ECS 7001 - NN & NLP
Assignment 1: Word Representation and Text Classification with Neural Networks

February 25, 2020

Text classification tasks are some of the most popular applications of NLP methods, particularly in an industrial setting, where for example sentiment analysis — recognizing the sentiment towards a product or individual — can be invaluable. In this assignment, you will eventually implement a text classifier capable of performing a real-world task. You will reach this goal by stages, each achieved during one lab session:

- In Part A of the assignment, to be carried out in Lab 3, week 3 (the current lab session) you will learn how to train the simpler form of word embeddings using `word2vec`.
- In Part B, to be carried out in Lab 4, week 4, you will implement a first simple classifier using simple word embeddings as inputs.
- In Part C (Lab 5, week 5) you will learn how to train more complex word embeddings using LSTMs;
- And finally, in Part D (Lab 6, Week 6), you will put everything together and implement a text classifier.

When all parts of the assignment are completed, you will have to submit two things:

- A PDF report describing what you did (instructions below);
- Your completed Python code.

The deadline for returning all completed parts of the assignment (Parts A, B, C and D) is **10:00:00 Monday 2nd March 2020**.

Part A: Word Embeddings with Word2Vec [20 marks]

For this part of the assignment, worth 20 marks in total, you will have to carry out the steps specified by the Lab script ("Lab 3: Skip-gram Model for Word2Vec" - separately provided). These marks are broken down among the several parts of the assignment as follows (please refer to the lab script):

1. Preprocessing the training corpus [3 marks].
   Your report must include in your report the output of the 'Sanity check' at the end of this Section

2. Creating the corpus vocabulary and preparing the dataset [3 marks].
   Again, you must include the output of the sanity checks at the end of the Section

3. Building the skip-gram neural network architecture [6 marks].
   To get the credit, you must include the output of the `model.summary()` command in the sanity check part

4. Training the models (and reading McCormick's tutorial) [3 marks].
   One point for each answer to one of the three questions

5. Getting the word embeddings [2 marks].
   To get the credit, you must include in your report the output of the instruction printing the DataFrame (summarized)

6. Exploring and visualizing your word embeddings using t-SNE [3 marks].
   Include in the report the output of the `plt.annotate` command

Part B: Basic Text Classification [25 marks]

For this part of the assignment, worth 25 marks in total, you must carry out the steps specified by the Lab script ("Lab 4: A Neural Network Classifier Using Simple Word Embeddings" - separately provided). These marks are broken down among the several parts of the assignment as follows (please refer to the lab script):

1. Build a neural network classifier using one-hot word vectors, and train and evaluate it [5 marks].
   In your report, include the output of the `model.summary()` command to show your model structure, and plot the training and validation loss in a graph.

2. Modify your model to use a word embedding layer instead of one-hot vectors, and to learn the values of these word embedding vectors along with the model [5 marks].
   Again, include the output of the `model.summary()` command to show your model structure, and plot the training and validation loss in a graph.

3. Adapt your model to load and use pre-trained word embeddings instead (either the embeddings you built in part A or from another pre-trained model), and train and evaluate it [7 marks].
   Again, include the output of the `model.summary()` command to show your model structure, and plot the training and validation loss in a graph.

4. One way to improve the performance is to add another fully-connected layer to your network. Try this, and explain why the performance does not improve. What can you do to improve the situation? [8 marks]
   Plot the training and validation loss of the new model, and other models you try. In your report, describe the differences you see and discuss why they occur.

Part C: Using LSTMs for Text Classification [25 marks]

For this part of the assignment, worth 25 marks in total, you will have to carry out the steps specified by the Lab script ("Lab 5: - LSTMs for Text Classification" - separately provided). These marks are broken down among the several parts of the assignment as follows (please refer to the lab script):

1. Section 2, Readying the inputs for the LSTM [2 marks].
   For this part, show the output you obtain from the sanity check.

2. Building the model (section 3 of the script): [6 marks].
   For this part, show the structure of the model you obtain.

