MSAI-437: Deep Learning
Group Project #1 - Fakes Detection[1]

(50.0 points + 5.0 bonus points)

Winter 2023

Description: As language models become more powerful and more adept at generating human-like text, there is ever-increasing concern about their ability to generate fake content at scale. Earlier work (see arxiv.org/abs/1905.12616) shows that the more powerful models used to generate fake content can be repurposed to detect fakes. In this assignment, you will work with different language model architectures to build a "Fakes Detector". Other than using the specified architectures, most of the design space is up to you (e.g., task framing, use of data provided, model sizes, hyper-parameters, etc.). You can experiment with your models on the labeled datasets. Once you are confident that your models are working correctly, you will apply your best model to a "blind" dataset to generate `[REAL]` or `[FAKE]` labels.

The domain for this project is scientist biographies. I obtained approximately 20,000 real biographies for scientists and used these to fine-tune a GPT-2 model. I generated a comparable number of fake biographies from this trained model. I sub-selected from each of these to construct dataset sets balance in length and class label. Real and fake biographies are stored in separate files (without class labels). Special tokens `<start_bio>` and `<end_bio>` delimit each biography. In addition, I created a mixed dataset consisting of a 50/50 mix of real and fake biographies. Observations in the mixed dataset include a class label after the `<end_bio>` token. Lastly, I created a blind dataset similar to the mixed dataset with class labels omitted.

Description	**File Names**	**Approx. Obs.**
Real biographies from Wikipedia	`real.{train/valid/test}.txt`	8000/1000/1000
Fake biographies from GPT-2	`fake.{train/valid/test}.txt`	8000/1000/1000
Mixed real and fake biographies	`mix.{train/valid/test}.txt`	8000/1000/1000
Blind text set without labels	`blind.test.txt`	500

Files ending with the `.txt` extension are in raw text format. Each corpus is also available in lowercase and tokenized with the NLTK (see nltk.org) in files ending with a `.tok` extension. You can use any portion of these datasets as you see fit.

Tasks:

1. **FFNN** (5 pts): Implement a feed-forward neural network language model (similar to the Bengio model covered in class) and use it to perform Fakes Detection. There are a large number of examples available on the Web. You are permitted to consult these, but please implement your own code. The design of the network, hyper-parameters and how you use the data are up to you. A common paradigm is to train the neural network on labeled examples and then use this trained network to perform binary classification. Hint: A language model performs multi-class classification over a semi-labeled corpus -- Fakes Detection is a binary classification task. The probability over a sequence may be helpful.

   Please describe how you organized and used the datasets provided, the operation of your FFNN and hyper-parameter settings. Present the learning curves for training and test perplexity, and report Fakes Detection results in a confusion matrix with labeled axes. Please discuss any limitations of your approach and possible solutions. Please do not exceed one page for this.

[1] In this assignment "Fakes" refers to computer generated content.

2. **LSTM** (15 pts): Implement a recurrent neural network using LSTM cells and use it to perform Fakes Detection. Do not re-implement the LSTM cell -- please use the module available in PyTorch. Again, there are many examples available on the Web. You are permitted to consult these, but please implement your own code. The design of the network, hyper-parameters and how you use the data are up to you. A common paradigm is to train the neural network on labeled examples and then use this trained network to perform binary classification. Hint: A language model performs multi-class classification over a semi-labeled corpus -- Fakes Detection is a binary classification task.

   Please describe how you organized and used the datasets provided, the operation of your LSTM LM and hyper-parameter settings. Present the learning curves for training and test perplexity, and report Fakes Detection results in a confusion matrix with labeled axes. Please discuss any limitations of your approach and possible solutions. Please do not exceed one page for this.

   For reference, I attached my learning curves and confusion matrix.

3. **Transformers** (20 pts): Select a transformer-based model from Huggingface and fine-tune it to perform Fakes Detection as binary classification. The architecture could be encoder-only (BERT-based), decoder-only (GPT-based) or encoder-decoder (see T5). Many examples are available, including https://huggingface.co/docs/transformers/tasks/sequence_classification. Select a model size appropriate for your resources.

