Meeting the 2020 Duolingo Challenge on a Shoestring

Tadashi Nomoto
National Institute of Japanese Literature
10-3 Midori Tachikawa 190-0014 Japan
nomoto@acm.org

Abstract

What is given below is a brief description of the two systems, called gFCONV and c-VAE, which we built in a response to the 2020 Duolingo Challenge. Both are neural models that aim at disrupting a sentence representation the encoder generates with an eye on increasing the diversity of sentences that emerge out of the process. Importantly, we decided not to turn to external sources for extra ammunition, curious to know how far we can go while confining ourselves to the data released by Duolingo (Mayhew et al., 2020). gFCONV works by taking over a pre-trained sequence model, intercepting the output its encoder produces on its way to the decoder. c-VAE is a conditional variational auto-encoder, seeking the diversity by blurring the representation that the encoder derives. Experiments on a corpus constructed out of the public dataset from Duolingo, containing some 4 million pairs of sentences, found that gFCONV is a consistent winner over c-VAE though both suffered heavily from a low recall.

1 Introduction

A major driver for our participating in the challenge was the curiosity to see whether recent approaches to sentence encoding with the variational auto-encoder (VAE) have any relevance to the generation of diverse sentences. (Bowman et al., 2016) were the first to explore the use of VAE in language generation. The work demonstrated that VAE provides a continuous code space for sentences, where any randomly picked data point in the space can be decoded to yield a coherent sentence, which is significant given that the conventional RNNs do not provide such a capability. The problem with VAE however, is that it has no mechanism to ensure that the meaning of the source sentence is passed over to the output, which often causes a sentence to be altered, or deformed beyond recognition. While VAE is a popular approach people turn to as a way to diversify sentences the model generates, no definitive answer has been found on how to control or tame what it spews out. A typical solution is to fuse a VAE code with the output of a regular sentence encoder, in order to encourage the decoder to output a sentence that retains some semantic features present in the source sentence (Gupta et al., 2017). Also noteworthy is a recent work by (Guu et al., 2018), who building on an idea similar to VAE, talk about modeling the distribution of cosine similarities between word vectors for the input and target. (Li et al., 2015) is something of an odd ball in the pursuit of the diversity in sentence generation. The authors argued that we could achieve the diversity by discouraging the decoder to select candidates that are similar to the input. A clear advantage they have over others is that their scheme does not involve any learning and is straightforward to implement.

The idea that one can view a latent representation as a sample drawn from some probabilistic distribution inspired people to explore its potential in a wide range of tasks and domains. (Miao et al., 2015), while working on document modeling, suggested that we use VAE as a way to get a compact representation for a document. (Fang et al., 2019) argued for using a sample based distribution over Gaussian distribution for a latent code to better express the holistic property of the source sentence.

In this work, we focus on two approaches, both based on VAE: one that attempts to achieve the diversity by generalizing the sentence representation produced by the encoder; and another which randomly perturbs the encoder's output during the sentence generation. We report here their respective performance on a test corpus we carved out of the official training data. For the final submission, we went along with the latter approach.

Proceedings of the 4th Workshop on Neural Generation and Translation (WNGT 2020), pages 129–133
Online, July 10, 2020. ©2020 Association for Computational Linguistics
www.aclweb.org/anthology/D19-56%2d

