20-00-0546-iv Foundations of Language Technology

Homework 9
Solutions Text classification

22. December 2020

Please send Python Notebook files (.ipynb) for programming parts and plain text or PDF for essay questions. In case your submission consists of several files, compress these to a zip-file. Indicate clearly which submission corresponds to which question. Include comments in your program code to make it easier readable. The deadline for the homework is **28.01.2021 18:00 CET**.

9.1 Homework

In this homework, you should get familiar with three techniques for classification: (i) Multi-Layer Perceptrons with static embeddings as input representations, (ii) Naive Bayes with your self-defined feature sets, and (iii) BERT, the recent state-of-the-art in deep learning, based on contextualized representations. Note that only part (i) will be graded. Parts (ii) and (iii) are for your own self-study.

Many machine learning algorithms have hyperparameters, for example the number of layers used in a deep neural network. You should experiment with different values for your hyperparameters and print out the performance for the different values. By such trial and error, you also learn which hyperparameters work well and you may benefit from this knowledge for future tasks.

Another issue is the correct evaluation of your approaches: flaws in the evaluation may invalidate your research. For example, a basic requirement is that you train your methods on the one set and evaluate them on a distinct set. The development / test set should be large enough to obtain a robust estimate of the performance of your approaches. If you select them too small, your good results are maybe just the result of chance.

Homework 9.1 *(10 points) In this homework, we use a deep neural network to classify movie reviews as positive or negative.*

(a) (1 Point) Define a function `data_reader(filename:str)` *to read the movie reviews in the given filename. Map the positive sentiments to 1.0 and the negative sentiments to 0. The function returns a tuple containing the reviews as a list of strings and the sentiments as a numpy array -> ([str], numpy.array).*

(b) (1 Point) Define a function `load_fast_text_embeddings(filename:str)` *to load the embeddings in a dictionary in the given filename. The keys of the dictionary are the words. The values are the corresponding embeddings. The function returns a tuple containing the dictionary and the dimension of the embeddings -> (dict[str, numpy.array], int).*

(c) (1 Point) Define a function `tokenize_sentences(sentences:[str]):` *to tokenize every sentence in sentences. This function should also remove stop words and punctuation. You are allowed to use tokenizers and the stopword list from the nltk library. The function returns a list of lists -> [[str]]. Hint: We consider every review to be a single sentence.*

(d) (1 Point) Define a function `map_to_vectors(tokenized_sentences:[[str]])` *which maps the given tokenized sentences to average embeddings. An average embedding is the average of the embeddings of the individual tokens in a sentence. This function returns a numpy array containing the embeddings of the corresponding sentences.*

(e) (4 Points) Check the function `MLP(input_shape:np.array)` *that defines our multi-layer perceptron (MLP).*

(i) How many layers does this model have (including the input and output layer)? (1 point)

(ii) What is the size of the matrix that connects the input layer and the first hidden layer? (1 point)

(iii) How many units/neurons are in the output layer? Why? (1 point)

