CS 6320.002: Natural Language Processing Fall 2019

1 Baseline System – 20 points

Writeup Question 1.1: Write 1-2 paragraphs describing your chosen baseline. What paper was it published in? What machine learning algorithm does it use? What kind of preprocessing does it require? What are the features?

Our baseline System/Model consisting of two Recurrent Neural Networks – Encoder and Decoder. One encodes sequence of symbols into fixed -length vector representations and other decodes this representation back to sequence of symbols. The two RNN models are trained to maximize the conditional probability of output sentence given the input.

We use Recurrent Neural Network machine learning algorithm. The network hidden layers have the loop where the output or cell state at each step becomes the inputs to the next step, hence the name. This recurrence forms a memory. It allows contextual information to flow through the network and act as input/output at any of the steps.

Load the data (input contains the English and corresponding translated French sentences). The data is cleaned by converting to lower case and splitting over spaces. The cleaned data is transformed to vectors and dictionary. There are no features.

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
- Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio
(Submitted on 3 Jun 2014 (v1), last revised 3 Sep 2014 (this version, v3))

Writeup Question 1.2: Give step-by-step instructions for how to run your baseline system. What libraries are needed? What commands are used to run it, with what arguments, in what order?

Our baseline model is RNN which uses Encoder and Decoder (2 models) for machine translation. The model requires 2 files of input and output data. Input data is English sentences and the output data is the French translation of English sentences. These 2 must be put in a folder named data and then the model converts the data into dictionaries and vectors and save it in the same folder.
Further, the model takes data inputs from these dictionaries and vectors which it created in the first step.
We need to set some hyper parameters, which we chose as:
max_length = 20 #max number of words per sentence
num_epochs = 10 #number of epochs for the model to run
vocab_size = 15000 #total vocab size
All these are defined in main.py. As the model trains, it uses Encoder.py and Decoder.py and saves these 2 models recurrently for each epoch.
These 2 models are later called in the eval function. So, we have a total of three models:
Baseline.model
Encoder.ckpt
Decoder.ckpt

The Baseline.model calls the other two when it evaluates the test data.

To run the baseline model,

- keep all the supporting .py files in the same folder and
- provide the path of the input and output data (training set)
- then run the model from main.py

2 Trained Model – 10 points

Writeup Question 2.1: Write 1-2 paragraphs describing the training process. What learning scheme did you use (unsupervised, semi-supervised, or supervised)? Where there any hyperparameters, and if so, how did you tune them? Did you run on a CPU or GPU? How long did it take?

Our baseline model is RNN which uses Encoder and Decoder (2 models) for machine translation. We have used Supervised learning where 2 sets of training data (English sentences and their French translated sentences) are used to train the model. The model uses Encoder and decoder based on RNN to train the model. Our baseline model calls these two models.
The hyper-parameters used are:

- max_length = 20 #max number of words per sentence
- num_epochs = 10 #number of epochs for the model to run
- vocab_size = 15000 #total vocab size
- hidden-size = 100 #number of neurons used in encoder and decoder
- embedding_size=2000 #size to embed the input data

Various combinations of epochs, hidden size and embedding used have been used to keep the training time and efficiency of the model optimised.

We ran the model on CPU and it took more than 10 hours to train the data.
When we used the entire training set (more 2 million sentences) it took forever to train the model and thus we have reduced the size of the training data to 4000 sentences. Training the model on CPU (as we didn't have any GPUs) took a lot of time and hence the reduction in the training size.

3 Evaluation – 20 points

Writeup Question 3.1: Give step-by-step instructions for how to run evaluate.py. What arguments does it take?

- Load the saved model (Baseline.model which in turn loads the encode and decode models).
- The sentences from test data corpus are fed to function sentence_to_vec line by line to generate the vectors. And finally the vectors are translated back to sentences in other language using the trained model by calling model.eval(vecs)
- Output of model.eval(vecs) is sent to the vec_to_sentence function to finally print the translated sentences.

Writeup Question 3.2: Write 1-2 paragraphs describing the evaluation. What was the performance of your baseline system? Is that good or bad? Why is the metric appropriate for your task? If there are other metrics commonly used for the task, why did you choose this one?

We are evaluating our baseline model against the 2008 test data for europarl.

Team Members:

Vinit Kumar - vxk170008
Aniket Ashok - axa170068
Suraj Raghavendra Vadvadgi - srv180000