Internship of Master 2 - Computer Science Research

Generative Adversarial Nets in Neural Machine Translation

Author :
NGO HO Anh Khoa
anh-khoa.ngo-ho@u-psud.fr

Internship Supervisor :
Alexandre Allauzen
allauzen@limsi.fr

Host Organization :
LIMSI

Preface

This report is prepared to fulfill the requirement of the master program of Paris-Saclay University on "Machine Learning, Information and Content". It is realized in Computer Science Laboratory for Mechanics and Engineering Sciences (LIMSI), a laboratory of French National Center for Scientific Research (CNRS), which concentrates on Engineering - Information Sciences and Cognitive Science - Linguistics.

I am very appreciated to Mr. Alexandre Allauzen, my supervisor at LIMSI. He gave me very in-time valuable instructions during my research. I also would like to express my gratitude to my LIMSI colleagues for their enthusiastic encouragements and precious instructions.

Contents

List of Figures

Abstract

Generative adversarial networks (GANs) has progress in natural language processing (NLP), specially neural machine translation (NMT). In fact, GANs is a promising approach in improving machine translation. GANs in NMT include a machine translator playing role of generative model and a discriminative model evaluating translated sentence quality. However, GANs confont two problems: The generator differentiation and the capacity of evaluating partially generated sentence. In this research, we apply the work of [10] in English-French translation task and propose two approaches for sentence evaluation. The experiments show an unstable training in general but there is also remarkable improvements in BLEU score.

Key words

Generative adversarial networks, neural machine translation, language model, reinforcement learning, deep learning.

CHAPTER 1

Introduction

Machine translation (MT), a sub-field of natural language processing (NLP), concentrates on translating text from one language to another. Neural machine translation (NMT), an end-to-end learning approach, has an ability to learn directly the mapping from one source-language sentence to the corresponding target-language sentence by using the encoder-decoder framework. This framework includes two recurrent neural networks (RNNs), which means that a RNN encoder consumes the source-language sentence and a RNN decoder generates the target-language sentence [13].

The common approach of training RNNs is maximizing the log predictive likelihood of each true word in the training sentence known the previous observed words. However, in the inference stage, the prediction is based on the previous predicted words. This discrepancy between training and inference causes the exposure bias in the inference stage [12].

[2] proposes scheduled sampling (SS) that the training process is modified by using the predicted previous word, forcing the model to deal with its own mistakes. Another approach is calculating the loss function on the entire generated sentence. In this case, the back-propagation is based on the loss on the entire generated sequence (BLUE score) instead of each transition.

This research concentrates on generative adversarial networks (GANs), an approach of [5]. The main idea of this framework is based on a game theoretic scenario which means that a generative model and a discriminative model are simultaneously trained. In this case, the generative model learns to generate samples and the discriminative model tries to distinguish between the samples from training data-set and from the generator.

In machine translation, the generator - translator generates a sequence and then the discriminator evaluates its quality. The back-propagation of this translator is based on the feedback of the discriminator. However, applying GANs in NMT has two problems.

Firstly, GAN is designed initially for the continuous data, which means that it has difficulties in directly generating sequences of discrete tokens. In other words, the gradient of the output of the discriminator shows how to slightly change the data. However, if the output is a word - discrete token, it is impossible to slightly modify this word. A promising approach to tackle this problem is reinforcement learning. This approach bypasses differentiation problem by performing directly gradient update with reward [12, 10].

Secondly, GAN could not calculate loss for a partially generated sequence. In fact, GAN gives only the loss - score for an entire sequence. It is non-trivial to balance the quality of a partially generated sequence in the current time and of a full sequence in the future. A proposed solution is N-time monte-carlo search strategy [12, 10].

In this report, based the research of [10], we mainly make the following contributions:

- We rebuild the conditional sequence generative adversarial nets of [10] (CSGAN) for English-French translation task.
- We propose two strategies for calculating the sequence score in NMT: Discriminator strategy and Language model (LM) strategy.

Chapter 2

Related work

2.1 Scheduled Sampling

The main idea of Schedual Sampling (SS) is that for every word to predict, the model selects the true previous word or predicted word by flipping a coin with a probability. This probability is followed a schedule of decrease depending on the expected speed of convergence. [2] shows that their approach has a slight improvement in general. [6] demonstrates that it is improper and leads to an inconsistent learning algorithm. In detail, the model, under a trivial solution of SS, memories distribution of words conditioned on their positions in the sentence, rather than on the previous words. In comparison between GANs and SS, SS does not require to train an additional discriminator network.

2.2 Adversarial Learning

The idea of generative adversarial networks has remarkable success in NMT. Professor Forcing (PF) [8], Sequence GAN (SeqGAN) [12], Conditioned sequence GAN-NMT (CSGAN-NMT) [10], Adversarial Dialoge Generation (ADG) [9] are based on the framework of GANs by using two models generator and discriminator.

For Professor Forcing [8], it is adversarial domain adaptation between hidden states from sampling mode (Inference) and teacher forcing mode (Training). In detail, there are two running modes: Teacher-Forcing mode and Free-Running mode. Teacher-Forcing mode means that the prediction of the next token is based on the true token. Free-Running mode means that the prediction is based on the self-generated token. The generator generates the behavior of the network (Output and Hidden states) which is indistinguishable whether its inputs from Teacher-Forcing mode or from Free-Running mode. The discriminator estimates the probability of the behavior of the network produced in Teacher-Forcing mode.

SeqGAN [12] shows a different gradient training policy that the feedback of discriminator is the probability of a sentence coming from ground truth data. In detail, the optimization target of discriminator is minimizing the cross entropy between the ground truth label and the predicted probability. CSGAN-NMT of [10], based on this policy, applies GAN in NMT but its feedback is the probability of sentence being human-generated sentence. ADG of [9] has a similar approach but concentrates on open-domain dialogue generation, which generates sequences indistinguishable from human-generated dialogue utterances.

In general, the main difference between PF and other GANs is the objective of discrimination. In fact, GANs discriminate between human-generated/real samples and machine-generated samples. For PF, the model concentrates on the input of the token prediction. The input is from the training set or the self-generated token.

On point important is that PF uses the intermediate hidden vectors of the generator for the discrimination procedure, which makes the system differentiable. However, SeqGAN, CSGAN-NMT and ADG use mainly its sequence output. An advantage of PF over GANs is that PF can be used to learn a generative model over discrete random variables without requiring to approximate back-propagation through discrete spaces.

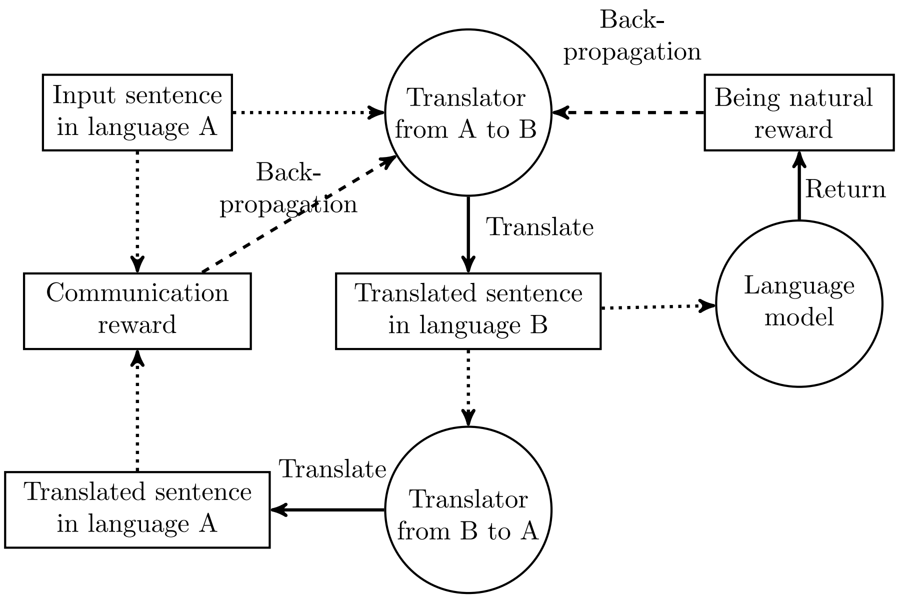


Figure 2.1: Dual learning architecture

2.3 Dual Learning

[7] proposes neural machine translation system by using monolingual data in a dual-learning game. In this case, there are two agents or translators which perform two opposite translation tasks in a closed loop. Through a reinforcement learning process, an agent updates automatically with the feedback of the other agent. The feedback is the language-model likelihood of the output of a translation model (Being natural reward) and the reconstruction error of the original sentence after the primal and dual translation (Communication reward). Its architecture is found in figure 2.1. The objective is to maximize the weighted sum of the reward based on monolingual data and the likelihood on bilingual data. One remarkable result is that the model of 10% of bilingual data could get a comparable accuracy to the model of full bilingual data.

It is important to realize that dual learning uses a language model (LM) to evaluate the quality of a sentence being natural. In the case of GAN, a discriminator detects whether the input sentence is generated by human or machine. In general, both kinds of model are based on reinforcement learning and assign reward for each sentence.

