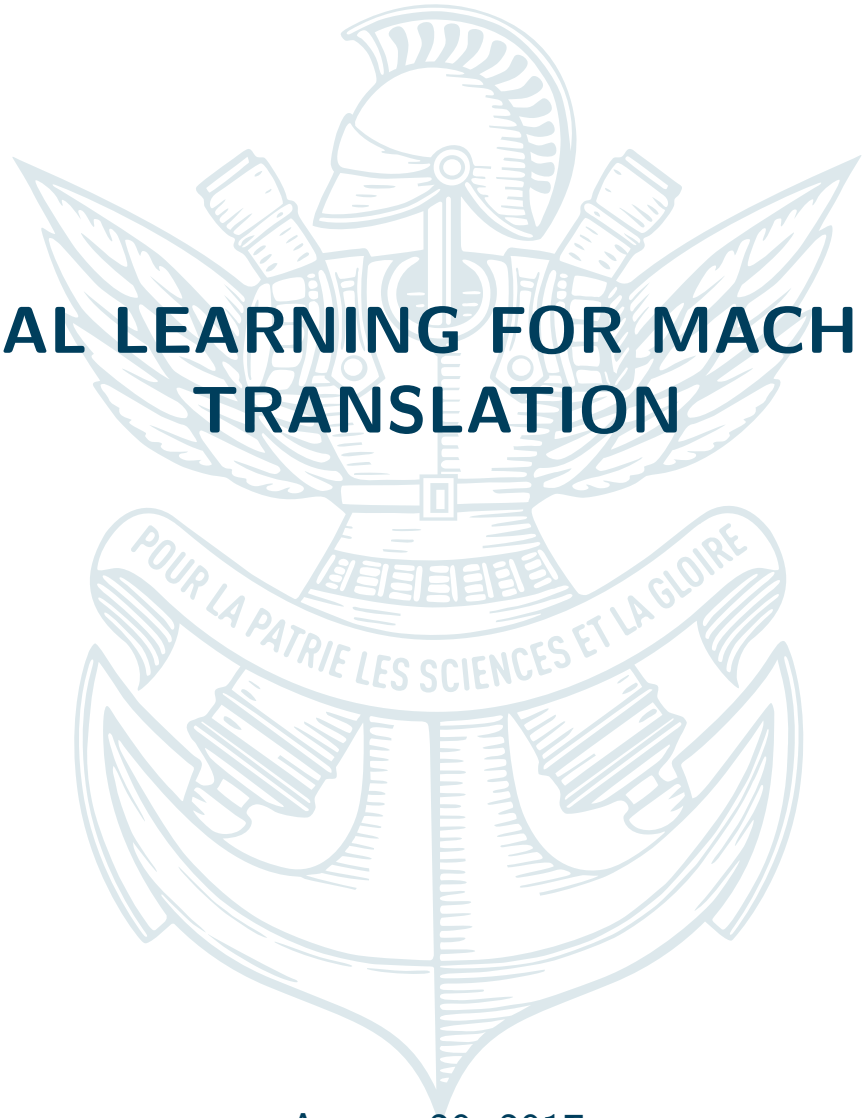
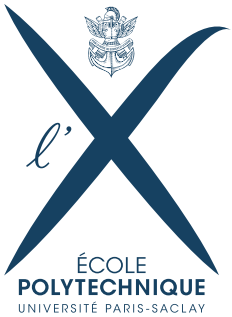


DUAL LEARNING FOR MACHINE TRANSLATION

August 20, 2017

MinhQuang PHAM - Ecole Polytechnique
Supervisor: François YVON
Place: Laboratoire Informatique pour Mécanique et Science d'Ingénieur (CNRS)



ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

I, the undersigned, do hereby certify:

1. That the results presented in this report are the result of my own work;
2. that I am the author of this report;
3. that I did not use any third-party source or result without explicitly mentioning it, using the standard bibliography rules.

Date: 10/08/2016
Signature preceded by the handwritten words *"I declare that this work could not be suspected of any plagiarism"*:

Contents

Introduction

1.1 Abstract

Recently, Neural Machine Translation has achieved state of the art performance in machine translation. However, the method still suffers from the shortage of training data. Several approaches aim to ease this problem by enlarging the training data by pseudo parallel corpora or by using language model in addition to translation model to render translations more fluently. My work in this internship is based on recent article "Dual learning on Machine Translation". The paper proposed a novel method involving the learning of 2 dual translations (for example: English to French and French to English) that help each other to improve their performances. The goal of my internship is to reimplement the method on an other data set, and to improve its performance by adjusting the policy gradient's estimation and rewards of reinforcement learning.

1.2 Context

Machine Translation is a sub field in Computational Linguistic that investigates in algorithms allowing computer to translate a text in one language to other languages. Machine Translation has been studied from many decades, probably since 1950. Today, Statistical Machine Translation has became most popular approach in the field. Statistical approaches aim to model the probability of a sentence in target language being a translation of a sentence in source language, then this distribution is used to infer best translation of any sentence in source language (i.e decoding or inference). Recently, among statistical models, Neural Machine Translation has achieved state of the art performance in Machine Translation thanks to it's robust modeling capacity. However like other statistical methods in Machine Translation, the Neural Network Machine Translation consumes large training data to make progress whereas this resource is quite limited.

The training data used in Statistical Machine Translation, and in Neural Machine Translation in particular, are essentially parallel corpora which are collections of sentence pairs consisting of a sentence in source language (i.e language from which we translate) and its translation in target language (i.e language to which we translate) (e.g "I like rice" in English and "J'aime du riz" in French). Parallel corpora are collected thanks to works of many researchers in the field of Computational Linguistic. Therefore, parallel corpora are limited in quantity and the collecting needs much work of translators. Moreover, the quality of collected corpora are hardly verified. Another difficulty is the effect of Out-of-Domain data. Large parallel corpora are usually domain-biased. Indeed, each parallel corpora often has a particular context and talks about one special theme. For example, European Parliament Proceedings or Canadian Hansards are well known parallel and they are all about parliamentary proceedings or political debates; another example is News Commentaries provided in Workshop on Statistical Machine Translation, this corpora consists of political and economic commentaries crawled from the website `https://www.project-syndicate.org`. The contains of different corpora

usually have different meaning, thus have different translation style and different vocabularies. These differences make Data-driven methods like Statistical Machine Translation biased to the training data. In order to make progression, some efforts take advantage of monolingual corpora.

Monolingual corpora are collections of text in only one language (e.g, collections of news in English, or collections of books in French). They are vastly collected and very rich in syntax and morphology. In deed, monolingual data is usually easier to obtain than parallel corpora, which are usually domain- or project-specific information, and can be harvested from many available sources, like manuals, technical documentations, marketing and sales information, etc. Because of the abundance of contexts and themes, monolingual corpora provide much knowledge about a language thus they are used to train language models which estimate probability of one word given words preceding it or words surrounding it (its context). Language model is an important part in Dual learning mechanism and will be described more in detail in following chapters.

