CSE354 Sp20 - Assignment 3 Description

Natural Language Processing

Stony Brook University
CSE345 - Spring 2020

Goals.

- **Implement a sentiment analysis system "from scratch".**
- **Experience working with dense word embeddings**
- **Implement a user factor adaptation for sentiment analysis.**
- **Utilize RNNs and Transformers.**
- **Gain experience with a deep learning framework (tensorflow or pytorch).**
- **Encourage original thought or experimentation in NLP model design.**

General Requirements. You must use Python version 3.6 or later, along with Tensorflow 1.15 or 2.1, or Pytorch 1.4.0.

Python Libraries. The only data science, machine learning, or statistics libraries that you may import are those that are listed in this assignment. Of these libraries, you may not use any subcomponents that specifically implement a concept which the instructions indicate you should implement (e.g. a complete RNN (though LSTM or GRU Cells are OK) or complete transformer networks, logistic regression, sentiment analysis,). Other Python default, non-data science libraries (e.g. sys, basic IO, re, random,csv) may be used -- ask if unsure.

```
import numpy as np

from sklearn import Ridge //just like L2 regularized logistic regression

from sklearn import PCA
import scipy.stats as ss//for distributions

from gensim.models import Word2Vec
import tensorflow as tf
import pytorch //not good to include both pytorch and tensorflow
```

You may use pre-defined layers within tensorflow or pytorch packages but not entire prebuilt archictures (i.e. RNN regression) you need to explicitly specify each layer with your approach.

Submission.

1. **Place all of your code in a single file,** `a3_<lastname>_<id>.py`, **and you must submit it to: Blackboard under assignment 3. All three stages of your code should run with:**
   python3 `a3_LAST_ID.py ‘food_train.csv’ 'food_trial.csv'`

2. **Submit your predictions for the shared task data (see below) to the Class Kaggle competition (won't be available for late submissions). Indicate your username in your blackboard submission.**

Academic Integrity. Copying chunks of code or problem solving answers from other students, online or other resources is prohibited. You are responsible for both (1) not copying others work, and (2) making sure your work is not accessible to others. Assignments will be extensively checked for copying of others’ work. Problem solving solutions are expected to be original using concepts discussed in the book, class, or supplemental materials but not using any direct code or answers. Please see the syllabus for additional policies.

Data

You will be working with review data from 3 categories of Amazon products. Each files comes in csv with the following columns:

id -- a unique id from the review

rating -- a number between 1 and 5 corresponding to review rating (5 is best)

vote -- how many votes the review got for being useful (not available for test data)

asin -- the Amazon product id

user_id -- a unique id for the given reviewer

reviewText -- the text of the review.

Non-shared task data:

1. food: train   trial -- food product reviews (small: 5280 train, 458 trial)
2. music train   trial -- digital music reviews (medium: 17,306 train, 1558 trial)

Shared task data:

3. musicAndPetsups train   trial -- music + pet supply reviews (large: 34,481, 3,251)
   test -- the test data. You must submit predictions for each of these, using your system from stage 3 (or whichever you find strongest), to the shared task.

Dataset 3 which is half dataset 2 and half pet supply data will be run as a Kaggle class shared task, whereby a separate test-set will be with-held until the shared task is complete. In the meantime, you are given "trial" data to evaluate how well you are doing. The final part of the assignment will evaluate your system on the shared task.

Stage I: Basic Sentiment Analysis with Word2Vec (35 Points)

Your objective is to implement basic sentiment analysis, utilizes word2vec embeddings of all words within product reviews. Specifically, you will predict the "rating" column from only the "reviewText" column. Predictiving sentiment on a scale is sometimes referred to as "ratings prediction" which we will use henceforth for this task.

Input: Your code (for all stages) should take **two** command line parameters for the review dataset location (the first being the training data, the second being the data to test on).

```
python3 a3_LAST_ID.py ‘food_train.csv’ 'food_trial.csv'
```

Task Steps:

1.1 Read the reviews and ratings from the file

1.2 Tokenize the file. You may now use any existing tokenizer. Here is a suggestion: https://github.com/channel960608/happiestfuntokenizing

1.3 Use GenSim word2vec to train a 128-dimensional word2vec model utilizing only the training data.
Note: you will need to handle OOV yourself if you want your model to work for OOV words.