CHAPTER 3

GAN for NMT

3.1 Architecture

The architecture of GAN includes a generator - machine translator and a discriminator. The components have two opposite objective functions. The generator tries to make fool the discriminator by improving its translated sentences whereas the discriminator tries to discriminate well. In this case, the discriminator distinguishes human-translated sentence from machine-translated sentence by providing the probability of a sentence being human-translated. Therefore, the machine translator learns from this discriminator feedback. The architecture of GAN is in 3.1 and 3.2

Generator

In this research, the generator G is an attention-based neural machine translation model [1]. Its attention mechanism which learns to align and translate jointly allows the model to predict a word by automatically searching for its relevant parts of a source sentence. The probability of a word predicted y_i is:

$$p(y_i|y_{1:i-1}, x) = g(y_{i-1}, s_i, c_i) \tag{3.1}$$

where:

- x: Source sentence
  $x = (x_1, ..., x_{T_x})$
- $y_{1:i-1}$: Previous translated words
  $y_{1:i-1} = (y_1, ..., y_{i-1})$
- s_i: Decoder hidden state for time step i
  $s_i = f(s_{i-1}, y_{i-1}, c_i)$

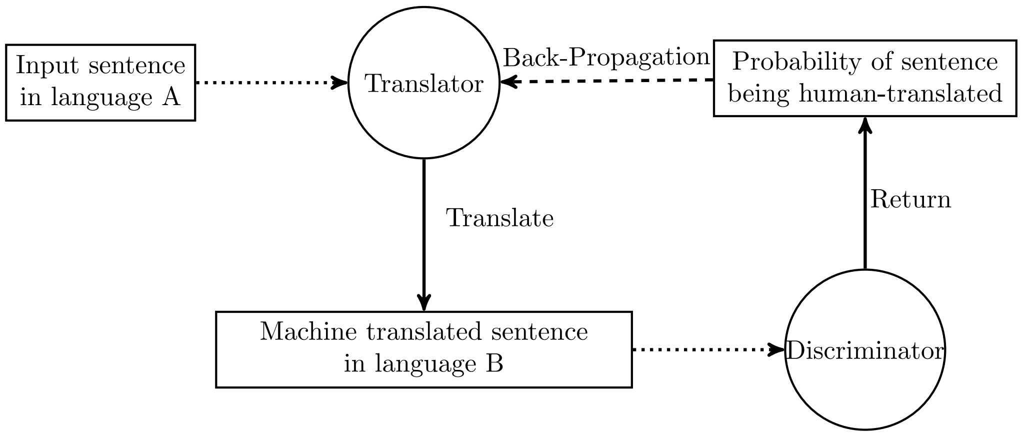

Input sentence in language A
Translator
Back-Propagation
Probability of sentence being human-translated
Translate
Return
Machine translated sentence in language B
Discriminator

Figure 3.1: Main architecture of GAN: Train Translator

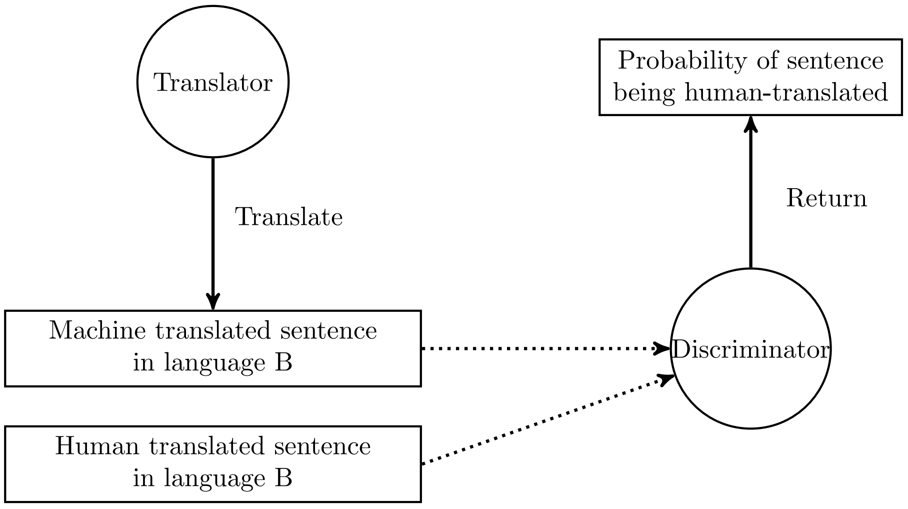

Translator
Translate
Probability of sentence being human-translated
Return
Machine translated sentence in language B
Discriminator
Human translated sentence in language B

Figure 3.2: Architecture of GAN: Train Discriminator

- c_i: Distinct context vector at time step i
  $c_i = \sum_{j=1}^{T_x} a_{i,j} h_j$
- h_j: Annotation of x_j from a bidirectional RNN
  $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$
- $a_{i,j}$: Weight of annotation h_j at step i
  $a_{i,j} = \frac{exp(e_{ij})}{\sum_{t=1}^{T_x} exp(e_{i,t})}$
- $e_{i,j}$: Alignment model scoring how well the inputs around j and the output at position i match
  $e_{i,j} = tanh(s_{i-1}, h_j)$

Here f and g are nonlinear transform function and implemented as gated recurrent unit (GRU) in the experiment of this research

The objective function of the machine translator is based on actor-critic reinforcement learning, that generating a word is an actor and gets a reward from the discriminator D. In detail, the discriminator evaluates a part of a sentence including this word and all of its previous predicted words. As a result, the objective is maximizing the expected end reward of a sentence from the start state [12].

$$\triangledown J_G = \sum_{y_t} R_D^G((y_{1:t-1}, x), y_t) \triangledown_\theta \, log \, p_G(y_t | y_{1:t-1}, x) \tag{3.2}$$

Discriminator

In this research, the discriminator is a convolutional neural network (CNN) binary classifier which provides the probability of a sentence being human-translated.

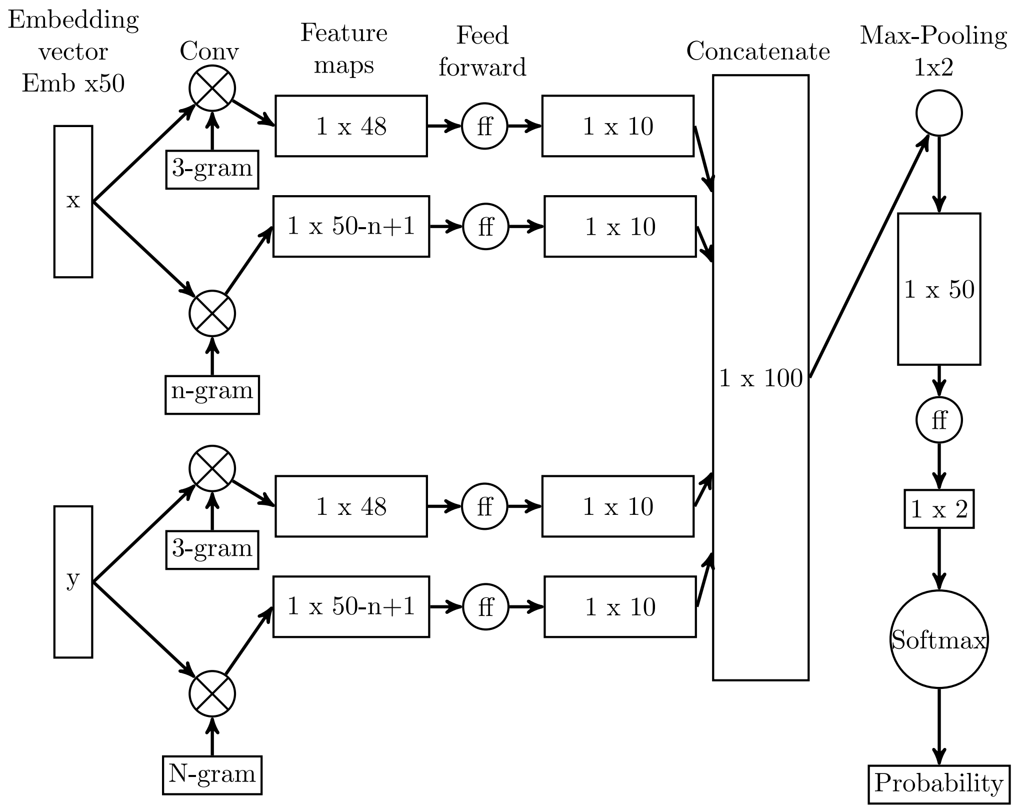


Figure 3.3: Discriminator architecture

The objective function of discriminator is

$$L(p, \hat{p}) = -p * log(\hat{p}) - (1 - p) * log(1 - \hat{p}) \tag{3.3}$$

where:

- p: True probability
- $\hat{p}$: Predicted probability

The input of CNN discriminator is both source-language x and target-language y sentence. These sentences are transformed into embedding vectors (Word dimension x Sentence length). The architecture of discriminator in this research includes fives filters (3-gram, 5-gram, 7-gram, 9-gram and 11-gram filter) to provide fives different feature maps. These feature maps are concatenated and then applied max-pooling operation. The final probability is returned by using the function Soft-max. The size of the maps are modified by a feed-forward layer. The architecture of discriminator is shown in 3.3.