The dual learning is mechanism for machine learning which involves 2 learning tasks in which output of one task can be used as training data for other task. For example, in Generative Adversarial Network, the score calculated by Discriminator Network is used to train the Generator Network and in reverse sample generated by Generator Network is used to train Discriminator Network. For Machine Translation, the situation can be described as the following two native speakers communication game:

1. The first native speaker, who only understands language A, sends a message in language A to the second native speaker who only understands language B, using a translation model which converts the message from language A to language B.

2. The second native speaker receives the translated message in language B. She checks the message and informs the first native speaker whether it is a natural sentence in language B. However, the second native speaker may not be able to verify the correctness of the translation since she does not receive, or even does not understand, the original message in language A. Then she sends the received message back to the first native speaker through another translation model, which converts the received message from language B back to language A.

3. After receiving the message from the second native speaker, the first native speaker checks it and notifies the second native speaker whether the message she receives is consistent with her original message. Through the feedback, both native speakers will know whether the two translation models perform well and can improve them accordingly.

4. The game can also be started from the second native speaker with an original message in language B, and then the two native speakers will go through a symmetric process and improve the two translation models according to the feedback.

It is easy to see from the above description, although the two native speakers may not have aligned bilingual corpora, they can still get feedback about the quality of the two translation

models and collectively improve the models based on the feedback. This game can be played for an arbitrary number of rounds, and the two translation models will get improved through this reinforcement procedure (e.g., by means of the policy gradient methods). In this way, we develop a general learning framework for training translation models through a dual-learning game.

1.3 Related work

The idea of using monolingual corpora in addition to parallel corpora in training translation models has been already studied in several papers. The approaches recently proposed could be considered to be 2 main paths.

In the first path, monolingual corpora in the target language are used to train language models, which are then integrated to the translation models trained from parallel bilingual corpora to improve the translation quality. This idea comes from the fact that a language model is trained to perceive the structure of an usual sentence in a certain language (statistically). Therefore, language model of target language can give good judgment of how good a sentence generated from translation model is in target language, hence a good indication to make model translate better. Several papers demonstrated improvement by using language model in addition to translation model; some of those are "Large Language Models in Machine Translation" of Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och and Jeffrey Dean of Google (cite here); and "On Using Monolingual Corpora in Neural Machine Translation" of C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio (cite here). The first paper performs log-linear model in which the mathematics of the problem were formalized by (Brown et al., 1993), and re-formulated by (Och and Ney, 2004) in terms of the optimization

$$\hat{e} = \arg\max_{e} \sum_{m=1}^{M} \lambda_m h_m(e, f)$$

(f is referred to source sentence and e is a candidate for the translation) where $\{h_m(e, f)\}$ is a set of M feature functions and $\{\lambda_m\}$a set of weights. Among feature functions, they used one or more feature functions of the form $h(e, f) = h(e)$ in which case it is referred to as a language model. The second deploys neural network structure (recurrent neural network in particular) for translation model and language model. The language model is then fused to translation model by 2 ways. In first way, they use language model in addition to score hypothesis (partial translation) in decoding process. At each time step, the translation model proposes a set of candidate words; the candidates are then scored according to the weighted sum of the scores given by the translation model and the language model. The second way fuses the hidden layer of each time step in recurrent network of model language in to output layer of recurrent network of translation model.

$$p(y_t|y_{<t}, x) \propto exp(y_t^T(W_o f_o(s_t^{LM}, s_t^{TM}, y_{t\hat{a}\check{L}\check{S}1}, c_t) + b_o))$$

where subscripts LM,TM denote the hidden states of the neural language model and neural translation model respectively. If the reader is not familiar to these notations and notions,

please go to Neural Machine Translation and Neural Language Model sections to have explanation in more detail.

In the second path, monolingual corpora are used to enlarge training bilingual corpora by producing pseudo bilingual sentence pairs. These pseudo bilingual sentence pairs are generated by using the translation model trained from parallel corpora to translate monolingual corpora of source language. Then pseudo bilingual pairs will be fused to ground-truth parallel corpora to make a large training data set for training translation model. These approaches may seem unrealistic but have shown positive results reported in some papers such as "Improving Neural Machine Translation Models with Monolingual Data" of Rico Sennrich, Barry Haddow and Alexandra Birch [7] and "Semi-supervised model adaptation for statistical machine translation" of N. Ueffing, G. Haffari, and A. Sarkar(cite here)

Out of translation task context, monolingual corpora are used to build embedding vectors representing words in 2 language. For example, in paper "BilBOWA: Fast Bilingual Distributed Representations without Word Alignments" (cite here) of Stephan Gouws; Yoshua Bengio and Greg Corrado, the authors use large monolingual corpora of 2 language in addition to a parallel corpora between those 2 language to infer a set of cross-lingual word embedding vector which are word representing vectors so that words having similar distributional syntactic and semantic properties in both languages are represented using similar vectorial representations (i.e. embed nearby in the embedded space). These vectors are demonstrated to be useful in document classification task or machine translation task(e.g "Bilingual Word Embeddings for Phrase-Based Machine Translation" of Will Y.Zou, Richard Socher, Daniel Cer and Christopher D. Manning (cite here)). In the extension of Dual Learning algorithm, I propose some adjustments in reward by adding distance between set of word embedding vectors of source sentence and set of word embedding vectors of predicted translation.

Beside the use of monolingual corpora, an other bootstrapping technique employed in Dual Learning mechanism is the duality of 2 translation tasks. Many task in machine learning can be coupled in sense of duality (i.e a couple of primal task and dual task) such as: Generator and Discriminator in Generative Adversarial Network; translating from language A to language B and vice versa; speech recognition and text to speech generating in Speech processing; search and query/keyword suggesting in Search engine. Dual learning mechanism is recently studied. One application of Dual Learning in Image translating task was recently studied in paper "DualGAN: Unsupervised Dual Learning for Image-to-Image Translation" of Zili Yi, Hao (Richard) Zhang, Ping Tan and Minglun Gong. The paper "Dual Supervised Learning" of Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu and Tie-Yan Liu applies Dual Learning mechanism in a supervised problem formulated from 2 dual tasks.

Dual learning

2.1 Mechanism, pseudo code and major elements

In the context where there are two different languages A and B (e.g English and French) and we want to build translation model to translate from language A to language B (i.e A is source language and B is target language), we define primal task translating from A to B and dual task translating from B to A; then we might define primal model translation model from language A to language B and dual model translation model from language B to language A. In the context of Dual learning, primal model and dual model are trained by applying the 2 native speakers communication game introduced in the Introduction part.

In traditional mechanism of machine translation, the mathematical approximation of learning problem is to maximize cross entropy $\frac{1}{N}\sum_{i=1}^{N} P(y_i|x_i)$ where x_i are source sentences and y_i are human translation (usually called references) of x_i in target language. Unfortunately, for unpopular languages we do not have enough data for this training style. To bootstrap, Dual learning proposes another objective function independent from references in target language. Dual learning replaces direct score $P(y|x)$ for translation model (primal or dual), which is derived from target sentence, by the expectation of reward for translations that it can predict. The reward r of a translation denoted by s_{mid} of sentence s in source/target language predicted by primal/dual model is a combination of 2 reference-independent scores which are the score evaluated by language model in target/source language denoted by $r_1 = P_{LM}(s_{mid})$ (that will be described in detail in following sections) and the log probability $r_2 = logP(x|s_{mid})$ given by dual/primal model (partner model): $r = \alpha * r_1 + (1 - \alpha) * r_2$ (will be denoted $R_{DL}(s_{mid})$ in the rest of report).