1.4 Extract features: utilizing your word2vec model, get a representation for each word per review. Then average these embeddings (using mean) into a single 128-dimensional dense set of features.

1.5 Build a rating predictor using L2 *linear* regression (can use the SKLearn Ridge class) with word2vec features.

1.6 Run the predictor on the second set of data and print both the mean absolute error and then Pearson correlation between the predictions and the (test input) set.

Checkpoint Output: Your code should output the MAE and Pearson product-moment correlation (can call scipy.stats for this) on the provided file for the trial dataset (it should work with other data in the same format as well).

Also, if these ids exist in the provided trial csv, then print the prediction and true value for them:

food: 548, 4258, 4766, 5800

music: 329, 11419, 14023, 14912

Stage II: User-Factor Adaptation (30 Points)

Your objective is to upgrade your rating predictor to utilize user-factor adaptation to improve prediction (see Lynn reading and slides 20 -- 42 of human-centered NLP for background on this method).

Task Steps:

2.1 Grab the user_ids for both datasets.

2.2 For each user, treat their training data as "background" (i.e. the data from which to learn the user factors) in order to learn user factors: average all of their word2vec features over the training data to treat as 128-dimensional "user-language representations".

2.3 Run PCA the matrix of user-language representations to reduce down to just three factors. Save the 3 dimensional transformation matrix (V) so that you may apply it to new data (i.e. the trial or test set when predicting -- when predicting you should not run PCA again; only before training).

2.4 Use the first three factors from PCA as user factors in order to run user-factor adaptation, otherwise using the same approach as stage 1.
Note: You can choose whether to re-insert the original features as well: It is easier to implement if so. If not, you need to make sure you user factors and thus you will either have a 384 dimensional feature vector per document or 512, depending on if you reinsert the original features; Reinserting the original features is theoretically not necessary but it can help with scaling).

Checkpoint Output: In addition to the previous checkpoint, your code should output the MAE and Pearson product-moment correlation for the trial datasets again.

Also, if these ids exist in the provided trial csv, then print the prediction and true value for them:

food: 548, 4258, 4766, 5800

music: 329, 11419, 14023, 14912

note: food dataset may not have background for all trial users.

Your MAE and Pearson r should either be the same (i.e. within 2%) of your previous results or better.

You may also go ahead and try submitting to the class shared task at this point, though your final submissions should be made after you complete stage III (you may submit once per hour to see how you stack up versus others).

Stage III: Deep Learning (35 Points)

Your objective here is to build the most accurate rating predictor possible using either Recurrent Neural Networks or Transformer Networks.

Task Requirements: You must use either a recurrent neural network or a transformer as part of your solution.

Task Steps:

3.1 Start with word2vec embeddings *per word* -- these may already be in memory from stage 1. (optionally, you may use an improved or higher dimensional version for these steps. Make sure stage 1 still uses the prescribed version though.)

3.2 Input the word2vec embeddings as the input sequence to either:

1. a Recurrent Neural Network (LSTM or GRU) or
2. a Transformer Sequence Network (TSN)

(Only need to do one of the two. It's easier to make mistakes with RNNs)

3.3. Add a linear regression output layer on top of your RNN or TSN to output a continuous valued ratings score. Note: linear regression uses "mean squared error" as the cost function rather than cross-entropy loss.

3.4. Add the user factors from stage 2 to your deep learning module. There are multiple ways to do this (you only need to do 1; doing multiple is ok):

- Add the factor values as additional tokens before the '<s>' during input.
    - would need the factors to fill 128 dimensions
- "Adapt" the output of the RNN or TSN to the factors (i.e. if there was 128 dimensions in your hidden state and you had 3 factors, you would turn that into 128*3 = 384 features as input to the final linear regression).
- (advanced) Use the factor values as a query within an attention layer within your network.

3.5. Submit your system to the class Shared Task system to receive evaluation over part of the held-out data. Information to do this is at the link below. Optionally, continue to improve your system to try to attempt to have the most accurate system in the class (top 5 will receive extra credit on the assignment and bragging rights).

Checkpoint Output:

Submit your best system to the class shared task (before the deadline, you can make one submission per hour but must designate 2 submissions as those to be counted for the final competition. Results over a partial set of the test data will be made public after each submission.) **note:** Recommend submitting to the shared task early