3.2 Policy gradient training

This section describes three strategies for assigning the reward for each sentence.

Based on the idea actor-critic of reinforcement learning, instead of calculating only the reward for a full sentence, the model have to provide the reward for each word. N-time Monte-Carlo search (MC-search) strategy [10, 12, 9] and discriminator strategy [9] are two approaches for partially generated sequences. For N-time Monte-Carlo search strategy, the discriminator is trained to distinguish finished sequences. Therefore, MC-search has to generate full sequence for each word. It means that this strategy concentrates more on the long-term reward than on the immediate interests.

For discriminator strategy, the discriminator is able to assign directly the reward for each token. [9] mentioned that discriminator strategy is more time-effective but less accurate than Monte Carlo search. In fact, MC-search requires repeating the sampling process of each prefix of each sequence, which is significantly time-consuming.

The main idea of these two above approaches is using discriminator to evaluate the target-language sentence quality which means human-translated level. Moreover, another method is using language model which evaluates the level of confidence that the translated sentence is natural in target-language. It is adopted in Dual Learning [7]. Therefore, a strategy proposed in this research is combining both reward from discriminator and language model.

N-time Monte-Carlo search strategy

This approach tackles the reward for a partially generated sequence by generating the full sentence based on the known part of this sentence. In detail, a sentence is translated by Beam-search method. For each predicted word in this sentence, the current generator generates N samples of the rest words by multinomial sampling method. As a result, reward for a predicted word is

$$R_D^G((y_{1:t-1}, x), y_t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D(y_{1:L_n}^n, x),\ y_{1:L_n}^n \in MC^G((y_{1:t}, x), N) & \text{for } t < L \\ D(y_{1:T}, x) & \text{for } t = L \end{cases} \tag{3.4}$$

where:

- $D(y_{1:Ln}^n, x)$: Probability of a sentence being human-translated from the Discriminator
- $y_{1:L_n}^n$: Translated sentence from MC-search
- L_n: Length of the sentence sampled by MC-search
- N: Number of sentence sample from MC-search
- T: Length of translated sentence by Beam-search

Discriminator strategy

The reward is assigned directly from the discriminator for each partially generated sentence, which means that the discriminator learns to discriminate not only full sentences but partially generated sentences. In detail, the translated sentence from beam-search is broken into partial sequences. Each partial sequence is assigned a reward by the discriminator.

$$R_D^G((y_{1:t-1}, x), y_t) = D(y_{1:t}, x) \tag{3.5}$$

where:

- y ranges from $y_{1:2}, y_{1:3}, y_{1:4}, ...$ to $y_{1:T}$

In this case of training discriminator, an problem occurred is the repetition continuous of the initial words of a sentence. For example, the token y_1 appears in all of partial sequences. Therefore, one example is randomly sampled from machine-generated and human-generated partial sequences for the negative and positive example respectively.

Language model strategy

The reward is from two components the discriminator (Monte-Carlo strategy or Discriminator strategy) and the language model. It is modified by a parameter α increasing for each batch. The language model is trained with a large monolingual data-set, based on the hypothesis that target-language has a significantly larger data-set than source-language [7]. In this research, the language model is a standard model of RNNs without any specific modifications.

$$R_D^G((y_{1:t-1}, x), y_t) = \alpha * D(y_{1:t}, x) + (1 - \alpha) * LM(y_t | y_{1:t-1}) \tag{3.6}$$

where:

- $LM(y_t|y_{1:t-1})$: Probability of y_t based on its previous words, provided by a language model.

3.3 Problem of a strong discriminator

For GAN, the generator needs the reward passed back from the discriminator. This reward guides the generator to reach the gold-standard target. If the discriminator is too strong in the beginning of the training period, the generator could receive a reward insignificant or a null reward, leading to a breakdown of the training process. The solutions are mentioned in this section.

Teacher Forcing

[9, 12, 10] use Teacher Forcing of [8]. In this case, for each training iteration, there are two periods of update for discriminator and generator. For the period of generator update, the generator updates with the reward from the discriminator and then the reward of the true target sentence (Teacher Forcing). In detail, the discriminator automatically assigns a reward of 1 to the golden target-language sentence and the generator uses this reward to update itself.

$$\bigtriangledown J_G = \sum_{y_t} 1 * \bigtriangledown_\theta log\, p_G(y_t|y_{1:t-1}, x) \tag{3.7}$$

CNN or RNN discriminator

[10], based on the result of GAN with RNN(BiLSTM, LSTM) and CNN, shows that RNN-based discriminator is not stable because of its sharp decrease of BLEU score. They explained that this discriminator RNNs could achieve the high score after few updates, which means that it is too strong for the generator. Therefore, they use CNN for their experiment.

For the comparison of CNN and RNN, [11] proposed an experiment between CNN, GRU and LSTM. RNNs are well suited to encoder order information and long-range context dependency. CNN are considered good at extracting local and position-invariant features, which is good for classification. In detail, RNN (Bi-RNN) chooses the last hidden state to represent the sentence, which probably causes the bad performance of text classification prediction if this classification depends on a few local regions. Moreover, GRU and CNN are comparable when sentence length is small. [11] concludes that the performance in text classification depends on how often the comprehension of global/long-range semantics is required.

CNNs use convolutional layers and max-pooling operation. The convolution layer involves a filter of n-gram, in which each filter creates a one feature map at every position in the sentence. The model applies a-max-over-time pooling operation to the feature map and take its maximum value. As a result, the pooling operator tries to capture the most important features.

In general, the discrimination between human-generated and machine-generated requires the comprehension of long-range semantics. Therefore, RNNs are too good for this task. In this research, RNNs are used for generator - translation model and CNNs for discriminator.

Other solutions

The number of updates iteration is also a solution to keep balance between generator and discriminator. Moreover, the intial accuracy of discriminator is an hyperparameter to decide whether to update discriminator [8, 10]. In [8], if its accuracy is greater than an upper bound, model does not update discriminator. These solutions are also applied in this research.

3.4 Algorithm

In the pre-training period, a generator G is pre-trained using the maximum likelihood estimation on the parallel training set until a translation performance threshold. This generator translates multinomially the sentences of the same training set into the machine-translated sentences for training discriminator. In the similar case, this discriminator D is also pre-trained on both parallel training set and these machine-translated sentences from G until a classification performance threshold.

In GAN training period, G translates a batch sampled randomly into machine-translated sentences by beam-search method. For each sentence, its reward is computed by three different strategies: Monte-Carlo search strategy (E.q 3.4), discriminator strategy (E.q 3.5) and language model strategy (E.q 3.6). Teacher-Forcing pushes generator to generate the golden target sentence (E.q 3.7). Simultaneously, G prepares also machine-translated sentences for updating discriminator (E.q 3.3).

```
Pre-train the generator G on the parallel training set
Pre-train the discriminator D
while not convergence do
    Sample randomly a batch: Input sentences and its human-translated sentences $(X, hY)_{1:S}$
    for g-steps do
        for $s \in 1{:}S$ do
            G translates $X_s$ into machine-translated sentence $mY_{1:T} = (my_1, ..., my_T)$ by
             beam-search
            for $t \in 1{:}T$ do
                Compute reward $R_D^G(my_{1:t-1}, my_t)$ by Equation 3.4, 3.5 or 3.6
            end
        end
        Update G with Discriminator: Update with $mY_{1:S}$ by Equation 3.2
        Update G with Teacher Forcing: Update with $hY$ by Equation 3.7
    end
    for d-steps do
        G translate $X_{1:S}$ into machine-translated sentences $mY'_{1:S}$ as Negative examples
        $hY_{1:S}$ Human-generated sentences as Positive examples
        Update D: Update by Equation 3.3
    end
end
```

CHAPTER 4

Experiments and Results

4.1 Setup

The experiment is English-French translation task with a bilingual data-set. Its training set *train*, its development set *dev* and its testing set *test* have 19972, 506 and 469 sentences respectively. A monolingual data-set for language model has 62227 sentences. Vocabulary size for source-language is 7751 words and for target-language 9067 words.

All of training process use Adadelta optimizer. Embedding vector dimension is 512. RNN's hidden layer dimension is 1024. For Monte-Carlo search strategy, number of sentence rolled is 20 sentences and the sentence's maximum length is 50 words. Each batch has 80 sentences and the model is evaluated by BLEU score each 100 batches.

The initial BLEU score of NMT model is 38.850. Monte-Carlo search strategy has the initial discriminator with an accuracy of 0.782 whereas the model discriminative of discriminator strategy has an accuracy of 0.835. The accuracy of discriminator ranges between 0.75 and 0.85 during the training period. In our experiments, we use the initial NMT model as our baseline model which is trained under the maximum likelihood estimation.