Intuitively, the former score indicates how a translation of primal/dual model is normal in the target/source language. This score is good information for training translation model because a well constructed sentence is highly possibly a good translation. Moreover, we can compute first score with high accuracy because we always have large resource for training a language model with high quality. The interesting point in Dual learning is leverage of the duality of 2 translation models. The second score marks the interaction between 2 models. The partner model helps the player model to improve it's translation performance based on it's "knowledge". The game of translation will be played by 2 translation models, each one will help its partner improve by it's current knowledge. Signal from dual model is possibly good training data for primal model to improve performance; thus primal model will give better feedback for dual model; hence they both get improvement. However, this scenario does not have any theoretical guarantee for improvement. If one model has too low performance, its signal are highly inaccurate and mislead the training of the other model. Therefore, there might be some thresholds for the beginning performances of 2 models and conditions for additional support models (e.g language models in case of translation task) to assure the improvement. Following is the pseudo code of dual learning algorithm which was proposed in article [8].

Procedure 1 The dual learning algorithm

Input: Monolingual corpora D_A,D_B, initial translation models Θ_{AB} and Θ_{BA}, language models LM_A and LM_B, hyperparamter α, beam search size K, learning rates $\gamma_{1,t}$ and $\gamma_{2,t}$

1: **repeat**
2: t = t + 1
3: Sample sentence s_A and s_B from D_A and D_B respectively
4: Set s =s_A
5: Generate K sentences $s_{mid,1}$, ..., $s_{mid,K}$ using beam search of size K according to translation model $P(. \mid s, \Theta_{AB})$
6:
7: **for** k=1 **to** K **do**
8: Set the language reward for the k-th sampled sentence as $r_{1,k} = LM_B(s_{mid,k})$.
Set the backward reward for the k-th sampled sentence as $r_{2,k} = P(s_A \mid s_{mid,k}, \Theta_{BA})$
Set total reward of the k-th sample as $r_k = \alpha r_{1,k} + (1-\alpha) r_{2,k}$.
9: **end for**
10: Compute the stochastic gradient of Θ_{AB}

$$\Delta_{\Theta_{AB}} \hat{\mathbb{E}}[r] = \frac{1}{K} \sum_{k=1}^{K} [r_k \Delta_{\Theta_{AB}} logP(s_{mid,k} \mid s; \Theta_{AB})]$$

11: Compute the stochastic gradient of Θ_{BA}

$$\Delta_{\Theta_{BA}} \hat{\mathbb{E}}[r] = \frac{1}{K} \sum_{k=1}^{K} [(1-\alpha) \Delta_{\Theta_{BA}} logP(s_{mid,k} \mid s; \Theta_{\Theta_{BA}})]$$

12: Model updates:

$$\Theta_{AB} \leftarrow \Theta_{AB} + \gamma_{1,t} \Delta_{\Theta_{AB}} \hat{\mathbb{E}}[r]$$
$$\Theta_{BA} \leftarrow \Theta_{BA} + \gamma_{2,t} \Delta_{\Theta_{BA}} \hat{\mathbb{E}}[r]$$

13: Set $s = s_B$.
14: Go through line 4 to line 12 similarly
15: **until** convergence

Inside the Dual learning for translation task, there are some important structures that play principal roles; they are Neural translation model and Neural Language Model. 2 next sections is provided to give details of their specific structures and also a brief summary on statistical machine translation that will help the reader become familiar to notions used in this report. One more point that we need to pay attention is that the objective of the game is to maximize the reward. The reward is final score given for a complete sequence of tokens (units forming sentence, and almost like words) sequentially predicted by translation model. Therefore, the problem can be closely formulated in form of a Reinforcement Learning problem, then the optimization will be performed by Policy gradient which is one of optimization method used in

Reinforcement Learning. After section of Neural Translation model, I can explain clearly how to formulate Reinforcement Learning problem and why we use Policy gradient as optimization method.

2.2 Neural Language Model

2.2.1 Statistical Language Model

I would like to start with a brief introduction of statistical language model to help the reader easily understand the way of reasoning and further structures. One essential component of any statistical machine translation system is the language model (particularly in target language), which measures how likely it is that a sequence of words in a certain language would be regular to a native speaker. Formally, a language model is a function that takes an sentence in language A and returns the probability of the sentence being produced by a native speaker in language A. It is obvious to see the benefits of such a model. We would like our machine translation system not only to produce output words that are true to the original in meaning, but also to make them close to fluent sentences in target language. In fact, the language model supports difficult decisions about word order and word translation. For instance, a probabilistic language model p_{LM} should prefer correct word order to incorrect word order:

$$p_{LM}(the\ house\ is\ small) > p_{LM}(small\ the\ is\ house)$$

It is more likely that an English speaker would utter the sentence "the house is small" than the sentence "small the is house". Hence, a good language model p_{LM} assigns a higher probability to the first sentence. This preference of the language model helps a statistical machine translation system to find the right word order.

Mathematically, language model p_{LM} assigns a probability distribution over all possible sequences of words such that those constructed by native speaker have probability larger than others. The problem is not easy because set of all possible sequences of words is infinite, a sequences of words can be as long as possible. Therefore, we want to restrict the problem to modeling probability of sentences. However, one sentence can be very long and very abundant. Fortunately, a probability of a sentence $y_1, ..., y_T$ can be decomposed as

$$P_{LM}(y_1, ..., y_T) = \prod_{i=1}^{T} P_{LM}(y_i | y_1, ..., y_{i-1})$$

We still want to limit problem to certain length k, so we might intuitively assume that sequence of words has Markov property.

$$P_{LM}(y_i | y_1, ..., y_{i-1}) = P_{LM}(y_i | y_{i-k}, ..., y_{i-1})$$

where conditional probability of a word only depends on a certain number of previous words. This is called n-gram language model.

To guide decisions when building a language model, we need a measure of a model's quality. If we have some held-out sample text, we can compute the probability that our language model assigns to it. A better language model should assign a higher probability to this text. Concretely, the most used evaluation of a language model is the perplexity computed on a text.

$$PP = 2^{-\frac{1}{n} log P(x_1,...,x_n)}$$

The lower perplexity the better language model.

2.2.2 Recurrent Neural Network Language Model

There are many ways to design a probability distribution as language model. The simplest one is counting

$$P_{LM}(y_i|y_1, ..., y_{i-1}) = \frac{Count(y_1, ..., y_{i-1}, y_i)}{Count(y_1, ..., y_{i-1})}$$

We might also find the solution among parameterized models. Recurrent Neural Network language model is one of those. This type of language model is based on Recurrent Neural Network structure which is known to be very suitable for sequence modeling.