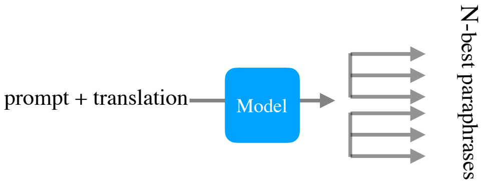

Figure 1: Translation as Paraphrase

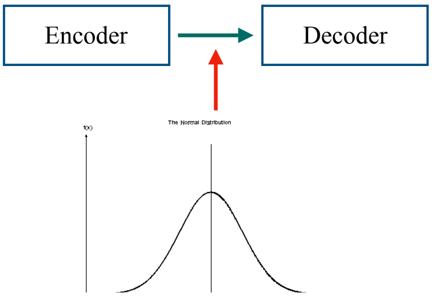

Figure 2: gFCONV

2 Translation as Paraphrase

Our effort revolved around two questions: (1) how best to incorporate likelihood scores of target translations that were provided as part of the training data, and (2) how *not* to rely on an external resource while building a solution. We wanted to know how far we can go using only the data made available to us at the competition, and nothing more. Our answer to the first question takes advantage of the fact that a set of translations associated with each English prompt are considered an equivalence class in the sense that if we take any pair from the set, we can substitute one for the other without significantly affecting its meaning. We may take the likelihood that a human picks a particular sentence (call it X) as a good translation for some prompt (P) as the probability of its being a paraphrase of some other sentence (say Y) from a group of possible translations of which X is part. The intuition here is that if X is more typical as a translation of P, it is more likely to serve as a paraphrase of whatever other way we may have to express P in the target language. Following this idea, we created training data by randomly sampling a pair of sentences (both in the same language) that appear as alternate translations for a given prompt in accordance with their popular rating. For each prompt, we sampled 2,000 pairs of translations (which may include pairs consisting of identical sentences), resulting in 4,601,000 training instances (which amount to 2,300 prompts plus those provided in the development and test set) (Mayhew et al., 2020).[1] We set aside 100 prompts as a private development set and another 100 for testing. We included in each training instance an English prompt as well as its translation in order to prevent paraphrases the algorithm generates from diverging from the meaning of the prompt (Table 1).

Figure 1 shows a schematic picture of how our approach works. We feed into the system a prompt and its translation which we assume to be given (via AWS, for example). Out comes its paraphrases (or translations in varied styles). The model we built is essentially one based on Fairseq's convolution to sequence architecture of the type called 'fconv_iwslt_de_en' (call it FCONV) which features 4 convolutional layers for the encoder and 3 for the decoder.[2] The embedding dimension for the input and output token was set to 256. We did not use pre-trained embeddings for either of the languages we dealt with. Neither did we make any architectural change to FCONV. We simply trained it as it was given. A departure comes in the testing phase. Following (Guu et al., 2018), we applied a Gaussian noise on the output of the encoder as it was sent to the decoder (Fig. 2).

$$u = E(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, k) \tag{1}$$

where x is an input and $E(x)$ is an output from applying an encoder E on x. u denotes an input to a decoder. A larger noise means a greater disruption in the latent representation coming from the encoder, which we hoped would lead to an increase in the diversity of sentences being generated. We randomly sampled a noise from a normal distribution with the mean set to 0 and the variance rang-

[1] For this year's challenge, we worked only on the English-Japanese track. We included both test and development sets as part of training data, as a way to prevent the algorithm from stumbling upon unknown tokens in the test set. We don't see this as much of a problem because each prompt in development and test sets carries no more than one translation, i.e. a training pair we get from the development and test set has a same sentence for both source and target. We made use of MeCab for tokenizing sentences in Japanese.

[2] https://github.com/pytorch/fairseq

Table 1: Training Instances. The source part of each input consists of two sections: the first section contains a sentence in an original language, followed by a translation in a target language, demarcated by a separator '@@@@.'

SOURCE	that apple is very big . @@@@ その 林檎 は 非常 に でかい です 。 i like to work . @@@@ ボク は 仕事 は 好き です 。 he drinks milk . @@@@ 彼 は 牛乳 を 飲み ます 。 what are her strengths ? @@@@ 彼女 の 長所 は なん でしょ う ？ what has she done ? @@@@ 彼女 は 何 を やり 終わっ た ん です か ？
TARGET	その りんご が とても でかい 。 私 は 働く の が 好き 。 かれ は ぎゅうにゅうを 飲み ます 。 何 が 彼女 の 強い 所 な の ？ 何 を 彼女 は 終わっ た の ？

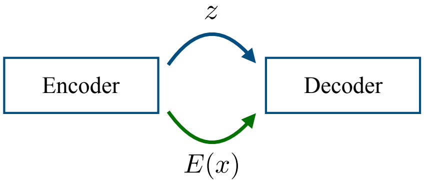

Figure 3: Conditional VAE

ing from 0 to 0.6. In what follows, we refer to the scheme as gFCONV.