The code, based on the framework NMTPY [14], is found in [16].

4.2 Results

This section shows the results of three strategies mentioned above. To compare with discriminator strategy, Monte-Carlo search strategy and language model strategy are significantly time-consuming. It is well reported in [9]. The training process of these two strategies have not been finished, which causes the difficulties in our evaluation.

Monte-Carlo search strategy

The result of this strategy is in figures 4.1 and 4.2. The loss of baseline model shows clearly a drop whereas GAN witnesses a fluctuation in the first epoch shown in figure 4.1. Figure 4.2 demonstrates the BLEU score (Development set) of the first 100 batches decreasing significantly from 38.85 to 6.67.

Some sentence samples of the first 100 batches are in figure A.1. These sentences show that discriminator assigns the wrong rewards, which could lead to a large drop of BLEU score in the first batches. In fact, these sentences are similar to the human-translated sentences but not the target-language sentences.

Based on this result, it is impossible to conclude the performance of this strategy. Therefore, it could not prove a positive effect on the loss score.

Discriminator strategy

The result of this strategy is in figures 4.3, 4.4, 4.5 and 4.6. In figure 4.3, from the same starting point, the loss of both translation models (GANs and baseline model) have similar downward patterns. However, it is well recognized that the loss of baseline model decreases sharply and reaches the lower points. In detail (Figure 4.5), there is a large gap between loss from discriminator-reward and teacher-forcing in the first 2000 batches. In the next period, this gap reduces slightly, which means that the discriminator assigns more and more the correct rewards and the translated sentences are more and more similar to the target-language sentences. 4.6 shows the relationship between BLEU score and generator loss from discriminator, we observe that the BLEU score is considerably influenced by this kind of loss.

Figure 4.4 shows a dramatic downfall of BLEU score (Development set) from 38.85 to 5.24 in the first 100 batches, followed by slight improvements in the next 4000 batches. From batch 4500^{th}, there is an increasing but unstable changes from about 7.5 to under 30. The score then fluctuates significantly and ranges between about 10 and about 40. The highest BLEU score of GANs found is about 42.340 at validation 183^{th} (Table 4.1).

In general, this strategy could not lead to a stable training based on BLEU score result. However, it has also positive improvements in the end of training period.

Model	Initial model	GANs	Improve
Train	46.71	94.73	+48.02
Dev	38.85	42.340	+3.49
Test	38.39	36.15	-2.24

Table 4.1: BLEU score of Discriminator strategy

Language model strategy

This experiment includes two sub-experiments, that the discriminator reward is calculated by Monte-Carlo strategy or discriminator strategy. The result of the first sub-experiment (Language model - Monte-Carlo strategy) is in figures 4.7 and 4.8. This strategy has the similar figures of Monte-Carlo search strategy. It does not show an improvement that its BLEU score (Development set) drops to 7.26 in first 100 batches. In the same case of Monte-Carlo search strategy, we could not have a final conclusion of the performance for language model strategy.

The result of the second sub-experiment (Language model - Discriminator strategy) is in figures 4.9 and 4.10. We observe that the loss of generator and baseline both decrease slightly and have the similar pattern of discriminator strategy. Its BLEU score has also a collapse in the first 100 batches, occurred in all strategies. Its best BLEU score found is 42.550 at validation 81^{th} (Table 4.2).

Model	Initial model	GANs	Improve
Train	46.71	80.23	+33.51
Dev	38.85	42.550	+3.7
Test	38.39	37.06	-1.13

Table 4.2: BLEU score of Language model - Discriminator strategy

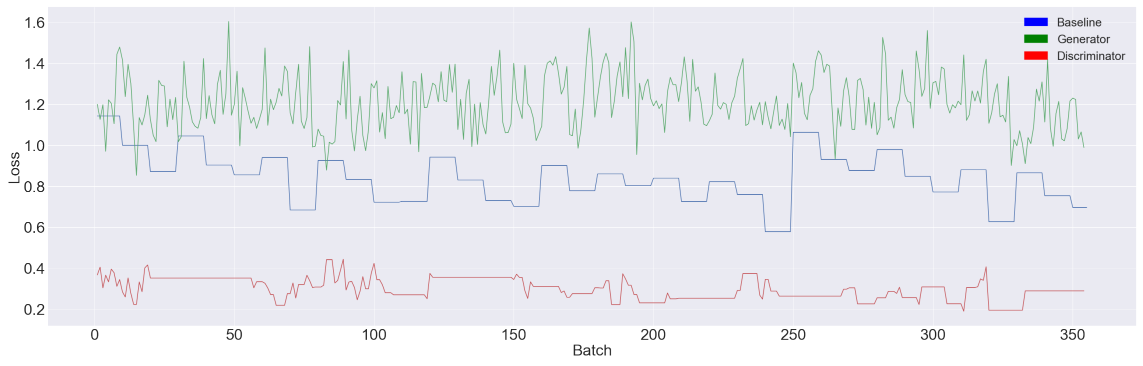


Figure 4.1: Loss in monte-Carlo search strategy.

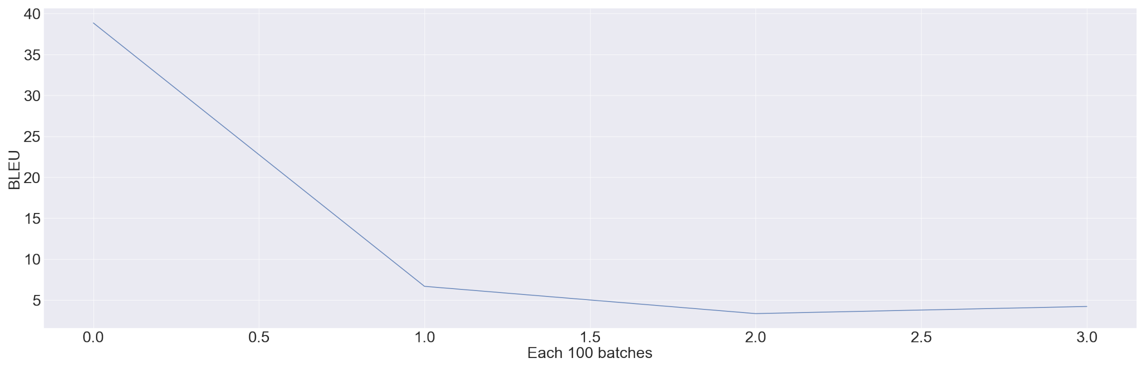


Figure 4.2: BLEU score in Monte-Carlo search strategy.

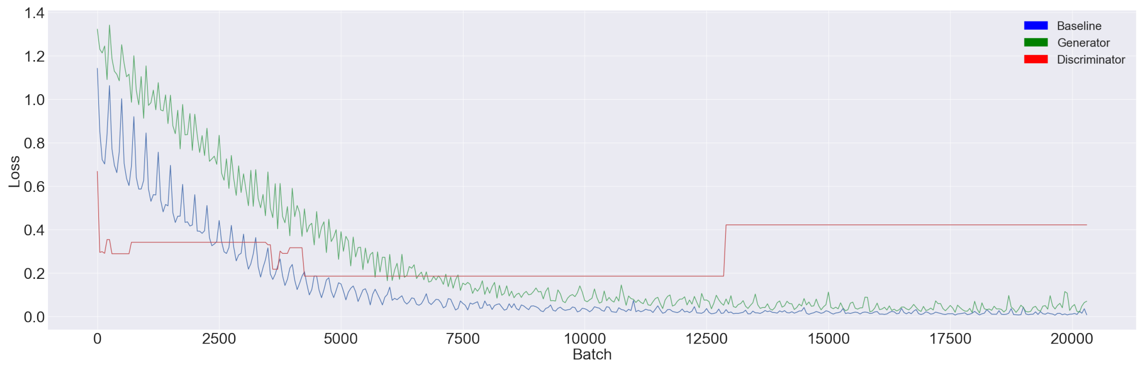


Figure 4.3: Loss in discriminator strategy.

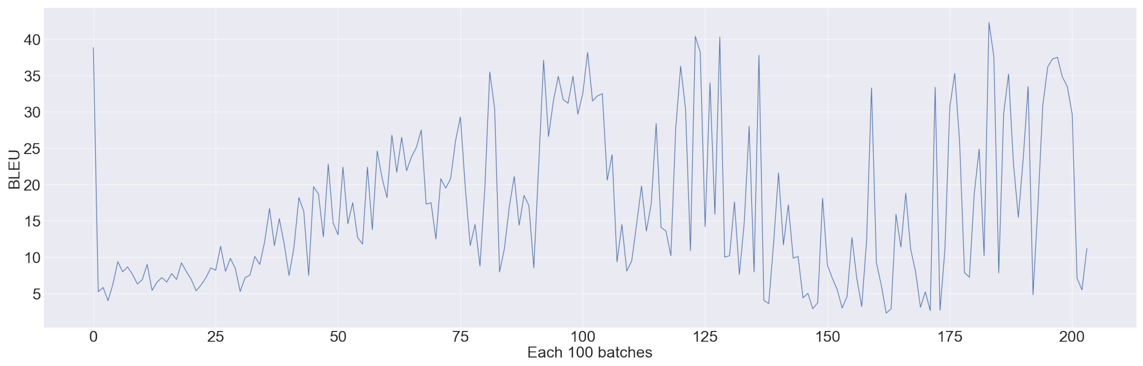


Figure 4.4: BLEU score in discriminator strategy.