In simplest form, Recurrent Neural Network has an input layer x, hidden layer s (also called context layer or state) and output layer y. At time step t, input to the network is denoted as $x(t)$, output of network is denoted as $y(t)$, and $s(t)$ denotes hidden state of the network. In order to present a word in mathematical form, we first define a vocabulary set containing certain number of words (also contains a "special word" named UNK reserved for words not included in the vocabulary), then we can present each word $x(t)$ by 1 of K encoding vector $x(t) \in R^{|V|}$ and

$$x(t)_j = 1_{j=i}$$

where i is index of $x(t)$ in vocabulary. Hidden and output layers are then computed recursively as follows:

$$s(t) = \sigma(R * s(t-1) + E * x(t))$$

$$y(t) = g(O * s(t))$$

where σ is sigmoid function and g is softmax function that outputs a probability distribution over the vocabulary. The model is parameterized by matrix $E \in R^{b \times |V|}$, $R \in R^{b \times b}$ and $O \in R^{|V| \times b}$ where b is size of hidden layer.

The Recurrent Neural Network language model then assigns the conditional distribution as following

$$P_{LM}(x_i|x_1, .., x_{i-1}) = y(i)_{v(x_i)}$$

where $y(i)$ is i-th output after feeding Recurrent Networks $x_1, ..., x_{i-1}$ and $v(x_i)$ is index of word x_i in the vocabulary.

Recurrent Neural Network category have 2 more complex structures: Gated Recurrent Unit and Long Short Term Memory. These structures still have 3 layers as simple Recurrent Neural Network, but are more complicated in the design of hidden layer where signals from the input and from previous hidden state have to go through special gates before arriving to current hidden state. In my implementation of Dual Learning, I use Gated Recurrent Unit Language Model. The structure can be found in paper "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling" of Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio (cite here).

2.3 Neural Translation Model

2.3.1 Statistical Translation Model

With a likely idea as language model, Statistical translation model aims to assign a conditional probability over couples of sentences consisting a source sentence in source language and a target sentence in target language such that given a source sentence, target sentence has same meaning as source sentence have more probability than others. For example

$$P_{TM}(Je\,t'aime \mid I\,love\,you) > P_{TM}(Je\,t'écoute \mid I\,love\,you)$$

2.3.2 Recurrent Neural Network Translation Model

Recurrent Neural Network Translation Model uses encoding-decoding structure to model the probability of couple source-target sentences. The model consists of 2 major parts: encoder and decoder. Encoder cumulatively enrolls the syntactic and semantic information of source sentence into hidden states as same as what language model does to encode a sequence of words. Decoder part will use these hidden states to assign conditional probability $P_{TM}(y_i|y_1, ..., y_{i-1}, x_{1,...,T})$ recursively.

In more detail, I will describe briefly the structure that I implemented in my work. This structure was originally presented in paper "Neural Machine Translation by jointly learning to align and translate" of Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio.

The input to encoder is source sentence denoted $x_{1,...T}$. Each word is presented by 1 of K coded word vector $xi \in R^{|V_s|}$ where V_s is vocabulary of source language. The encoder uses bidirectional Recurrent Neural Network structure where hidden states h_i is composed of forward hidden state $\overrightarrow{h}_i$ and backward hidden state $\overleftarrow{h}_i$:

$$h_i = \begin{pmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{pmatrix}$$

The forward hidden states are recursively computed:

$$\overrightarrow{h}_i = \begin{cases} 0, & \text{if } i = 0 \\ (1 - \overrightarrow{z}_i) \odot \overrightarrow{h}_{i-1} + \overrightarrow{z}_i \odot \overrightarrow{\underline{h}}_i, & \text{otherwise} \end{cases} \tag{2.1}$$

Where proposal hidden state $\overrightarrow{\underline{h}}_i$ is

$$\overrightarrow{\underline{h}}_i = tanh(\overrightarrow{W} \times \bar{E} \times x_i + \overrightarrow{U} \times \left[\overrightarrow{r}_i \odot \overrightarrow{h}_{i-1}\right])$$

And additional gates $\overrightarrow{z}_i$, $\overrightarrow{r}_i$ are:

$$\overrightarrow{z}_i = \sigma(\overrightarrow{W}_z \times \bar{E} \times x_i + \overrightarrow{U}_z \times \overrightarrow{h}_{i-1})$$

$$\overrightarrow{r}_i = \sigma(\overrightarrow{W}_r \times \bar{E} \times x_i + \overrightarrow{U}_r \times \overrightarrow{h}_{i-1})$$

Where $\bar{E} \in R^{m*|V_{src}|}$ is the word embedding matrix; $\overrightarrow{W}$,$\overrightarrow{W}_z$,$\overrightarrow{W}_r \in R^{n\times m}$;$\overrightarrow{U}$,$\overrightarrow{U}_z$,$\overrightarrow{U}_r \in R^{n\times n}$ are weight matrices (m is size of word embedding vector in source language, n is size of hidden state of encoder)

Similarly, the backward hidden states are computed with reversed order sequence $x_{T,..,1}$ and also are under-scripted from T to 1. Therefore, hidden state h_i captures information of $x_{1,..,i}$ and $x_{T,..,i}$.

These hidden states will be fed to decoder in order to compute conditional probability $P(y_i|x_{1,..,T}, y_1, .., y_{i-1})$. For each time step i, the hidden state of decoder s_i is recursively calculated by previous state and current word of target sentence y_i:

$$s_i = (1 - z_i) \odot s_{i-1} + z_i \odot \underline{s}_i$$

Where proposal state:

$$\underline{s}_i = tanh(W \times E \times y_i + U \times [r_i \odot s_{i-1}] + C \times c_i])$$

And additional gates z_i and r_i are:

$$z_i = (W_z \times E \times y_i + U_z \times s_{i-1} + C_z \times c_i)$$

$$r_i = (W_r \times E \times y_i + U_r \times s_{i-1} + C_r \times c_i)$$

Where $E \in R^{m\times|V_{trg}|}$; $W, W_z, W_r \in R^{n\times m}$; $U, U_z, U_r \in R^{n\times n}$; $C, C_z, C_r \in R^{n\times 2n}$ (m is size of word embedding vector in target language, n is size of hidden state of decoder).
The term c_i appearing in above equations is called context vector which is combination of hidden states h_i provided by encoder. The weight of h_i in c_i will be assigned as following:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j$$

where

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})}$$

and

$$e_{ij} = v_a^T tanh(W_a \times s_{i-1} + U_a \times h_j)$$

where $v_a^T \in R^{n_a}$; $W_a \in R^{n_a \times n}$,$U_a \in R^{n_a \times 2n}$ After obtaining hidden state s_i and context vector c_i, the conditional probability $P(y_i|x_{1,..,T}, y_1, .., y_{i-1})$ will be computed as:

$$P(y_i|x_{1,..,T}, y_1, .., y_{i-1}) \propto exp(y_i^T \times W_o \times t_i)$$

where

$$t_i = \left[max(\tilde{t}_{i;2j-1}; \tilde{t}_{i;2j})\right]_{j=1;...;l}^{T}$$

$$\tilde{t}_i = U_o \times s_{i-1} + V_o \times E \times y_{i-1} + C_o \times c_i$$

