L665 ML for NLPSpring 2018
Carlos Sathler (cssathler@gmail.com)

Assignment 2

Task 1 – POS Tagging with neural nets

The neural net described by Helmut Schmid in the paper referenced in the homework [1] requires considerable amount of pre-processing and it needs a lexicon. I noticed the paper was written more than two decades ago and I was curious to see if there were more recent attempts to do pos tagging with neural networks. As I researched the topic I found several recent sources describing pos tagging using neural nets. My favorite was "Bidirectional LSTM-CRF Models for Sequence Tagging" [2]. In this paper the authors document several experiments with long short-term memory networks (LSTM). For my homework task I chose to implement a bi-directional LSTM, which in the paper is reported as having second best performance in the authors' dataset of choice, the Penn TreeBank. (The best performing architecture combined regular LSTM with conditional random field (CRF)). Below are the details of my implementation:

Data:

I used the Brown corpus for this task 1. The data was partitioned as follows: 90% for training and 10% for testing. During training, I used 30% of the training partition for validation. I developed two Jupyter notebooks. The first one uses the "universal" tag set, which only has 13 classes. The second notebook uses the complete Brown corpus tag set, with 473 classes.

Feature and target extraction:

Tokens and tags were extracted with method brown.tagged_words and documents were extracted with the brown.sents method. I created two sequences: an input sequence of tokens for each sentence and a sequence of tags with one-to-one correspondence to the tokens in the first sequence. The tag sequence is the target. All sequences were expanded/truncated to a length = 45 tokens/tags, which is large enough to prevent truncating more than 95% of all sentences in the corpus.

Network architecture:

I created a network with 4 layers: Input, Embedding, Bidirectional LSTM, and Output MLP layer. The input layer feeds 45 long token sequences to the embedding layer which vectorizes each token to a dense vector of dimension 20. The vectorized tokens are the input to a bi-directional LSTM layer with 100 nodes. The output of the LSTM layer is passed to an MLP layer with softmax activation and number of nodes = number of class to predict.

A key feature of the architecture is the use of the TimeDistribution Keras layer wrapper [3]. This wrapper applies the MLP "dense" layer, repeated times, to the entire input sequence, so the output of the architecture is the entire sequence of tags that the network is trying to predict. Excellent explanations on how the TimeDistribution wrapper works and when to use it can be found here https://github.com/keras-team/keras/issues/957 and here https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/.

Hyper-parameter tuning:

Little time was spent in tuning, since parameters borrowed from the Keras documentation performed well from the beginning ("rmsprop" optimizer; "categorical_crossentropy" loss). I did try a few different batch sizes for training, and settled with 32. I used early stopping and 10 epochs was enough to train the model successfully. At one point I experimented with regularization using a dropout layer and dropout in the LSTM

layer, but with no significant improvement. The final model doesn't use regularization and still achieves good results in the test dataset.

Results:
Accuracy on the test set was 98.76% when predicting using a 13 tags tag set, and 97.64% when predicting using a 473 tags tag set. While these results are better than the ones reported by Huang et al. [2], the length of the sequence impacts the accuracy metric and so does the size of the tag set. Because Huang et al. do not provide details on the size of their tag set and the sequence length used in their model, I'm not able to determine if my results are indeed superior. Note also that one could attempt to remove padding before evaluating accuracy, however padding is also a target for prediction, so this procedure would introduce bias to the evaluation, just as much as does the selection of any sequence length value.

Code:
POS tagging with 13 tags tag set (tagset='universal')
https://github.com/csathler/IU_MSDS/blob/master/NLP/assignment2/assignment2_task1_1.ipynb

POS tagging with 473 tags tag set
https://github.com/csathler/IU_MSDS/blob/master/NLP/assignment2/assignment2_task1_2.ipynb

[1] Schmid, Helmut. "Part-of-speech tagging with neural networks." *Proceedings of the 15th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1994.

[2] Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." *arXiv preprint arXiv:1508.01991* (2015).

[3] Layer Wrappers - Keras Documentation, keras.io/layers/wrappers/.

L665 ML for NLPSpring 2018
Carlos Sathler (cssathler@gmail.com)

Task 2 – Dialog Act classification

My work for this task was based on the paper Dialog Act Recognition Approaches [1], the tutorial "Text Classification Using Neural Networks [2] and the paper Dialog Act Classification in Domain-Independent Conversations Using a Deep Recurrent Neural Network [3]. The first paper outlines several approaches for dialog act (DA) detection and classification, one of which is the use of neural networks. The paper mentions several MLP implementations for dialog act classification with accuracy results ranging from 62% to 93%, on a variety of datasets. The paper also describes several types of features relevant for DA, one of which is prosodic information. According to the authors, "most researchers agree on the fact that the lexical/syntactic information is not generally sufficient to explain DAs" [1, p233]. My homework exercise will attempt to verify if this assertion holds true on the Switchboard Dialog Act Corpus (SwDA) [4] using a bag of words BOW/MLP neural net approach similar to the one described in the tutorial above mentioned [2]. My results confirm that prosodic features do improve DA classification in the SwDA dataset, and show that a BOW/MLP architecture achieves accuracy comparable to that of more sophisticated neural net models using convolution and word vectorization.

