Generative Adversarial Network for Pun Generation

Leena Dighole
lbd47

Amit Chawla
ac1910

Abhishek Bhatt
ab2083

Abstract

In this project, we will focus on the generative task of synthesizing a pun sentence given a pair of word senses. Our goal is to explore and reproduce the architecture discussed in the paper, Pun-GAN: Generative Adversarial Network for Pun Generation by Luo et al., (Luo et al., 2019) as our main reference. We also aim to replicate the presented results and experiment with some architecture modifications for possible improvements. Through this exercise, we hope to learn and possibly enhance the application of a GAN based approach to an exciting NLP task.

1 Problem

A homographic pun refers to the clever and amusing use of a word with two meanings, a very crucial aspect of human intelligence and natural language. The example, “I used to be a banker but I lost interest” fits this definition, since the word “interest” may be ambiguously interpreted as either curiosity or profits. A reasonable approach to pun generation could be through supervised language modeling. However, lack of a large-scale pun corpus with clearly labeled sentences for two word senses poses a challenge to effectively train such a language model. The proposed architecture in the later sections can provide a good way to generate pun sentences for supervised learning in downstream tasks.

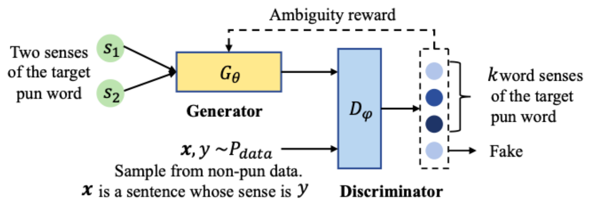
Ambiguity reward
Two senses of the target pun word
s_1
s_2
G_θ
Generator
D_φ
k word senses of the target pun word
Fake
$x, y \sim P_{data}$
Sample from non-pun data.
x is a sentence whose sense is y
Discriminator

Figure 1: Pun-GAN Architecture

2 Goal

The paper Pun-GAN: Generative Adversarial Network for Pun Generation(Luo et al., 2019) proposes the application of a Generative Adversarial Network(Goodfellow et al., 2014) into pun generation task. The generator can be any language model based architecture that can generate a pun sentence with ambiguous meanings for a given word. The discriminator is a word sense classifier, that discriminates a real sentence to its correct word sense class and labels a generated sentence as fake. In such a setting, the discriminator output can be used as an ambiguity reward to the generator, encouraging it to effectively generate pun sentences via reinforcement learning.

- The paper proposes a biLSTM generator implementation as described in (Yu et al., 2018). The discriminator is also an LSTM network. Our goal is to do an end-to-end implementation of the proposed architecture and replicate the presented results. Though the authors have made the codebase publicly available at `https://github.com/lishunyao97/Pun-GAN`, this would be a good exercise to enhance our understanding about the problem and limitations of the existing model to solve this multimodel RL+NLP based task.

- We can use the reproduced code implementation and results as a baseline to further experiment with some enhancements, for example joint training of discriminator and generator, or using self-attention/transformer blocks instead of LSTMs. We expect to come up with additional plausible enhancements while working on this project .

3 Achievability

- The generator(Yu et al., 2018) seq2seq model uses 2 LSTM layers of 128 units each and the discriminator consists of a single layer of bi-directional LSTM.
- As LSTM training is not parallelizable, we will need significant amount of time to train them. We already have access to Google Colab GPU capabilities. We are planning to use the Rutgers Ilab GPU resources for additional capacity as required.
- The authors have used the English Wikipedia corpus with one word sense using an unsupervised WSD tool to train the generator. In the interest of time, we are planning to use the same pre-massaged corpus data as used by the authors to train our model.
- Similarly, for the discriminator training we will be using the SemCor which is a manually annotated corpus for WSD, consisting of 226K sense annotations. The discriminator will be a multiclass classifier rather than traditional Binary classifier.
- The models will be implemented using Python3.6 and PyTorch 1.4.0 (as available on the ilab servers)
- For evaluation of the model, we will be using the pun dataset from SemEval 2017 task7 (Miller et al., 2017). The dataset consists of 1274 human-written puns where target pun words are annotated with two word senses.
- We will be using 3 metrics for evaluating the models performance: - unusualness is measured by subtracting the log-probability of training sentences from the log-probability of generated pun sentences. - diversity is measured by the ratio of distinct unigrams (Dist-1) and bigrams (Dist-2) in generated sentences.

4 Related Work

Pun or humor generation is an exciting NLP application with many researchers trying to come up with effective generative models. The challenges for pun generation include specialized language modelling and training with limited availability of labeled data-sets. The early researches were based upon generating a pun with some specific rules or templates. The paper (Petrović and Matthews, 2013) proposes a specific template based pun such as, "I like my X like I like my Y, Z". The X,Y,Z will be filled with the words having some similarity and uncommonness as well. The later improvements were done by Yu et al. in his proposed paper (Yu et al., 2018) which emphasizes on generating a sentence with two word senses with the help of a joint decoding algorithm.
Training with pun labeled datasets is hard in practice and hence the earlier papers follow more rule-based approaches. The generator of Pun-GAN can be simply be any trainable pun generation model. For simplicity we will be using the pun-GAN generator model used in paper (Yu et al., 2018). The biggest challenge in GAN text generation is the sampling process not being differentiable. This makes it difficult to propagate the gradients from the discriminator to the generator. Thus for training the generator we will be using the ambiguity reward policy gradient method of Reinforcement learning. We will be using this reward policy idea from recent researches like the one in this paper(Guo et al., 2017). This paper also expands the traditional binary discriminator approach to the multi class classifier for the better evaluation of ambiguity and diversity of the generated pun sentences.

References

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information.

Fuli Luo, Shunyao Li, Pengcheng Yang, Lei li, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. Pun-gan: Generative adversarial network for pun generation.

Tristan Miller, Christian F Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232, Sofia, Bulgaria. Association for Computational Linguistics.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1660, Melbourne, Australia. Association for Computational Linguistics.