And $U_o \in R^{2l \times n}$, $V_o \in R^{2l \times m}$, $C_o \in R^{2l \times 2n}$, $W_o \in R^{|V_{trg}|}$

2.4 Optimization method

Authors of the paper "Dual Learning" found that the reward of the game can be considered as a function of s, s_{mid} and translation models θ_{AB} and θ_{BA}, so that the parameters of two translation model could be optimized through gradient method for reward maximization. Given source sentence s, the translation s_{mid} is sampled from distribution $P_{TM}(s_{mid}|s, \theta)$. Therefore, the parameters should be optimized to maximize the expectation of reward:

$$\rho(\theta, s) = E_{s_{mid} \sim P_{TM}(s_{mid}|s,\theta)} \left[r_{DL}(s_{mid})\right]$$

The gradient of $\rho(\theta, s)$ can be obtained as following:

$$\begin{aligned}
\frac{\partial \rho(\theta, s)}{\partial \theta} &= \frac{E_{s_{mid} \sim P_{TM}(s_{mid}|s,\theta)} \left[r_{DL}(s_{mid})\right]}{\partial \theta} \\
&= \sum_{s_{mid}} \frac{P_{TM}(s_{mid}|s, \theta) r_{DL}(s_{mid})}{\partial \theta} \\
&= \sum_{s_{mid}} \frac{P_{TM}(s_{mid}|s, \theta)}{\partial \theta} r_{DL}(s_{mid}) \\
&= \sum_{s_{mid}} P_{TM}(s_{mid}|s, \theta) \frac{\partial log P_{TM}(s_{mid}|s, \theta)}{\partial \theta} r_{DL}(s_{mid}) \\
&= E_{s_{mid} \sim P_{TM}(s_{mid}|s,\theta)} \left[\frac{\partial log P_{TM}(s_{mid}|s, \theta)}{\partial \theta} r_{DL}(s_{mid})\right]
\end{aligned} \tag{2.2}$$

This gradient could be approximated by sampling method as described in the pseudo code:

$$\Delta_{\Theta_{AB}} \hat{\mathbb{E}}[r] = \frac{1}{K} \sum_{k=1}^{K} [r_k \Delta_{\Theta_{AB}} log P(s_{mid,k} \mid s; \Theta_{AB})]$$

In order to increase the gain of game, the optimal search should following this gradient. However, this estimation is not stable and still not exploits astruces in Reinforcement Learning. We may see a better definition and approach in next part of the report.

Personal works

3.1 Analyses of the problem

The 2-agent game of Dual Learning can be decomposed into 2 actions:

1) The player translation model(primal or dual) takes decisions by choosing words to build a sentence that maximizes the reward received for its complete prediction.

2) The partner translation model(dual or primal) uses the pseudo translation predicted by player model to optimize it's own parameters.

The second action could be reasonable according to the work of Senrich in "Improving Neural Machine Translation Models with Monolingual Data" while pseudo parallel corpora actually helps improve performance of translation model.

The first action is indeed the game that we want to win. Therefore, it is better to organize it more systematically in terms of mathematics. Following, I would like to give a brief description about the context of a Reinforcement Learning problem and then redefine the optimization problem of Dual Learning in form of a Reinforcement Learning problem.

3.1.1 Reinforcement Learning problem's context

Reinforcement Learning studies problems of sequential decision making where each previous decision has affect on situation where next decision is chosen. Concretely, there are 3 main elements in a Reinforcement Learning problems: state(i.e, situation), action(i.e, situation where action is decided) and signal(i.e reward for chosen action) from environment. The decision maker and environment interact at each of a sequence of discrete time steps, $t = 0, 1, 2, 3, ...$ At each time step t, the decision maker receives some representation of the environment's state, $S_t \in S$, where S is the set of possible states, and on that basis selects an action, $A_t \in A(S_t)$, where $A(S_t)$ is the set of actions available in state S_t. One time step later, in part as a consequence of its action, the decision maker receives a numerical reward, $R_{t+1} \in \mathbb{R}$, and finds itself in a new state, S_{t+1}.

At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's policy and is denoted π_t, where $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent's goal, roughly speaking, is to maximize the total amount of reward it receives over the long run. If the sequence of rewards received after time step t is denoted $R_{t+1}, R_{t+2}, R_{t+3}, ...$, in general we seek to maximize the expected return, where the return, G_t, is defined as some special function of the reward sequence. In the simplest case the return is the sum of the rewards:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T$$

Where T is final step where decision maker reaches terminal state (i.e $S_T = terminal$) where the game terminates and decision maker continues in a new episode.

Consider how a general environment might respond at time t+1 to the action taken at time t. In the most general, causal case this response may depend on everything that has happened earlier. In this case the dynamics can be defined only by specifying the complete probability distribution:

$$P(R_{t+1} = r; S_{t+1} = s_0 | S_0; A_0; R_1; ...; S_{t-1}; A_{t-1}; R_t; S_t; A_t)$$

for all r, s_0, and all possible values of the past events: $S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t$.

If the state signal has the Markov property, on the other hand, then the environment's response at t+1 depends only on the state and action representations at t, in which case the environment's dynamics can be defined by specifying only

$$P(R_{t+1} = r; S_{t+1} = s_0 | S_t; A_t)$$

for all r, $s0$, S_t, and A_t

3.1.2 A better description

Recurrent Neural Network Translation Model generates(or decode) a translation y, given a source sentence x, by sequentially choosing words y_i based on a recursively computed conditional probability $P_{TM}(y_i|y_{1,...,i-1}, x)$. We could consider these conditional probabilities as policy of the translation model where prefix $y_{1,...,i-1}$ are states and words y_i are actions (i.e, set of possible actions is the vocabulary of target language of translation model). In addition, we suppose terminal states are all prefix ending with special word "End of sentence" denoted EOS that indicates the complete construction of one sentence and the game always begins from one state - a prefix containing only one special "Begin of Sentence" denoted BOS. Therefore, the game that translation model aims to win is to choose word sequentially knowing prefix consisting of word chosen up to moment such that the final construction receives largest score given by the language model of target language and the partner translation model. The context of this game also has Markov property where

$$\begin{aligned} P(R_{t+1} = r; S_{t+1} = s_0 | S_0; A_0; R_1; ...; S_{t-1}; A_{t-1}; R_t; S_t; A_t) &= P(R_{t+1} = r; S_{t+1} = s_0 | S_t; A_t) \\ &= 1_{r=R(S_t+A_t)} * 1_{s_0=S_t+A_t} \end{aligned} \tag{3.1}$$

Where $S_t + A_t$ is concatenation of word A_t and prefix S_t where A_t follows the end of S_t and $R(S_t + A_t)$ is deterministic reward given for partial construction of sentence $S_t + A_t$.