   Bonus (5 pts): Alternatively, you can implement and train a transformer-based language model to perform Fakes Detection like the FFNN and LSTM approaches above. Refrain from attempting to implement the Transformer from scratch. Several tractable codebases are available including, https://blog.floydhub.com/the-transformer-in-pytorch.

4. **Results** (10 pts): Use your best model to perform Fakes Detection on the Blind dataset, which consists of 500 unlabeled observations. You may also construct your own non-neural model. A simple rules-based approach may be effective -- working with incongruent dates is one approach. Please provide a `.csv` file containing your `[REAL]` or `[FAKE]` labels for each observation. I plan to share summary results for each group with the class. Discuss how you selected your model, any limitations and possible solutions. Please do not exceed ½ of a page for this.

You must save your final model parameters and all associated variables for grading purposes. Implementing your FFNN and LSTM as Python classes may facilitate this process. My implementations are approximately 20 lines of code each, excluding initializations. The bulk of your code will likely perform data handling and work with language model outputs. Minimal starter code is provided to help with the reading and tokenization of the data files, but you are not required to use this.

What to Submit: Please submit a single `.zip` file for the group. The zip file should contain the following:

- A single `.pdf` for all written responses, tables and graphs,
- A single `.py` file will all code for the FFNN and LSTM models (note: please do not submit Jupyter Notebooks),
- All files needed to recall the parameters and associated variables of your final FFNN and LSTM models,
- All code and checkpoints need to replicate transformer-based results, and
- A single script that will reproduce Fakes Detection results.

Only one group member needs to submit.

1. Sample FFNN Results:

		Actual	
		Real	Fake
Predicted	Real	127	117
	Fake	59	197

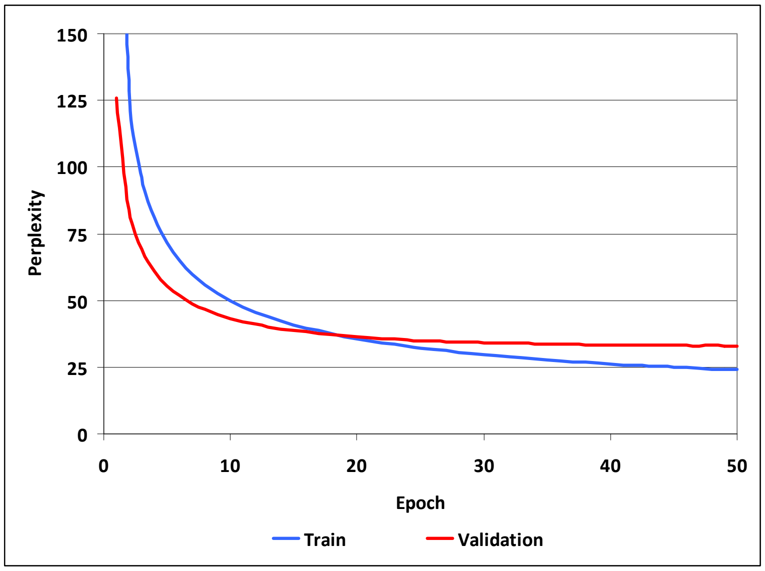
2. Sample LSTM Results:

Hyper-parameters:

Namespace(batch_size=20, clip=2.0, d_hidden=512, d_model=512, dropout=0.35, epochs=50, loadname=None, lr=0.0001, model='LSTM', n_layers=2, printevery=5000, savename='lstm_mix512', seq_len=30, testname='datasets/mix.test.tok', trainname='datasets/mix.train.tok', validname='datasets/mix.valid.txt', vocab_size=35150)

Training Time: ~1.0 hours

Learning Curves:



Confusion Matrix:

		Actual	
		Real	Fake
Predicted	Real	128	71
	Fake	116	185