Assignment 4

Due on 30 May, 2019 (23:59:59)

Click here to accept your Assignment 4

Introduction

Text Generation is the task of generating a chunk of text automatically. This text could be either a sentence, story, a poem, a novel etc. It is possible to create a poet robot or a writer robot that generates poems with a text generator!

Text Generation can be be modeled using deep learning models such as Feed-Forward Neural Networks (FNN) [1] and Recurrent Neural Networks (RNN) [3]. For the text generation task, FNNs are trained on a very large corpus to predict the next word as a bigram language model. Once the model is trained, it is straightforward to generate a new text by iteratively predicting the next word as a n-gram language model.

In this assignment, we will implement an n-gram (bigram) neural language model using feed forward neural networks for Text Generation by using DyNet[1] deep learning library.

0.1 Task 1: Build Feed-Forward Neural Network Language Model (FNN)

An n-gram language model with a Markov assumption is defined as follows for the words in a sentence:

$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \tag{1}$$

In the first assignment, you implemented a count-based language model to predict the n-gram probabilities. Now, you will apply a prediction-based neural language model using a feed-forward neural network like the one proposed by Bengio et al.[1].

Here, w_0 and w_{T+1} are the start and end tokens of a word sequence (sentence). Each token (word) in your training corpus will be represented as a one-hot vector. In order to predict $P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1})$, the one-hot vector in the history w_{t-1} feed into the feed-forward neural network to predict the probability of the next word w_t.

The feed-forward neural network is defined mathematically as follows:

$$y = U \cdot f(W \cdot x + b) + d \tag{2}$$

[1] http://dynet.io/

Where, $W \in \mathbb{R}^{H \times W}$ and $U \in \mathbb{R}^{W \times H}$ are the weight matrices between input-hidden layers and hidden-output layers respectively. Here, H is the dimension size of the hidden layer and W is the vocabulary size which is equal to the one-hot vector's dimension. b and d are the bias terms and weights of b and d are defined by $b \in \mathbb{R}^H$ and $d \in \mathbb{R}^W$ respectively. f is the *tanh* activation function in the hidden layer.

The i-th element of the output vector y is the unnormalized conditional probability of the word with index i in the vocabulary. To convert it into a proper probability distribution, we apply a softmax function in the output layer to turn the output scores into probabilities in the output layer.

$$P(v_i|w_1^{t-1}) \approx P(v_i|w_{t-n+1}^{t-1}) = \frac{e^{y(v_i, w_{t-n+1}^{t-1})}}{\sum_{j=1}^{k} e^{y(v_j, w_{t-n+1}^{t-1})}}, i = 1, 2, \cdots, k \tag{3}$$

where $y(v_i, w_{t-n+1}^{t-1})(i = 1, 2, \cdots, k)$ is the i-th element of output vector y for the input w_{t-n+1}^{t-1}, and v_i is the i-th word in the vocabulary.

Training of neural network language models is usually performed by maximizing the log-likelihood of the training data:

$$L = \frac{1}{T} \sum_{t=1}^{T} log(P(w_t|w_1^{t-1}; \theta)) \tag{4}$$

The recommended learning algorithm for neural network language models is stochastic gradient descent (SGD) method using backpropagation (BP) algorithm. A common choice for the loss function is the cross entroy loss which is equal to the negative log-likelihood here.

As for the training, you need to iterate over the corpus in windows of size 2, and you will apply the backpropagation for each bigram in the corresponding window. This will be repeated for many epochs to optimize the weights in your model. Eventually, the log-likelihood of the training data L given in Equation 4 will be maximized at the end of the training.

0.2 Task 2: Poem Generation

Once you build your feed-forward neural network language model, you will train it to use for poem generation. To generate a new poem, you need to start with start token. Then, you will predict one word at each time using the previous word and feeding this word as input to the neural network in the next time step. You will generate 5 new poems, where the number of lines of each poem will be taken from the user.

Configuring neural networks is difficult because the parameters can be different for each task and for each dataset. Try different values for parameters and report your results.

0.3 Task 3: Evaluation

After you generate your poems automatically by using your trained feed-forward neural network language model, you will compute the perplexity of each generated poem.

Dataset

You will use a poem dataset called Uni-Modal Poem [2], which is a large poem corpus dataset that involves around 93K poems. UniM-Poem is crawled from several publicly online poetry web-sites, such as Poetry Foundation, PoetrySoup, best-poem.net andpoets.org.

Submit

You are required to submit all your code (*all your code should be written in **Python** (Python 3.5)* long with a report in latex format (template). The codes you will submit should be well commented. Your report should be self-contained and should contain a brief overview of the problem and the details of your implemented solution. You can include pseudocode or figures to highlight or clarify certain aspects of your solution.

- report.pdf
- code/ (directory containing all your codes as Python file .py)

Grading

- Code (80 points): Task 1: 50 points, Task 2: 20 points, Task 3: 10 points
- Report: 20 points

Note: Preparing a good report is important as well as the correctness of your solutions!

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the

former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[2] Bei Liu, Jianlong Fu, Makoto P Kato, and Masatoshi Yoshikawa. Beyond narrative description: Generating poetry from images by multi-adversarial training. *arXiv preprint arXiv:1804.08473*, 2018.

[3] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.