The reward and the next state following each action are deterministic. There 2 scenarios to give rewards such that the total reward that the player model receives when it finishing the construction is equal to score of constructed sentence given by the language model of target

language and the partner translation model:

1) We do not give any reward until the end of the game. Therefore,

$$R(A_i|S_i) = \begin{cases} R_{DL}(S_i + A_i), & \text{if } A_i = EOS \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

2) We could break the score $R_DL(s)$ into pieces and gives for each partial construction S_i.

$$R(A_i|S_i) = \begin{cases} R_{DL}(A_1) & \text{if } i = 0 \\ R_{DL}(S_i + A_i) - R_{DL}(S_i), & \text{otherwise} \end{cases} \tag{3.3}$$

The second strategy give reward for each action immediately, thus can help optimization process converge faster. In my experiments, i implement optimization algorithm for Reinforcement Learning with the second strategy.

3.2 Adjustment

3.2.1 Policy gradient with Q-value's approximation

Obviously, at each step i, the policy for choosing a word is conditional probability $P(y_i|y_{1,\dots i-1})$ parameterized by parameters of recurrent model. In reinforcement learning, Policy gradient is one approach which approximate a stochastic policy directly using an independent function approximator with its own parameters. In our problem, the policy is represented by a neural network whose input is a representation of a state and whose output is action selection probability. The parameters of neural network policy is updated following the gradient of the expected of total reward.

The expected of total reward is defined as:

$$\rho(\pi) = \mathbb{E}\left[\sum_{i=1}^{T} r_i|\pi\right]$$

We define also the value of action-state pairs given a policy:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{i=t+1}^{T} r_i|\pi, s_i = s, a_i = a\right]$$

The value of action state is independent of step where action is taken and state is given. The gradient of ρ will be calculated as following:

$$\begin{aligned}
\frac{\partial \rho}{\partial \theta} &= \frac{\partial}{\partial \theta}\mathbb{E}_{s_{0,..,T},a_{0,..,T}\sim\pi}\sum_{i=1}^{T+1} r_i \\
&= \frac{\partial}{\partial \theta}\sum_{s_{0,..,T},a_{0,..,T}} P(s_{0,..,T},a_{0,..,T}|\pi)\sum_{i=1}^{T+1} r_i \\
&= \sum_{s_{0,..,T},a_{0,..,T}} \frac{\partial}{\partial \theta}P(s_{1,..,T},a_{1,..,T}|\pi)\sum_{i=1}^{T+1} r_i \\
&= \sum_{s_{0,..,T},a_{0,..,T}} \frac{\partial}{\partial \theta}\prod_{j=1}^{T} P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\sum_{i=1}^{T+1} r_i \qquad (3.4)\\
&= \sum_{s_{0,..,T},a_{0,..,T}} P(s_{0,..,T},a_{0,..,T}|\pi)\sum_{j=1}^{T}\frac{\partial}{\partial \theta}logP(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\sum_{i=1}^{T+1} r_i \\
&= \sum_{s_{0,..,T},a_{0,..,T}} P(s_{0,..,T},a_{0,..,T}|\pi)\sum_{j=1}^{T}\frac{\partial}{\partial \theta}logP(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\sum_{i=j+1}^{T+1} r_i+ \\
&+ \sum_{s_{0,..,T},a_{0,..,T}} P(s_{0,..,T},a_{0,..,T}|\pi)\sum_{j=1}^{T}\frac{\partial}{\partial \theta}logP(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\sum_{i=1}^{j} r_i \qquad (3.5)
\end{aligned}$$

We can prove that second term of left side of above equation is 0. Indeed,

$$\begin{aligned}
SecondTerm &= \sum_{s_{0,..,T},a_{0,..,T}} \frac{P(s_{0,..,T},a_{0,..,T})}{P(s_j,a_j|s_{0,..,j-1},a_{0,..,j-1},\pi)}\times \\
&\times\sum_{j=1}^{T}\frac{\partial}{\partial \theta}P(s_j,a_j|s_{0,..,j-1},a_{0,..,j-1},\pi)\sum_{i=1}^{j} r_i \\
&= \sum_{s_{0,..,T},a_{0,..,T}} P(s_{0,..,j-1},a_{0,..,j-1}|\pi)P(s_{j+1,..,T},a_{j+1,..,T}|s_{0,..,j},a_{0,..,j})\times \\
&\times\sum_{j=1}^{T}\frac{\partial}{\partial \theta}P(s_j,a_j|s_{0,..,j-1},a_{0,..,j-1},\pi)\sum_{i=1}^{j} r_i \qquad (3.6)\\
&= \sum_{s_{0,..,j-1},a_{0,..,j-1}}\sum_{i=1}^{j} r_i\times \\
&\times\sum_{s_{j,..,T},a_{j,..,T}}\frac{\partial}{\partial \theta}P(s_j,a_j|s_{0,..,j-1},a_{0,..,j-1},\pi)P(s_{j+1,..,T},a_{j+1,..,T}|s_{0,..,j},a_{0,..,j}) \\
&= \sum_{s_{0,..,j-1},a_{0,..,j-1}}\sum_{i=1}^{j} r_i\times\sum_{s_j,a_j}\frac{\partial}{\partial \theta}P(s_j,a_j|s_{0,..,j-1},a_{0,..,j-1},\pi) \\
&= 0
\end{aligned}$$

Because $\sum_{s_j,a_j} P(s_j,a_j|s_{0,..,j-1},a_{0,..,j-1},\pi)=1$.
Therefore

$$\begin{aligned}
\frac{\partial\rho}{\partial\theta} &= \sum_{s_{0,..,T},a_{0,..,T}} P(s_{0,..,T},a_{0,..,T}|\pi)\sum_{j}^{T}\frac{\partial}{\partial\theta}logP(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\sum_{i=j+1}^{T+1} r_i \\
&= \sum_{s_{0,..,T},a_{0,..,T}} P(s_{0,..,j-1},a_{0,..,j-1}|\pi)P(s_{j+1,..,T},a_{j+1,..,T}|s_{0,..,j},a_{0,..,j})\times \\
&\times\sum_{j}^{T}\frac{\partial}{\partial\theta}P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\sum_{i=j+1}^{T+1} r_i \\
&= \sum_{s_{0,..,j-1},a_{0,..,j-1}} P(s_{0,..,j-1},a_{0,..,j-1}|\pi)\times\sum_{s_j,a_j}\frac{\partial}{\partial\theta}P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi) \\
&\times \sum_{s_{j+1,..,T},a_{j+1,..,T}} P(s_{j+1,..,T},a_{j+1,..,T}|s_{0,..,j},a_{0,..,j})\sum_{i=j+1}^{T+1} r_i \\
&= \sum_{s_{0,..,j-1},a_{0,..,j-1}} P(s_{0,..,j-1},a_{0,..,j-1}|\pi)\times\sum_{s_j,a_j}\frac{\partial}{\partial\theta}P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\times Q^{\pi}(s_j,a_j)
\end{aligned} \tag{3.7}$$

In this formulation, the gradient of reward function depends on function $Q^{\pi}(s,a)$ which is very difficult to be estimated in our case because of large space of state (set of all possible prefix). There are several approaches to estimate this function:

1)we can estimate $Q^{\pi}(s,a)$ by using the actual return $R_t=\sum_{i=j+1}^{T} r_i$. Or we can sampling from prefix $s_{1,..,j}$ and then assign the average of returns to $Q^{\pi}(s_j,a_j)$.