(iv) What is the meaning of `Dropout`*? (1 point)*

(f) (2 Points) Finally, let's use the above implemented functions to train a model on a training set and evaluate the performance on a test set. ***Define an 80/20 training/test split in the code snippet below.*** *Run the code below to train and evaluate the model.*

```
reviews,sentiments=data_reader('dataset and embeddings/IMDB Dataset.csv')
# convert numeric labels to one hot vectors
onehot_encoder = OneHotEncoder(sparse=False)
```

```
sentiments=sentiments.reshape(len(sentiments),1)
sentiments = onehot_encoder.fit_transform(sentiments)


token2vector, dimensions = load_fast_text_embeddings(filename="dataset and embeddings/wiki-news-300d-10k.vec")


tokenized_reviews = tokenize_sentences(reviews)
#visual inspection if a reasonable tokenization technique was applied for question (c)
print('First Review')
print(tokenized_reviews[0])
print('Second Review')
print(tokenized_reviews[5])
print('Third Review')
print(tokenized_reviews[100])


# should be written by tutors
point_for_c =




embedded_reviews = map_to_vectors(tokenized_reviews)
#TODO
# split the vectorized reviews into train-, and testset
train_test_split = 0.8
train_x = None
train_y = None
test_x = None
test_y = None
#ENDTODO


assert(len(train_x)+len(test_x)==len(embedded_reviews))
assert(len(train_y)+len(test_y)==len(sentiments))


best_model = "model/best_model.ckpt"
# train the model and save the best one on dev set
cp_callback = callbacks.ModelCheckpoint(filepath=best_model, verbose=1, save_weights_only=True, monitor='val_acc', save_best_only=True)
model = MLP(input_shape=(dimensions,))
# use the binary cross entropy to measure the errors
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
model.fit(train_x, train_y, epochs=20, batch_size=20, callbacks=[cp_callback], validation_split=0.2)

model.load_weights(best_model)
# calculate the accuracy on test set
_,acc = model.evaluate(test_x,test_y)
print('accuracy on test set:', acc)

predictions=model.predict(test_x)
# convert predictions to one hot vectors
predictions_oneHot = np.where(predictions > 0.5, 1, 0)
print(predictions_oneHot)




# should be written by tutors
point_for_e_i =

```

```
66
67 # should be written by tutors
68 point_for_e_ii =
69
70
71 # should be written by tutors
72 point_for_e_iii =
73
74 # should be written by tutors
75 point_for_e_iv =
```

(g) *(0 Points) Play around with hyperparameters (e.g., size of hidden layers; number of hidden layers; dropout rate). What is the best accuracy that you get?*

Homework 9.2 *(0 points) The movie review data (Pang and Lee, 2004)*[1] *is also provided by NLTK. You can access it by using the following code:*

```
1 import nltk
2 from nltk.corpus import movie_reviews
3
4 # prepare review data as a list of tuples:
5 # (list of tokens, category)
6 # category is positive / negative
7 review_data = [(movie_reviews.words(fileid), category)
8     for category in movie_reviews.categories()
9     for fileid in movie_reviews.fileids(category)]
```

You can transform data and represent each instance as a frequency distribution of the top words. This might be helpful to define features. For example:

```
1 threshold = 1000 # Experiment with different thresholds
2
3 train_data = review_data[:int(0.8 * len(review_data))] # you may use the shuffle
    function of the random module to shuffle the data (see Handout 9)
4 fd_all_words = nltk.FreqDist(w.lower() for words, _ in train_data for w in words)
5 top_words = [word for (word, freq) in fd_all_words.most_common(threshold)]
6
7 review_data_fdist = [
8     (nltk.FreqDist(token.lower() for token in words if token in top_words),
9       category)
10    for words, category in review_data]
```

(a) *Train and tune a* `nltk.NaiveBayesClassifier` *for this task and define an appropriate feature set. If you use many features, training the classifier might take some time (more a problem of the NLTK classifiers than an argument against using many features). Thus, you might limit the number of movie reviews used for training during development. Make sure that you split the available movie reviews into training data, development data, and test data.*

Homework 9.3 *(0 points)*

(a) *Now try to use BERT or RoBERTa for the movie review classification problem. You may check* https://github.com/huggingface/transformers *(or other libraries) to do so. If you want to use BERT for the classification task, you should at first process your dataset in the way that Bert requires. After that you can use BertForSequenceClassification (a BERT model with a sequence classification head) by specifing the number of classes in your task.*

(b) *What is the difference between BERT and RoBERTa?*

(c) *Which approach gives you best performance: the MLP, the Naive Bayes Classifier, or BERT/RoBERTa? Be sure your comparison is fair! (What are criteria for fair comparison?)*

[1] See NLTK-book page Bo Pang and Lillian Lee. 2004. Asentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*.