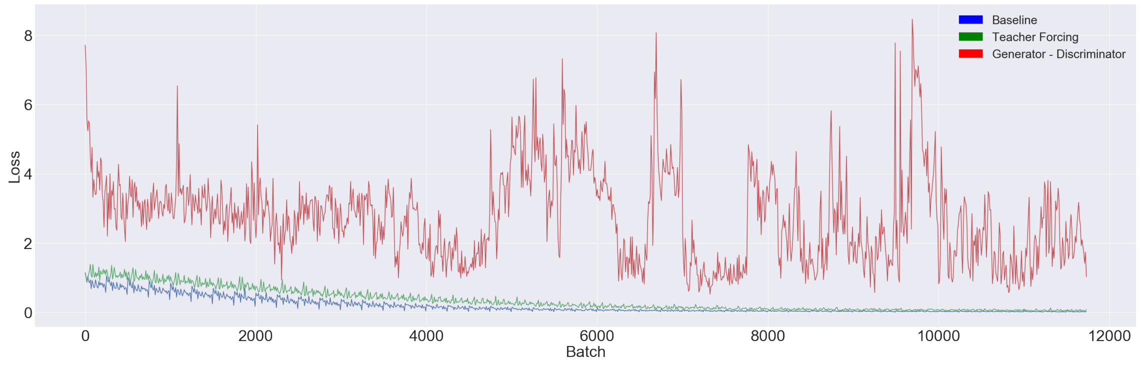


Figure 4.5: Loss detail in discriminator strategy.

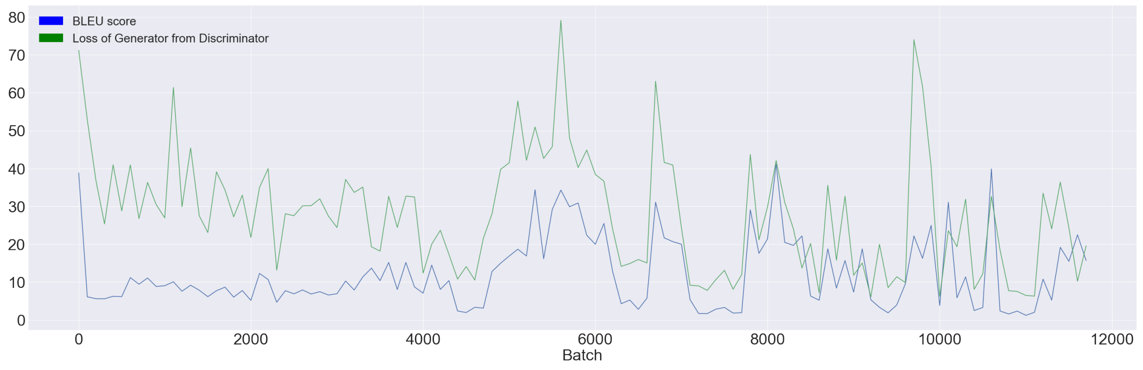


Figure 4.6: Relationship between BLEU score and loss from discriminator-reward (The value of loss is scaled to show clearly this relationship).

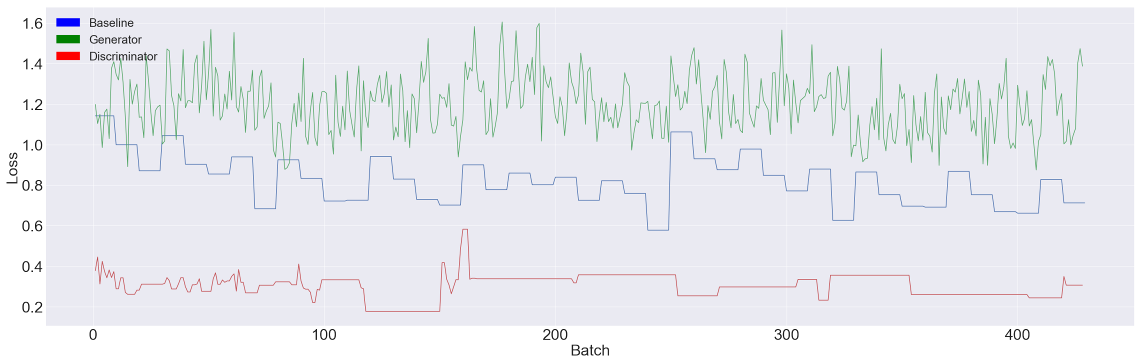


Figure 4.7: Loss in language model strategy with Monte-Carlo search.

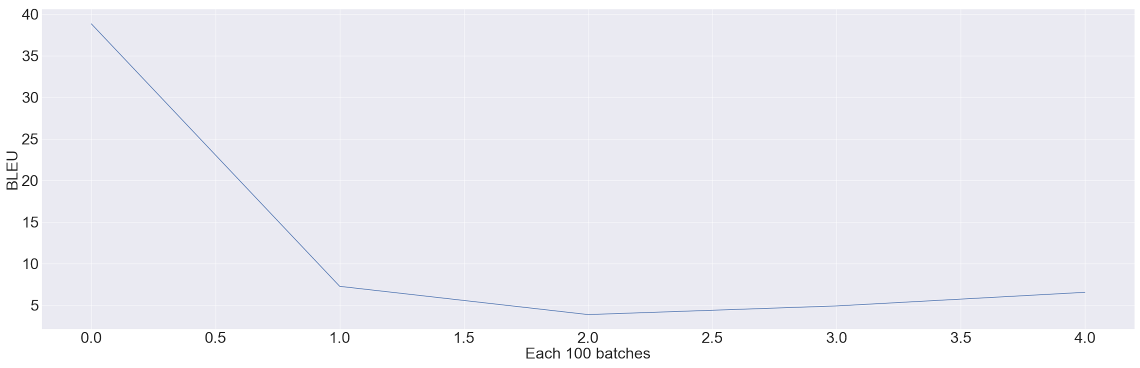


Figure 4.8: BLEU score in language model strategy with Monte-Carlo search strategy.

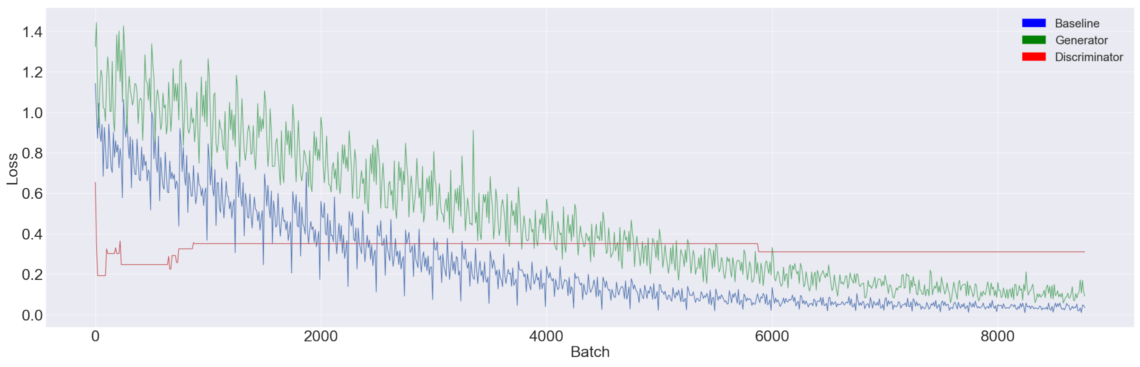


Figure 4.9: Loss in language model strategy with discriminator strategy.

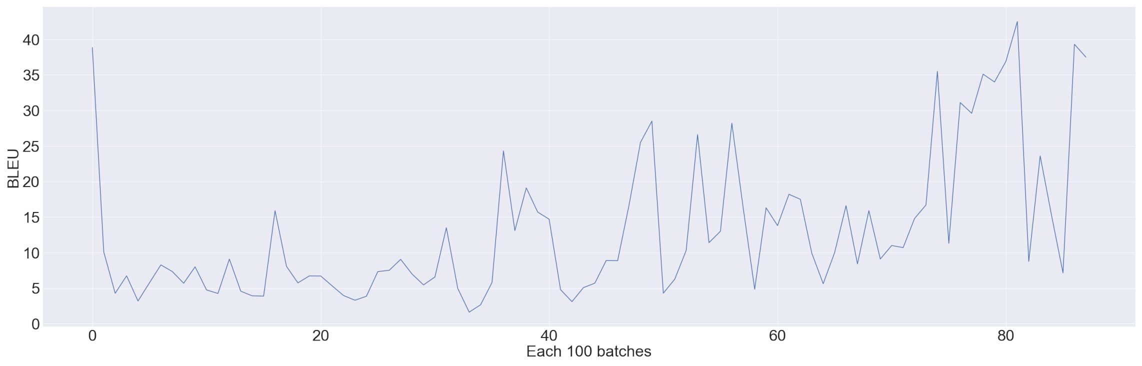


Figure 4.10: BLEU score in language model strategy with discriminator strategy.

