Natural Language Understanding
Assignment - 2

Saley Vishal Vivek
vishalsaley@iisc.ac.in

Abstract

The assignment involves following tasks

1. To train and evaluate neural Language Models on given copora (gutenberg).
2. Select the best model after evaluation to generate tokens.

1 Preprocessing

The corpora includes two different corpus: **gutenberg** and **brown** . Each corpus consist of multiple text files which include the natural text. Following preprocessing tasks are performed on each of the corpus:

1. Read each file sentence by sentence.
2. For each sentence, add start and end of sentence markers appropriately based on the model that is to be used.
3. Divide this list of sentences into three sets: *train*, *test* and *devset*.
4. Store these sets as individual files.

2 Language Models

Two neural networks textbfcharacter-level and textbfword-level are used. Each of these model are trained on a subset of given corpora and are evaluated against the test set. These results are then compared based on perplexity to generate new tokens from the model.

2.1 Unknown Handling

To handle words which are not in the vocabulary, less frequent words (words which occur only once) are replaced but ¡unk¿ token. Then the corpus is rebuilt with this replacement and used for training. Also for testing is word is not already present in vocabulary, it is replaced by ¡unk¿.

2.2 Smoothing

The outputs of the neural models are treated with softmax to generate probabilities which ensures that probabilities are never zero. This takes care of Smoothing in the models.

3 Training the networks

3.1 Embeddings Learning

For training the neural networks, embeddings are used. These embeddings are learned along with the training of the neural network. For learning these embeddings, a table is created whose size is set up to be equal to hidden layer size of the neural network and during backpropagation, this table is updated to capture context of the word under considertion. Embeddings here try to capture the past context of the word as the model used is **Unidirectional Recurrent Network**.

3.2 Network Architecture

A multi-level LSTM network is used to learn the Language Model. Each of the layer contain 700 neurons thus the context size is of 700 which also the embeddings size. Following are the configurations for character-level and word-level language Model

1. Character-level: One LSTM cell, single layer, 700 neurons, with time-step (past context length) of 8 units. Smaller length is chosen for character-level encoding to complement word size in the corpus.

2. Word-level: Two LSTM cells, two layer network with 700 neurons in each layer. Average sentence size in train data is around 26. To complement that time-step of 32 is chosen.

Table 1: Perplexity

Model	Train	Test
Character Level	54	87
Word Level	544	455

3.3 Loss function

Considering traditional Language Model setting, the outputs from both the neural networks are treated as probability distribution over vocabulary of respective models. Thus a softmax is applied to final output to obtain proper probabilities. Based on this interpretation, a cross entropy loss is used as loss function which is given by.

$$loss(B) = - \sum_{\forall x_i \epsilon B} t_{ij} * ln(y_{ij}) \tag{1}$$

where B is the batch input provided to LSTM cells, $t_{ij} = 1$ for actual target word zero otherwise and y_{ij} is the probability of target predicted by the neural network. The output is considered only after completion of all the time-steps.
Batch size is set 1000 for character-level network and set to 50 for word level network.

3.4 Run Lengths

Based on multiple runs, it is identified that more time is spent on training the network more the performance of the network. Thus empirically character-level network is trained for 15 epochs while word-level network is trained for 20 epochs. These many iterations brings down the loss to a accpetable lower values in both the models.

4 Evaluation

Perplexity results of the models are as shown in table 1. This is very large compared to traditional trigram model (39). This is due to limited training of LSTMs.

4.1 Analysis

Character-level model appears to be performing better on given corpus. This can be justified as first, since character vocabulary is smaller compared to word vocabulary, the conditional probability mass is distributed more across all the elements. Second, character-level sequence appears to be more prominent in given corpus than word-level

5 Credits

The code is inspired by http://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/