CSCI544: Homework Assignment №2

Due on September 30, 2021 (before class)

This assignment is an extension to HW assignment 1 on sentiment analysis. Please follow the instructions and submit a zipped folder containing:

1. A PDF containing a Jupyter Notebook response to the assignment. Your Jupyter Notebook should contain both code and text cells with sufficient comments such that the reader can understand your solution as well as your responses for some of the questions. On the Jupyter notebook, please print the requested values, too. If it is more convenient, you can also submit a PDF similar to assignment 1, i.e., initially explaining your solution and then merge a Jupyter notebook.

2. You also need to submit your Jupyter Notebook separately in .ipynb format such that it can be easily executed. Please include the version of required dependencies. You can consider that data.tsv is a raw dataset in the current directory that your notebook should read and perform all the required steps and generate the desired outputs.

The preferred library to implement neural models is PyTorch but TensorFlow (or Keras) is also acceptable. Please name your zipped file "HW2-YourFirstName-YourLastName-AAA.zip", where "AAA" is either "Py" or "TF" depending on the library you have used. You can also use publicly available implementations in portions of your solution but you need to include proper reference to your resources, e.g., url to the page that you used as a reference, books, etc.

1. Dataset Generation (5 points)

We will use the Amazon reviews dataset used in HW1. Load the dataset and build a balanced dataset of 250K reviews along with their ratings (50K instances per each rating score) through random selection. Create ternary labels using the ratings. We assume that ratings more than 3 denote positive

sentiment (class 1) and rating less than 3 denote negative sentiment (class 2). Reviews with rating 3 are considered to have neutral sentiment (class 3). You can store your dataset after generation and reuse it to reduce the computational load. For your experiments consider a 80%/20% training/testing split.

2. Word Embedding (30 points)

In this part the of the assignment, you will learn how to generate two sets of Word2Vec features for the dataset you generated. You can use Gensim library for this purpose. A helpful tutorial is available in the following link:
`https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html`

(a) (10 points)

Load the pretrained "word2vec-google-news-300" Word2Vec model and learn how to extract word embeddings for your dataset. Try to check semantic similarities of the generated vectors using two examples of your own, e.g., $King - Man + Woman = Queen$ or $excellent \sim outstanding$.

(b) (20 points)

Train a Word2Vec model using your own dataset. Set the embedding size to be 300 and the window size to be 11. You can also consider a minimum word count of 10. Check the semantic similarities for the same two examples in part (a). What do you conclude from comparing vectors generated by yourself and the pretrained model? Which of the Word2Vec models seems to encode semantic similarities between words better?

3. Simple models (20 points)

Using the Word2Vec features that you can generate using the two models you prepared in the Word Embedding section, train a perceptron and an SVM model similar to HW1 for class 1 and class 2 (binary models). For this purpose, you can just use the average Word2Vec vectors for each review as the input feature ($x = \frac{1}{N}\sum_{i=1}^{N} W_i$ for a review with N words). To improve

your performance, use the data cleaning and preprocessing steps of HW1 to include only important words from each review when you compute the average $x = \frac{1}{N}\sum_{i=1}^{N} W_i$. Report your accuracy values on the testing split for these models for each feature type along with values you reported in your HW1 submission, i.e., for each of perceptron and SVM, you need to report three accuracy values for "word2vec-google-news-300", your own Word2Vec, and TF-IDF features.

What do you conclude from comparing performances for the models trained using the three different feature types (TF-IDF, pretrained Word2Vec, your trained Word2Vec)?

4. Feedforward Neural Networks (25 points)

Using the features that you can generate using the models you prepared in the Word "Embedding section", train a feedforward multilayer perceptron network for sentiment analysis classification. Consider a network with two hidden layers, each with 50 and 10 nodes, respectively. You can use cross entropy loss and your own choice for other hyperparamters, e.g., nonlinearity, number of epochs, etc. Part of getting good results is to select good values for these hyperparamters.

You can also refer to the following tutorial to familiarize yourself:

`https://www.kaggle.com/mishra1993/pytorch-multi-layer-perceptron-mnist`

Although the above tutorial is for image data but the concept of training an MLP is very similar to what we want to do.

(a) (10 points)

To generate the input features, use the average Word2Vec vectors similar to the "Simple models" section and train the neural network. Train a network for binary classification using class 1 and class 2 and also a ternary model for the three classes. Report accuracy values on the testing split for your MLP model for each of the binary and ternary classification cases.

(b) (15 points)

To generate the input features, concatenate the first 10 Word2Vec vectors for each review as the input feature ($x = [W_1^T, ..., W_{10}^T]$) and train the neural

network. Report the accuracy value on the testing split for your MLP model for each of the binary and ternary classification cases.

What do you conclude by comparing accuracy values you obtain with those obtained in the "'Simple Models" section (note you can compare the accuracy values for binary classification).

5. Recurrent Neural Networks (20 points)

Using the features that you can generate using the models you prepared in the "Word Embedding" section, train a recurrent neural network (RNN) for sentiment analysis classification. You can refer to the following tutorial to familiarize yourself:

`https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html`

(a) (10 points)

Train a simple RNN for sentiment analysis. You can consider an RNN cell with the hidden state size of 50. To feed your data into our RNN, limit the maximum review length to 50 by truncating longer reviews and padding shorter reviews with a null value (0). Train the RNN network for binary classification using class 1 and class 2 and also a ternary model for the three classes. Report accuracy values on the testing split for your RNN model.

(b) (10 points)

Repeat part (a) by considering a gated recurrent unit cell.

Note that in total, you need to report accuracy values for:

2 (number of Word2Vec models) * (2 (Perceptron + SVM) + 2 (binary/ternary settings) (2 (FNN) + 2 (RNN))) = 2 (2+ 2 (2 + 2)) = 20 cases.