CHAPTER 5

Conclusion

In this research, based on the work of [10] in Chinese-English translation task, we apply GANs into English-French translation task. The two approaches proposed are assigning directly reward from discriminator and considering language model as an extra quality evaluator.

In our experiment, we recognize that the discriminator assigns the high rewards for the sentences which are similar to human-translated sentences but not target-language sentences. [9] mentioned also that GANs are more beneficial to tasks that the distribution of machine-translated sentences and of the target-language sentences are significantly different. Therefore, we conjecture that it could cause the problem of a dramatic drop of BLEU score in the first period and an unstable training (Figure 4.4, 4.10).

In general, the experiment result could not show clearly the positive performance of these approaches because of the lack of the final results. To our best knowledge, we observe that GANs have a similar but weaker effect of the log likelihood on training NMT.

Our ongoing work is focus on the experiments of the proposed strategies and its variants. We are analyzing the large drop of BLEU score and the reason of the unstable training. Our future work concentrates on improving NMT in the case of the small discrepancy between machine-translated and human-translated sentences, and also applying variational neural networks in GANs.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. ICLR 2015.

[2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, Noam Shazeer. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Network*. 9 Jun 2015.

[3] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, Samy Bengio. *Generating Sentences from a Continuous Space*. SIGNLL Conference on Computational Natural Language Learning (CONLL), 2016.

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, Victor Lempitsky. *Domain-Adversarial Training of Neural Networks*. Journal of Machine Learning Research 2016, vol. 17, p. 1-35.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. *Generative Adversarial Networks*. 10 Jun 2014.

[6] Ferenc Huszar. *How (not) to train your generative model: Scheduled sampling, likelihood, adversary?*. 16 Nov 2015.

[7] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, Wei-Ying Ma. *Dual Learning for Machine Translation*. NIPS 2016.

[8] Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, Yoshua Bengio. *Professor Forcing: A New Algorithm for Training Recurrent Networks*. NIPS 2016.

[9] Jiwei Li, Will Monroe, Tianlin Shi, Sebastien Jean, Alan Ritter, Dan Jurafsky. *Adversarial Learning for Neural Dialogue Generation*. 23 Jan 2017.

[10] Zhen Yang, Wei Chen, Feng Wang, Bo Xu. *Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets*. 15 Mar 2017.

[11] Wenpeng Yin, Katharina Kann, Mo Yu, Hinrich SchÃ$\frac{1}{4}$tze. *Comparative Study of CNN and RNN for Natural Language Processing*. 9 Jun 2015.

[12] Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu. *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*. The Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017).

[13] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 8 Oct 2016.

[14] Ozan Caglayan, Mercedes Garcia-Martinez, Adrien Bardet, Walid Aransa, Fethi Bougares, Loic Barrault. *NMTPY: A Flexible Toolkit For Advanced Neural Machine Translation Systems*. `https://github.com/lium-lst/nmtpy`

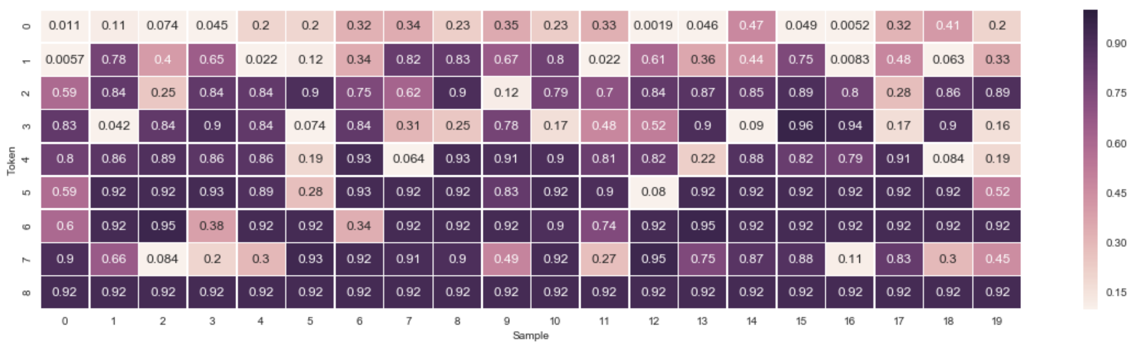
[15] Theano. `http://deeplearning.net/software/theano/`

[16] Experiment code. `https://github.com/ngohoanhkhoa/GAN-NMT`

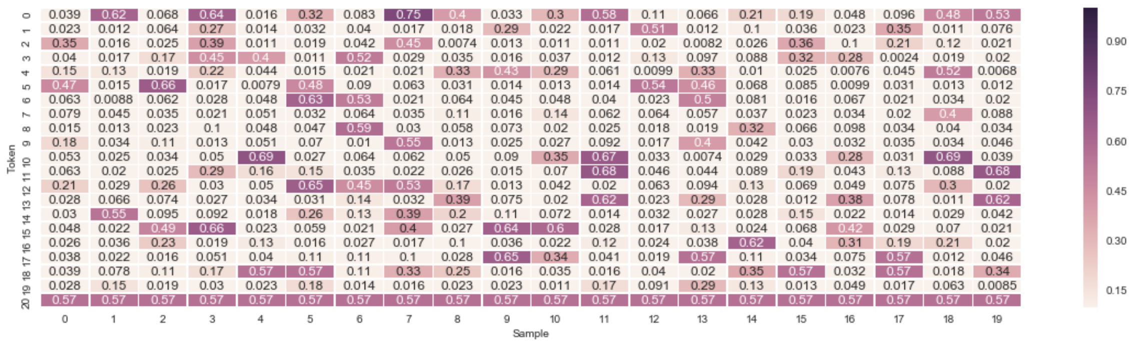
Appendix A

Monte-Carlo Strategy: Sentence rewards

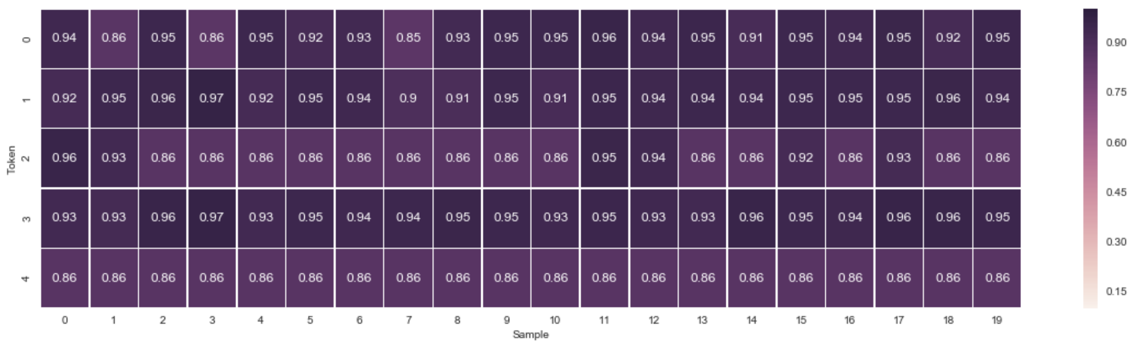
En sentence: I'd like to buy what I need . will you pay me back ?
Fr sentence: pourriez -vous me regler a mon retour ?



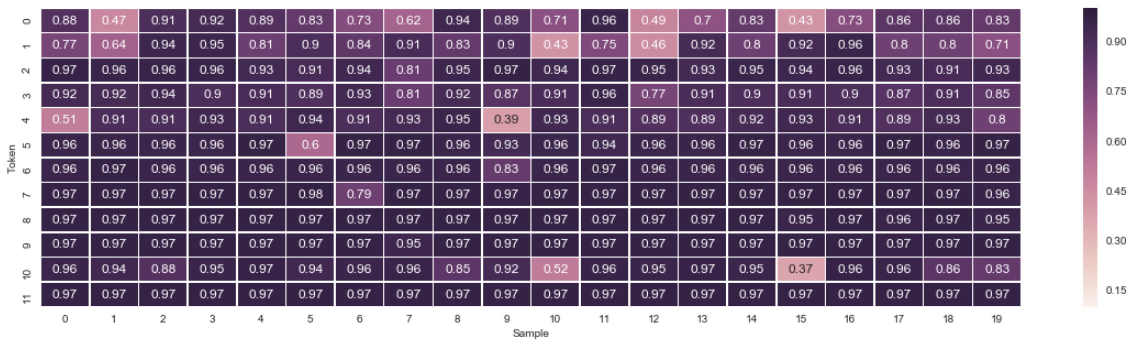
En sentence: here you go . a glass of water and a Michelob . please enjoy the rest of your meal .
Fr sentence: un d' eau et une drole de l' eau . veuillez vous en trouverez un peu plus bon repas .