2)$Q^{\pi}(s,a)$ can be approximated by a function $f_\omega:\mathcal{S}\times\mathcal{A}\to\mathcal{R}$ which minimizes $E\left[\hat{Q}\pi(s_t,a_t)-f_\omega(s_t,a_t)\right]$ by following estimated gradient $\frac{\partial}{\partial\omega}\left[\hat{Q}^{\pi}(s_t,a_t)-f_\omega(s_t,a_t)\right]^2$ where $\hat{Q}^{\pi}(s_t,a_t)$ is some unbiased estimator of $Q^{\pi}(s_t,a_t)$ such as R_t.

The formulation of policy gradient has an advantage is that there is no term $\frac{\partial}{\partial\theta}P(s_{0,..,j-1},a_{0,..,j-1}|\pi)$, thus we can estimate the gradient by sampling. In comparison to the estimated gradient that we obtained in previous section, this formulation has less variance. Indeed, the term $r(s_{mid})$ in $E_{s_{mid}\sim P_{TM}(s_{mid}|s,\theta)}\left[\frac{\partial logP_{TM}(s_{mid}|s,\theta)}{\partial\theta}r_{DL}(s_{mid})\right]$ is total reward accumulated from beginning state BOS, thus is more variant than term $Q^{\pi}(s_t,a_t)$ where all term before t are canceled out.

We still have a problem caused by large size of vocabulary. In the formulation of policy gradient, the term $\sum_{s_j,a_j}\frac{\partial}{\partial\theta}P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\times Q^{\pi}(s_j,a_j)$ need at least $T\times\mid V\mid$ computations of $P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\times Q^{\pi}(s_j,a_j)$ where $\mid V\mid$ is vocabulary of target language's size. To reduce the complexity, i propose an estimation of this term by sampling.

$$\sum_{s_j,a_j}\frac{\partial}{\partial\theta}P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\times Q^{\pi}(s_j,a_j)=$$

$$= \sum_{s_j,a_j} P(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi)\frac{\partial}{\partial\theta}logP(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi) \times Q^\pi(s_j,a_j)$$

$$= E\left[logP(s_j,a_j|s_{1,..,j-1},a_{1,..,j-1},\pi) \times Q^\pi(s_j,a_j)\right]$$

$Q^\pi(s_j,a_j)$ will be estimated by sampling.

Compared to original paper, to adjust the estimation of policy gradient, i propose another estimation based on above analyses. In each round, for a source sentence, k predicted translation s^i_{mid} are sampled from current model. Then for each predicted target sentence $s^i_{mid} = [y_1,...,y_{T_i}]$, i consider T_i prefix $[y_1,..,y_t]_{|t\in\{1,..,T_i\}}$; given each prefix $[y_1,..,y_t]$, a word a_{t+1} is sampled extending the prefix to $[y_1,..,y_t,a_{t+1}]$ from which a complete target sentence s_t is constructed by sampling. According to second strategy of giving reward, $Q^\pi([y_1,..,y_t],a_{t+1})$ will be estimated by $\hat{Q}^\pi([y_1,..,y_t],a_{t+1}) = \sum_{i=t+1}^{l(s_t)} r_i = \sum_{i=t+1}^{l(s_t)} R_{DL}(y_1,...,y_i) - R_{DL}(y_1,...,y_{i-1}) = R_{DL}(s_t) - R_{DL}([y_1,..,y_t])$ (where $l(s)$ is length of sequence s), and the policy gradient will be estimated as following:

$$\frac{\partial}{\partial\theta}\rho(\theta) \approx \frac{1}{k}\sum_{i=1}^{k}\frac{1}{T_i}\sum_{t=0}^{T_i-1}\frac{\partial}{\partial\theta}logP(a_{t+1}|y_{1,..,t},\pi)\hat{Q}^\pi([y_1,..,y_t],a_{t+1})$$

3.2.2 Adjustment of reward

The reward is only indication to improve the performance of translation model. The reward for each sentence s_{mid} constructed by primal/dual translation model given source sentence x is originally supposed to be a combination of log of probability of sentence given by language model of target/source language and log of probability $P(x|s_{mid})$ given by dual/primal translation model. The first term of the combination evaluates how fluent the translation is in target language whereas the second one evaluates how well the meaning is preserved according to the partner translation model.

Obviously, the quality of second term depends on the quality of the partner model. This score is objective and affected by partner model. Therefore, in my project, I search also to find another factor that evaluates also the correspondence between the meaning of the original sentence and the meaning of a translation predicted by translation model. There is a simple way to check the correspondence that is to check how words of 2 sentences are related to each other. For example, it is generally required that each word in translation have close meaning to at least one word in source sentence. If we have a dictionary, we can verify this easily. We can count the matches between 2 sentences and put the fraction of this number over length of sentence in terms of reward. Another approach that I find interesting to try is to exploit the bilingual word embedding estimated by 2 monolingual corpora and bilingual corpus. The following subsection will describe briefly how they are estimated by using 2 large monolingual corpora of source language and target language and one parallel corpus between these two languages.

3.2.3 Bilingual word embedding

Word embedding is one part of recurrent neural network language model where each word in source sentence is projected to a vector. In recurrent language model, hidden state at time step t is computed from t-th word in source sentence and hidden state at time step t-1:

$$s(t) = \sigma(R * s(t-1) + E * x(t))$$

And the output conditional probability :

$$y(t) = g(O * s(t))$$

In this framework, the word representations are found in the columns of matrix E, with each column representing a word. The model itself has no use of assumption on syntax, morphology or semantics. However, training such model to maximize likelihood $\sum_{y_{tg}} \sum_t y(t)(y_{tg}(t))$ over a large monolingual corpus will induce word representations with remarkable syntactic and semantic properties.

The bilingual word embedding aims to compute the representations of words in both language so that similar words in each language have similar embedding and additionally similar words across languages have similar representations. To achieve both goals, the objective of training problem could be designed as combination of a monolingual objective and a cross-lingual objective such as following:

$$\mathcal{L} = min_{\theta^{src},\theta^{trg}} \sum_{l \in \{src,trg\}} \sum_{\omega,h \in D^l} \mathcal{L}(\omega, h; \theta^l) + \lambda \Omega(\theta^{src}, \theta^{trg})$$

Where $\sum_{\omega,h \in D^l} \mathcal{L}(\omega, h; \theta^l)$ is cross entropy over data set for language l; and $\Omega(\theta^e, \theta^l)$ is penalization term imposing the cross-lingual relation on word embedding vectors.

In general bilingual setting, the cross-lingual objective is as following:

$$\Omega(\theta^{src}, \theta^{trg}) = \sum_i \sum_j a_{ij} \parallel r_i^{src} - r_j^{trg} \parallel^2 \tag{3.8}$$

Where r_i^{src} and r_j^{trg} are word embedding vectors of source language and target language respectively; a_{ij} are similarity between two words which are precomputed and fixed during the training of $\theta^{src}, \theta^{trg}$. Minimizing this objective loss assures that similar words across 2 languages (i.e, having high similarity) will be embedded nearby each other. The similarities a_{ij} can be estimated by word alignments between sentence pairs in a given parallel corpus. a_ij could be assigned co-occurrence of word i and word j.