Data:

I used the SwDA corpus put together by Christopher Potts [4] as suggested by the homework instructions. I used the Dialog Act Coder's Manual [5] to manually convert 303 DA classes to a more manageable 43 classes total, for my task. I set aside about 2% of the dataset for testing, which is the amount of data used by Khanpour et al. [3], so I could compare results. During training, 10% of the training dataset partition was used for validation.

Feature and target extraction:

I used sklearn's Tfidf vectorizer for feature extraction; this method generates a sparse matrix representation of a BOW. I created 3 notebooks, the first one of which uses the vectorizer defaults for analyzer (analyzer='word') and token pattern (token_pattern=(?u)\b\w\w+\b'), resulting in a BOW containing no special characters, and therefore no prosodic information. For the second and third notebooks I set analyzer='char' and token_pattern='.*' resulting in a BOW that includes prosodic information.

Network architecture:

For this homework task I implemented a straightforward MLP architecture with multiple layers and regularization via dropout and l1 and l2 regularizers on the output layer. I created 3 notebooks. The first and second notebooks have the exact number of layers and nodes; they are intended to allow comparison of accuracy results for BOW with vs. without prosodic information. The first notebook does not include prosodic features; the second one does. The third notebook uses BOW with prosodic information and contains two additional layers (1 MLP and 1 dropout) that yield much better accuracy results. The 3 notebooks are trained with early stopping and use the Keras SGD optimizer class. Values for learning rate, decay, momentum, and nesterov hyperparameters were borrowed from Keras documentation tutorials. I used the same batch size of 128 in all notebooks.

Hyper-parameter tuning:

I concluded after several runs that using the SGD optimizer class yielded much better results than the results possible with default values for the 'rmsprop' optimizer. Regularization was also very import to improve results. I tested different number of layers and drop out values before settling with 2 MLP layers for notebooks

L665 ML for NLPSpring 2018
Carlos Sathler (cssathler@gmail.com)

1 and 2. I tried several network depths to improve notebook 2 results on BOW with prosodic features. In the end I settled with 3 MLP layers to produce the final results that are captured in notebook 3.

Results:
Summary of results:

Notebook	Input	Architecture	Accuracy
1	BOW **without** prosodic information	2 MLP layers, 2 dropout layers	62.80%
2	BOW **with** prosodic information	Same as 1	68.44%
3	Same as 2	3 MLP layers, 3 dropout layers	72.34%

My best result with BOW/MLP are slightly inferior to the results reported by Khanpour at al. [4] for Convolutional Neural Nets (73.1%) and slightly superior to the results they report for Hidden Markov Chain (71%). Therefore, considering the scope of the homework, I'm comfortable with my results.

Opportunities for improvement:
The first opportunity for improvement would be to switch from a BOW input to a vectorized representation of the input tokens and prosodic information. That could be followed by a recurrent neural network implementation using LSMT layers, as described in [4].

It would also be interesting to include non-text features in the model. For example, can we improve DA classification results by including gender in the mix of input features? Are discussion topics relevant? With more time it would make sense to test not only different neural net types and architectures, but also non-textual features available in the transcript header files.

Code:
BOW without prosodic information
https://github.com/csathler/IU_MSDS/blob/master/NLP/assignment2/assignment2_task2_1.ipynb

BOW with prosodic information
https://github.com/csathler/IU_MSDS/blob/master/NLP/assignment2/assignment2_task2_2.ipynb

BOW with prosodic information and additional tuning (additional MLP and dropout layers)
https://github.com/csathler/IU_MSDS/blob/master/NLP/assignment2/assignment2_task2_3.ipynb

[1] Král, Pavel, and Christophe Cerisara. "Dialogue act recognition approaches." Computing and Informatics 29.2 (2012): 227-250.

[2] "Text Classification Using Neural Networks – Machine Learnings." Machine Learnings, Machine Learnings, 26 Jan. 2017, machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6.

[3] Khanpour, Hamed, Nishitha Guntakandla, and Rodney Nielsen. "Dialogue act classification in domain-independent conversations using a deep recurrent neural network." *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016.

[4] Potts, Christopher. "The Switchboard Dialog Act Corpus." Computational Pragmatics | The Switchboard Dialog Act Corpus, compprag.christopherpotts.net/swda.html.

[5] WS-97 Switchboard DAMSL Coders Manual, web.stanford.edu/~jurafsky/ws97/manual.august1.html.