En sentence: yes , it's next door .
Fr sentence: c' est ouvert .

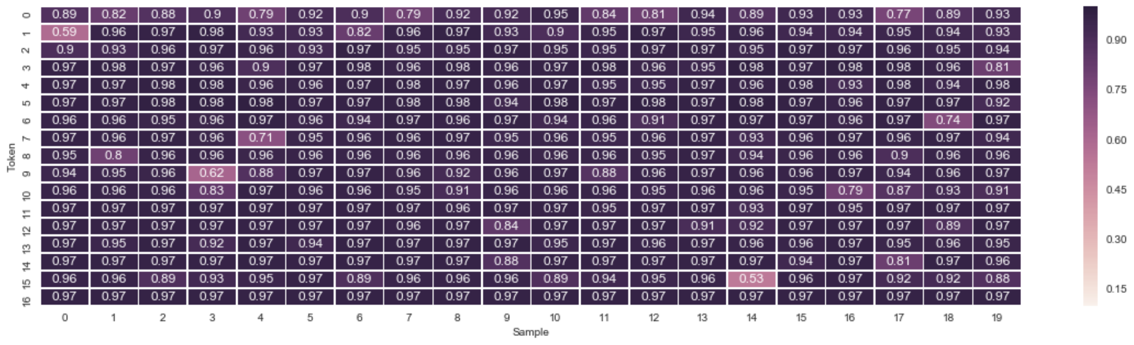


En sentence: are there any department stores around here ?
Fr sentence: y a-t-il des magasins magasins de mer dans le coin ?



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.88	0.47	0.91	0.92	0.89	0.83	0.73	0.62	0.94	0.89	0.71	0.96	0.49	0.7	0.83	0.43	0.73	0.86	0.86	0.83
1	0.77	0.64	0.94	0.95	0.81	0.9	0.84	0.91	0.83	0.9	0.43	0.75	0.46	0.92	0.8	0.92	0.96	0.8	0.8	0.71
2	0.97	0.96	0.96	0.96	0.93	0.91	0.94	0.81	0.95	0.97	0.94	0.97	0.95	0.93	0.95	0.94	0.96	0.93	0.91	0.93
3	0.92	0.92	0.94	0.9	0.91	0.89	0.93	0.81	0.92	0.87	0.91	0.96	0.77	0.91	0.9	0.91	0.9	0.87	0.91	0.85
4	0.51	0.91	0.91	0.93	0.91	0.94	0.91	0.93	0.95	0.39	0.93	0.91	0.89	0.89	0.92	0.93	0.91	0.89	0.93	0.8
5	0.96	0.96	0.96	0.96	0.97	0.6	0.97	0.97	0.96	0.93	0.96	0.94	0.96	0.96	0.97	0.96	0.96	0.97	0.96	0.97
6	0.96	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.83	0.96	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
7	0.97	0.97	0.97	0.97	0.97	0.98	0.79	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.96
8	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.97	0.96	0.97	0.95
9	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
10	0.96	0.94	0.88	0.95	0.97	0.94	0.96	0.96	0.85	0.92	0.52	0.96	0.95	0.97	0.95	0.37	0.96	0.96	0.86	0.83
11	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97

En sentence: the Japanese just can't get along without taking a bath every day .
Fr sentence: nous ne sont pas pris de nous faire un n' asseoir pas de la baignoire .



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.89	0.82	0.88	0.9	0.79	0.92	0.9	0.79	0.92	0.92	0.95	0.84	0.81	0.94	0.89	0.93	0.93	0.77	0.89	0.93
1	0.59	0.96	0.97	0.98	0.93	0.93	0.82	0.96	0.97	0.93	0.9	0.95	0.97	0.95	0.96	0.94	0.94	0.95	0.94	0.93
2	0.9	0.93	0.96	0.97	0.96	0.93	0.97	0.95	0.95	0.97	0.95	0.95	0.97	0.97	0.95	0.97	0.97	0.96	0.95	0.94
3	0.97	0.98	0.97	0.96	0.9	0.97	0.98	0.96	0.98	0.96	0.97	0.98	0.96	0.95	0.98	0.97	0.98	0.98	0.96	0.81
4	0.97	0.97	0.98	0.98	0.96	0.96	0.97	0.98	0.97	0.96	0.97	0.95	0.95	0.97	0.96	0.98	0.93	0.98	0.94	0.98
5	0.97	0.97	0.98	0.98	0.98	0.97	0.97	0.98	0.98	0.94	0.98	0.97	0.98	0.97	0.98	0.97	0.96	0.97	0.97	0.92
6	0.96	0.96	0.95	0.96	0.97	0.96	0.94	0.97	0.96	0.97	0.94	0.96	0.91	0.97	0.97	0.97	0.96	0.97	0.74	0.97
7	0.97	0.96	0.97	0.96	0.71	0.95	0.96	0.96	0.97	0.95	0.96	0.95	0.96	0.97	0.93	0.96	0.97	0.96	0.97	0.94
8	0.95	0.8	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.95	0.97	0.94	0.96	0.96	0.9	0.96	0.96
9	0.94	0.95	0.96	0.62	0.88	0.97	0.97	0.96	0.92	0.96	0.97	0.88	0.96	0.97	0.96	0.96	0.97	0.94	0.96	0.97
10	0.96	0.96	0.96	0.83	0.97	0.96	0.96	0.95	0.91	0.96	0.96	0.96	0.95	0.96	0.96	0.95	0.79	0.87	0.93	0.91
11	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.97	0.95	0.97	0.97	0.93	0.97	0.95	0.97	0.97	0.97
12	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.84	0.97	0.97	0.97	0.91	0.92	0.97	0.97	0.97	0.89	0.97
13	0.97	0.95	0.97	0.92	0.97	0.94	0.97	0.97	0.97	0.97	0.95	0.97	0.96	0.97	0.96	0.96	0.97	0.95	0.96	0.95
14	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.88	0.97	0.97	0.97	0.97	0.97	0.94	0.97	0.81	0.97	0.96
15	0.96	0.96	0.89	0.93	0.95	0.97	0.89	0.96	0.96	0.96	0.89	0.94	0.95	0.96	0.53	0.96	0.97	0.92	0.92	0.88
16	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97

Figure A.1: Some sentence rewards: Epoch 1 - Batch 62.

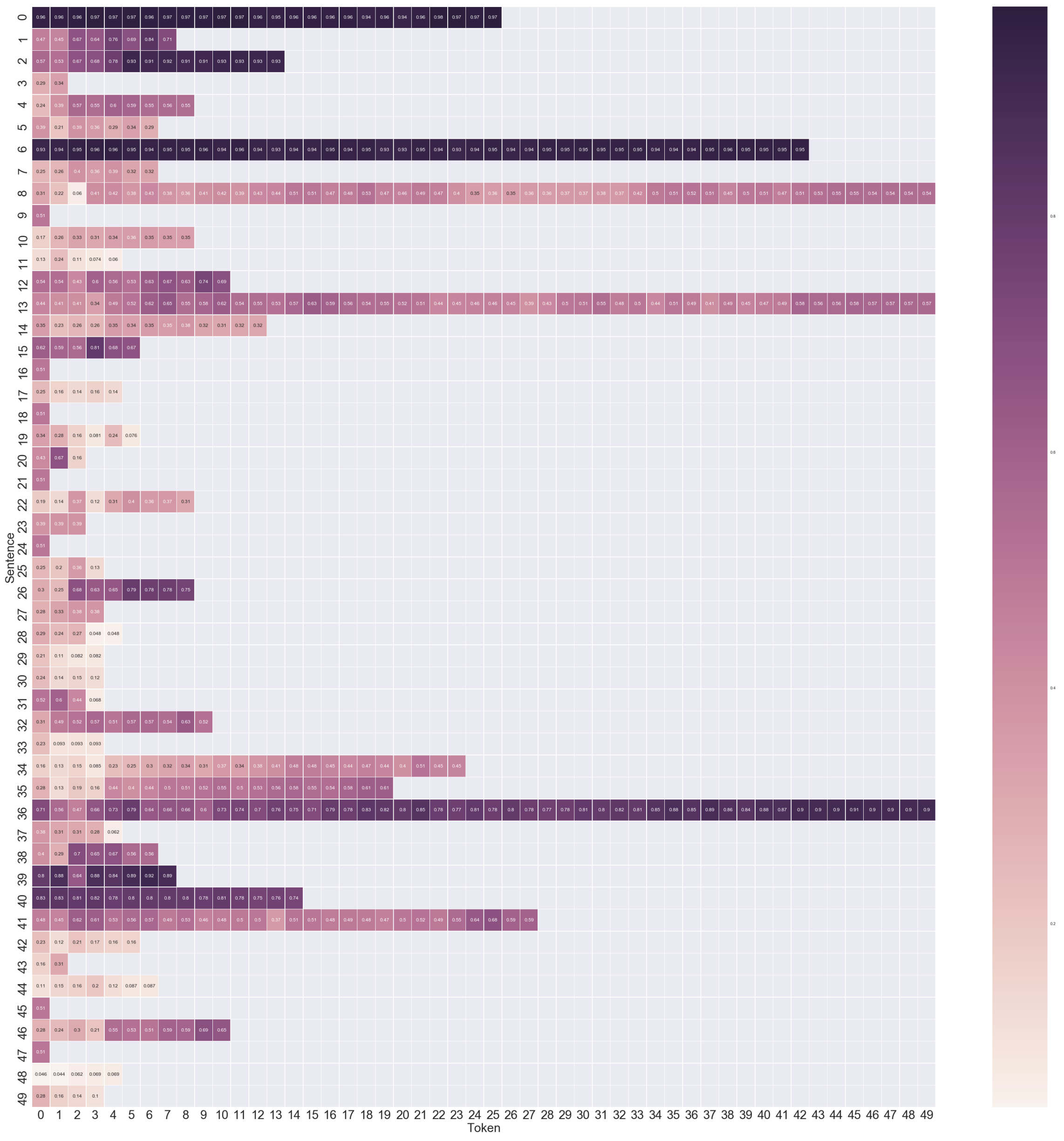


Figure A.2: Sentence rewards: Epoch 1 - Batch 1.