In the paper "BILBOWA: Fast Bilingual Distributed Representation without Word Alignment", the authors propose to estimate $\Omega(\theta^{src}, \theta^{trg})$ by sampling:

$$\Omega(\theta^{src}, \theta^{trg}) \triangleq \parallel \frac{1}{m} \sum_{\omega_i \in s^{src}}^{m} r_i^{src} - \frac{1}{n} \sum_{\omega_j \in s^{trg}}^{n} r_j^{trg} \parallel^2$$

where sentence pair s^{src} and s^{trg} is uniformly sampled from a given parallel corpus.

In my project, I use this method because of its rapid computation over large data set. To compute the semantic similarity between source sentence and a translation, I propose 2 metrics:
1) L_2 metric between 2 mean bags of words of 2 sentence.

$$d(s_{source}, s_{mid}) = \| \frac{1}{m} \sum_{\omega_i \in s_{source}}^{m} r_i^{src} - \frac{1}{n} \sum_{\omega_j \in s_{mid}}^{n} r_j^{trg} \|^2$$

2)

$$d(s_{source}, s_{mid}) = \frac{1}{m} \sum_{\omega_i \in s_{source}}^{m} d(r_i^{src}, s_{mid}) + \frac{1}{n} \sum_{\omega_j \in s_{mid}}^{n} d(r_j^{trg}, s_{source})$$

where $d(\omega, s) = \min_{\omega_i \in s} \| \omega - \omega_i \|^2$

3.3 Extension of Dual Learning with Q value

I present following an extension of dual learning with adjustment in reward and estimation of policy gradient

Procedure 2 The dual learning algorithm with Q value estimation

Input: Monolingual corpora D_A,D_B, initial translation models Θ_{AB} and Θ_{BA}, language models LM_A and LM_B, hyperparamter α, beam search size K, learning rates $\gamma_{1,t}$ and $\gamma_{2,t}$

1: **repeat**
2: t = t + 1
3: Sample sentence s_A and s_B from D_A and D_B respectively
4: Set s =s_A
5: Generate K sentences $s_{mid,1}, ..., s_{mid,K}$ using beam search of size K according to translation model $P(. \mid s, \Theta_{AB})$
6: **for** k=1 **to** K **do**
7: $s_{mid,k} = [y_1, ..., y_{T_k}]$
8: **for** t=0 **to** $T_k - 1$ **do**
9: a word a_{t+1} is sampled from $P(a \mid y_1, .., y_t)$, or $P(a \mid BOS)$ if $t = 0$
10: a complete sentence s_t is constructed from prefix $[y_1, .., y_t, a_{t+1}]$
11: $\hat{Q}^{\pi}([y_1, ..y_t], a_{t+1}) = R_{DL}(s_t) - R_{DL}([y_1, .., y_t])$
12: **end for**
13: **end for**
14: Compute the stochastic gradient of Θ_{AB}

$$\Delta_{\Theta_{AB}}\hat{\mathbb{E}}[r] = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{T_i}\sum_{t=0}^{T_i-1}\frac{\partial}{\partial\theta}logP(a_{t+1}|y_{1,..,t}, \pi)\hat{Q}^{\pi}([y_1, .., y_t], a_{t+1})$$

15: Compute the stochastic gradient of Θ_{BA}

$$\Delta_{\Theta_{BA}}\hat{\mathbb{E}}[r] = \frac{1}{K}\sum_{k=1}^{K}[(1-\alpha)\Delta_{\Theta_{BA}}logP(s_{mid,k} \mid s; \Theta_{\Theta_{BA}})]$$

16: Model updates:

$$\Theta_{AB} \leftarrow \Theta_{AB} + \gamma_{1,t}\Delta_{\Theta_{AB}}\hat{\mathbb{E}}[r]$$
$$\Theta_{BA} \leftarrow \Theta_{BA} + \gamma_{2,t}\Delta_{\Theta_{BA}}\hat{\mathbb{E}}[r]$$

17: Set $s = s_B$.
18: Go through line 4 to line 16 similarly
19: **until** convergence

Experiments

4.1 Experiments

4.2 Description of data

I implement 2 versions of Dual Learning on a data set consisting of a English-French parallel corpora of size 20000. The corpora has English - French sentence pairs which are short conversations in the domain of tourism. Beside,

With the bilingual corpora, i have trained 2 neural translation models such as English to French and French to English. I have used Nematus to train 2 models. Following are setting for training:

Vocabulary size	14000
embedding dimension	512
hidden layer size	1024
batch size	32
validation data	devel03

I only chose 14000 most frequent words whose frequencies are 99% of total of frequencies of words contained in Train10 and Hit which is large corpora used for training language models. I will explain Hit after.
I have obtained following results:

Table 4.1: English to French model

Data set	BLEU score
Devel03	39.49
Test09	35.91
Hit train	12.26
Test10	29.58

Table 4.2: French to English model

Data set	BLEU score
Devel03	36.73
Test09	34.12
Hit train	17.52
Test10	26.95

Hit is an other parallel corpora whose sentences are short dialogues collected during the Olympic hold in Beijing. Hit contains 60000 sentence pairs. I have shuffled sentences in Hit and took 90% to train language models, 5% for validation and 5% for test. Therefore,the file "Hit test" contains approximately 3000 sentences pairs.

4.3 Results with lowercased text

Table 4.3: English to French model

Data set	BLEU score
Devel03	35.33
Devel04	33.47
Test09	33.59
Hit train	14.12
Hit test	13.91
Hit dev	14.16
Test10	33.09

Table 4.4: French to English model

Data set	BLEU score
Devel03	35.65
Devel04	36.13
Test09	34.12
Hit train	13.87
Hit test	14.09
Hit dev	14.16
Test10	33.37

4.4 Language Model

4.4.1 Description of data

I use monolingual English Hit and French Hit corpora to train 2 recurrent neural network based - language models. Before training, i lowercased all words in 2 corpora and tokenized them.

4.4.2 Description of experience

I separate Hit corpora into 3 part: 90% for training, 10% for validation and 10% for testing. Following are setting for training:

Vocabulary size	14000
embedding dimension	512
hidden layer size	1024
batch size	32
validation data	Hit dev

Table 4.5: French model

Data set	Perplexity
Hit train	36.97
Hit validation	57.86
Hit test	54.6

4.5 Result

4.6 Analyses

4.7 Conclusion

Table 4.6: English model

Data set	Perplexity
Hit train	53.5
Hit validation	83.76
Hit test	79

Bibliography

[1] *Reinforcement Learning: An Introduction Second edition, in progress ****Draft****.* The MIT Press Cambridge, Massachusetts London, England, 2016.

[2] *Statistical Machine Translation.* CAMBRIDGE UNIVERSITY PRESS, âĂć.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR 2015 proceedings.*

[4] Stephan Gouws, Yoshua Bengio, and Greg Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceeding of the 2016 Conference on Empirical Methods in Natural Language Processing.*

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *âĂć.*

[6] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *âĂć.*

[7] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data.

[8] Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation.