Assignment 4

Due on 01 June, 2020 (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.

Text Generation can be be modeled using deep learning models such as Feed-Forward Neural Networks (FFNN) [1] and Recurrent Neural Networks (RNN) [3]. For the text generation task, FFNNs 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) level FNN for Text Generation by using DyNet[1] deep learning library.

0.1 Feed-Forward Neural Network Language Model (FNN)

The idea behind FNN language model with n-gram is that words in a word sequence statistically depend on the words closer to them, only $n-1$ direct predecessor words are considered when evaluating the conditional probability, this is:

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

In the given neural network, each words have pretrained vector representations [4]. The objective of the model is to find the parameters that minimize the perplexity of the training dataset. The model eventually learns the distributed representations of each word and the probability function of a sequence as a function of the distributed representations. The Neural model has a hidden layer with tanh activation and the output layer is a Softmax layer. The out of the model for each input of $(n-1)$ prev word indices are the probabilities of the $|V|$ words in the vocabulary.

The architecture of the model is given in Figure 1.

The y_i are the unnormalized log-probabilities for each output word i, computed as follows, with parameters b, W, U, d and H:

$$y = b + U \cdot tanh(d + Hx) \tag{2}$$

[1] http://dynet.io/

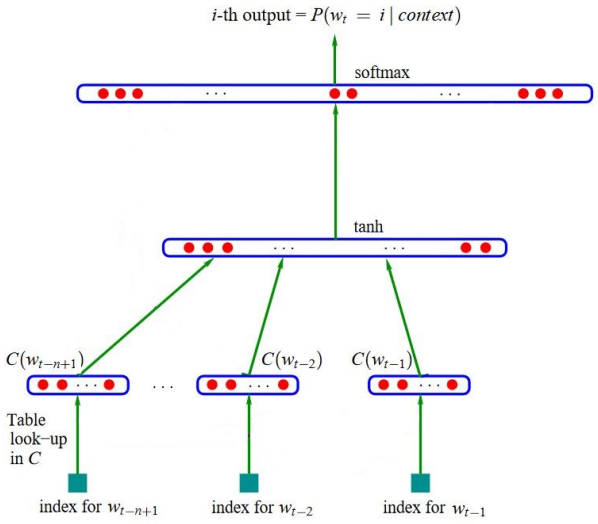

Figure 1: Feed-forward neural network language model

where the hyperbolic tangent tanh is applied element by element, and x is the word features layer activation vector, which is the concatenation of the input word features from the matrix C:

$$x = C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1}) \tag{3}$$

Let h be the number of hidden units, and m the number of features associated with each word (m is the dimension of the pretrained word vector). The free parameters of the model are the output biases b (with $|V|$ elements), the hidden layer biases d (with h elements), the hidden-to-output weights U (a $|V| \times h$ matrix), the hidden layer weights H (a $h \times (n-1)m$ matrix), and the word features C (a $|V| \times m$ matrix):

0.2 Task 2: Poem Generation

Once you build your FNN 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 to the input of the next time. You will generate 5 new poems, where the number of lines in 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 FNN 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. For the pretrained word vectors, GloVe [4][2] 6B word embeddings will be used (You can try different dimensions).

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

> **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.

[2] GloVe

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.

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.