We also looked at a conditional variational autoencoder (c-VAE), a close cousin of gFCONV for the sake of comparison. While both aim at building a latent representation that embraces the notion of uncertainty, c-VAE differs from the variance based approach in that it seeks to find a probabilistic distribution that defines a range of representations that the encoder churns out. In terms of formulae, this comes to the following (also see Fig. 3 for a visual intuition).

$$u = E(x) + r * z, \tag{2}$$

Here $z = \mu + \epsilon * v$ with $\epsilon \sim \text{Unif}[0, 1)$. μ and v are a mean and variance, defined as $\mu = g(x)$, and $v = f(x)$, respectively. x is an input, g and f are some arbitrary functions over x. $E(x)$ again denotes the output of an encoder. μ and v are learnable parameters, which means that they need to be trained to have them work. It is worth noting that gFCONV has no extra 'learnable' parameters. r is a hyper-parameter to be set manually, which determines the degree of contribution of z to a latent representation of x. We combine $E(x)$ and a representation sampled from a Gaussian distribution to build a final encoder output. Our decision to condition VAE on $E(x)$ is motivated by a frequent observation in the past literature that VAE is poor at preserving the meaning of the source sentence, often transforming it beyond recognition. Conditioning VAE on the input is a popular trick to discourage the algorithm from straying too far away from the source.

Implementation-wise, c-VAE was based on FCONV, from which we also built gFCONV. We kept all the hyper-parameter settings intact, e.g. the number of layers, the size and the number of filters, etc. We did not apply any scheduled annealing weight to the KL term in the loss function.

For gFCONV, we varied the variance parameter k (Eqn. 1) from 0.00 to 0.60 in increments of 0.05. For each value of k, we ran gFCONV on the test set 100 times, letting the model output 80 alternative translations for each prompt (Setting k to 0 reduces gFCONV to a vanilla FCONV). This had resulted in a pool of 8,000 candidates for a given prompt under a particular value of k. Out of which we retained only those that had a non zero similarity to gold translations by AWS.[3] We measured the similarity using LASER,[4] along with pre-trained word embeddings from FastText,[5] which LASER requires. We were interested to know how variance affected the performance, in particular how it contributed to improving the diversity.

[3] i.e. those found in the 'test.en_ja.aws_baseline.pred.txt' in the 'staple-2020-test-blind' directory.

[4] https://github.com/facebookresearch/LASER

[5] https://fasttext.cc/docs/en/crawl-vectors.html

Table 2: Variance vs. Performance. ‘P’ denotes precision, ‘R’ recall, and ’k’ variance. Numbers in red represent the baseline and those in blue the best performing system where we have a minimum divergence between Micro and Macro F1.

		UNWEIGHTED			WEIGHTED		
k	P	R	Micro F1	Macro F1	R	Micro F1	Macro F1
0.00	39.16	3.83	6.97	11.37	13.62	20.21	16.35
0.05	33.24	5.09	8.83	12.77	15.60	21.23	17.13
0.10	28.10	6.18	10.14	13.46	17.47	21.54	17.38
0.15	23.41	7.45	11.30	13.66	19.34	21.18	16.92
0.20	20.08	8.62	12.06	13.73	21.10	20.58	16.48
0.25	16.93	9.57	12.22	13.18	22.37	19.27	15.40
0.30	14.17	10.24	11.89	12.27	23.36	17.64	14.13
0.35	12.02	10.69	11.31	11.22	24.16	16.05	12.74
0.40	10.92	11.49	11.20	10.70	24.70	15.14	12.03
0.45	9.15	11.23	10.08	9.67	24.38	13.30	10.73
0.50	8.21	10.81	9.33	8.72	23.24	12.14	9.72
0.55	7.24	9.99	8.40	7.90	22.84	11.00	8.78
0.60	6.92	9.30	7.94	7.54	22.25	10.56	8.38