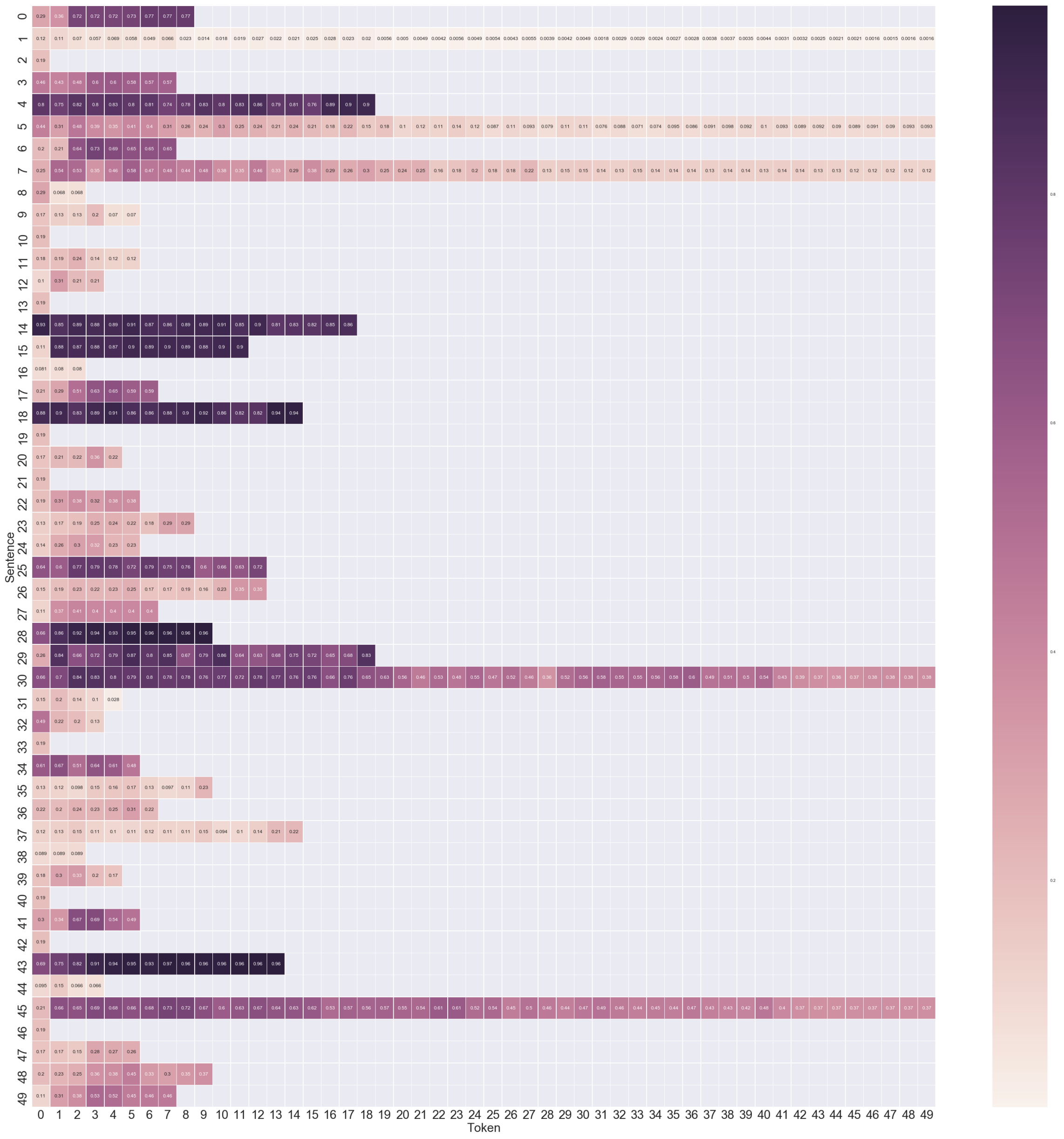


Figure A.3: Sentence rewards: Epoch 1 - Batch 20.

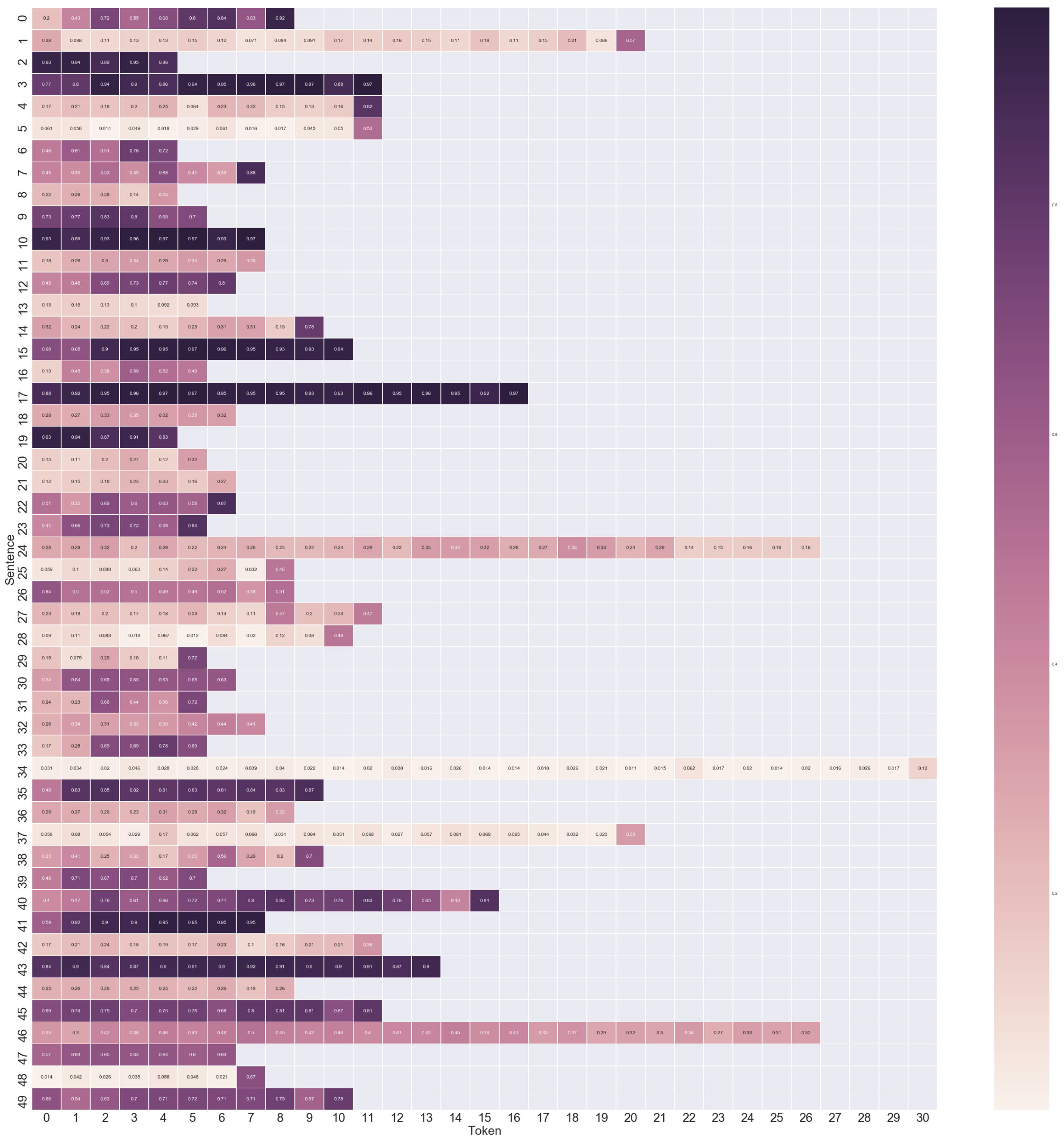


Figure A.4: Sentence rewards: Epoch 1 - Batch 62.