3. Section 4, training the model [6 marks].
   For this part, show the plot of training and validation accuracy through the epochs and comment on the optimal stopping point for the model.

4. Evaluating the model on the test data (section 5) [2 marks].
   For this part, show the output of the command printint test_loss and test_accuracy.

5. Section 6, extracting the word embeddings [2 marks].
   For this part, show the output of `model.summary()`

6. Visualizing the reviews [1 mark].
   For this part, you should include in the report the output of the command printing out the idx2word map.

7. Visualizing the word embeddings [2 marks].
   For this part, you should include the word embeddings for 10 of the words.

8. Section 9 [4 marks].
   For this paper, you have to write down your answers to the questions. 2 points each for questions 1 and 2.

Part D: A Real Text Classification Task [30 marks]

Task For the final part of the assignment, your task is to implement your own entry for SemEval. SemEval is an annual international series of workshops for the evaluation of NLP systems, based around public shared tasks - see the Wikipedia page at `https://en.wikipedia.org/wiki/SemEval`. Teams from research groups around the world enter, train their systems on given training sets and test on unseen test sets, and the organizers evaluate them and publish the results as a league table.

For this assignment, we'd like you to enter **SemEval 2019 Task 6, OffensEval: Identifying and Categorizing Offensive Language in Social Media**. You can read more about the task in this paper:
`http://www.aclweb.org/anthology/S19-2010/`
and see more information about the task and dataset here:
`http://sites.google.com/site/offensevalsharedtask/offenseval2019`

You will notice two things that make this more challenging than the previous parts of this coursework. First, although Subtasks A and B are document-level binary classification tasks (just as in Parts B and C above), Subtask C is a *multi-class* task. Second, the subtasks are inter-dependent: classification decisions from Subtask A affect the decisions to be made in B and C. Specifically we would like you to:

1. Build and evaluate a basic classifier for Subtask A, adapting one of your models from Parts B and C above [10 marks].

2. Build and evaluate a basic classifier for Subtasks A-C combined, treating this as one single multi-class problem. There are 5 possible classes: NOT, OFF-UNT, OFF-TIN-IND, OFF-TIN-GRP, OFF-TIN-OTH (see the README with the dataset) [10 marks].

3. Try to improve your overall accuracy on Subtasks A-C. You could try one or more of: using a CNN instead; using a different classifier; treat the three tasks separately in a pipeline; modify embeddings, loss function etc - see ideas in Week 5 & 6 lectures. [10 marks].

The training and test sets are available from the OffensEval web page on the link above.

You can use any of the classifiers and embeddings that you've built so far in Parts A-C of this assignment, and adapt them as you see fit (or you can start from scratch and approach it in a different way, of course). Feel free to download off-the-shelf embeddings or other tools too (although be aware that just downloading BERT and applying it off-the-shelf won't get you many marks).

For each subtask, include in your report your evaluation results; for part 3, include an explanation and justification of your choice of classifier architecture, training setup and parameters.

Some Questions you might want to ask yourself as you proceed:

- What's the best network configuration? DAN, CNN, RNN, LSTM, bi-directional LSTM ...? What are the best embeddings to use?
- What activation functions do I need – should they be different for the different subtasks?
- How should I best account for the fact that the classes in subtasks A-C are dependent (subtask B's labels only apply to one class from subtask A, and so on)?
- How far can I get with this training data alone?
- Can I improve things by using separate unlabelled data (e.g. for training embeddings), or even labelled data (e.g. using distant supervision)?
- Does re-training embeddings for this task, or using different pre-trained embeddings, help?

Submission

Please submit one zip file with all your answers to Parts A-D together.

As well as code, you should include text explanations, descriptions and answers to specific questions as necessary and as specified above. Code should be in Python; explanatory text can be either as a separate report in PDF format (not Word, please), or included together with the code as a Jupyter/Colab notebook.

For each section, marks will be awarded for correctness of code and classifier performance, but also for clarity of explanations and justifications.