Table 3: Conditional VAE

		UNWEIGHTED			WEIGHTED		
r	P	R	Micro F1	Macro F1	R	Micro F1	Macro F1
0.0	39.16	3.83	6.97	11.37	13.62	20.21	16.35
0.1	24.06	7.11	10.97	14.84	19.41	21.49	18.08
0.2	22.55	7.98	11.78	14.81	20.16	21.29	17.87
0.3	18.28	7.72	10.86	14.28	20.09	19.15	17.10
0.4	17.09	7.58	10.50	13.12	20.21	18.52	15.69
0.5	15.68	7.70	10.33	12.73	20.08	17.61	15.31

3 Results and Discussion

Results are provided in Table 2. The numbers shown were produced using the official scorer. In the following discussion, we concentrate on unweighted scores as our interest here is in knowing how much we improved the raw recall under the current setup. Note that weighted scores do not shed light on the true diversity of sentences we have garnered.

Looking at Table 2, we see gFCONV gaining on a vanilla FCONV, whose performance is represented by the numbers at $k = 0.00$. At $k = 0.25$, we see the raw recall jumping from 3.83 to 9.57, Micro F1 from 6.97 to 12.22, and Macro F1 from 11.37 to 13.18. Compare the difference between Micro and Macro F1 at $k = 0.00$ and that we have at $k = 0.25$. The difference for the latter is much smaller. This suggests that under gFCONV, the performance is more stable across test items compared to the vanilla FCONV. A large divergence at $k = 0.00$ indicates wild ups and downs in performance, suggesting that the model is doing beautifully well on some but failing miserably on others. In contrast to Micro F1, Macro F1 is blind to how many candidate translations there are for each prompt, so may not give us an accurate picture of how the model is doing on each prompt.

As with gFCONV, we ran c-VAE on the test set 100 times, obtaining 100 distinct pools of candidate translations for each prompt.[6] We report in Table 3, figures that represent performance on all the results combined in the manner we described for gFCONV. We varied r (in Eqn. 2) from 0.1 to 0.5 in 0.1 increments. We observe that c-VAE is somewhat behind gFCONV (in terms of divergence between Micro and Macro F1), though performing well over the baseline (numbers in red). A large gap between (unweighted) Micro and Macro

[6] We generated 8,000 hypotheses for each prompt under a particular value of r, 80 at each round.

Table 4: Official Results. By W. Recall and W. F1, we mean Weighted Recall and F1.

Phase	Rank	Precision	W. Recall	W. F1
DEV	6/6	0.369	0.183	0.181
TEST	6/6	0.349	0.212	0.194

F1 again shows that the model suffers from a fluctuating performance, swinging wildly from one test item to another. The final submission for the official evaluation was prepared using gFCONV at $k = 0.10$, under the pseudonym 'darkside,' with the official results shown in Table 4.[7]

4 Conclusions

We discussed two approaches as a way to tackle the Duolingo Challenge. One is gFCONV, which takes over a pre-trained sequence model, intercepts and perturbs the output its encoder produces on its way to the decoder. Another is c-VAE, a conditional variational auto-encoder, which seeks the diversity by blurring the representation that the encoder derives. Either approach, it was found, outperformed the vanilla FCONV. We also noted a large discrepancy between Micro and Macro F1, suggesting that the models' performance is not even and fluctuates wildly from item to item. Moreover, there were some test prompts for which the models were not able to find any translations. We recognize that this is an area we need to scrutinize to further improve the performance. In the long run, it would be interesting to see if we can bring to the task recent developments in VAE such as (Bouchacourt et al., 2018).

References

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. 2018. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *AAAI 2018*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Le Fang, Chunyuan Li, Jianfeng Gao, Wen Jun Dong, and Changyou Chen. 2019. Implicit deep latent variable models for text generation. In *EMNLP/IJCNLP*.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2017. A deep generative framework for paraphrase generation. *CoRR*, abs/1709.05074.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055.

S. Mayhew, K. Bicknell, C. Brust, B. McDowell, W. Monroe, and B. Settles. 2020. Simultaneous translation and paraphrase for language education. In *Proceedings of the ACL Workshop on Neural Generation and Translation (WNGT)*. ACL.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural variational inference for text processing. *CoRR*, abs/1511.06038.

[7] We did not submit the version at 0.25 which turned out to be the best, due to its late discovery, which came well past the deadline.