Deep Learning Challenge

Intro:

At Proteinea, we use deep learning to power our drug design pipeline, helping generate and evaluate new proteins that could later be used as vaccines, therapeutics, or food additives. We are growing fast and so do our products. If you are looking for a place to utilize and build knowledge in the field, and to add real value besides exposure to state of the art research, advisors, and challenges, we cannot wait to see what you have got !

1. Deep Learning Coding Problem : Conditional Generation of Dates (80 Points)

Motivation:

Deep Generative models are a well known tool in deep learning to produce synthetic high quality data similar to data seen during train time –i.e. fake data that looks real. You may have come across famous ***unconditional*** generation examples, like this-person-does-not-exist.com which shows high quality **randomly** generated photos of human faces using a type of generative models called GANs, or possibly the more involved diffusion models. If we could utilize the same kind of models to do a more deliberate (designed, **conditioned**) generation of faces, we can do amazingly fun things like make someone look younger or older – possibilities are limitless.

At Proteinea, conditional generation is central to some of our protein design work. We use generative models to create proteins that did not exist before (synthetic proteins) and condition them to satisfy certain properties (make them stable under higher temperatures, for example), while maintaining their original characteristics. This is akin to the way faceapp makes you look younger with the same facial properties.

Problem Description:

For this problem, you are asked to generate a date given a set of conditions, using any neural network architecture you would like. Your input (x) is the conditions on the date, and the output (y) is ANY date that complies with those conditions. This means that, like any generative model, there are many right answers per input x.

Dataset

You are given a **data.txt** file containing the entire dataset, one entry per line. Each line is in the format:

```
[day condition] [month condition] [leap year condition] [decade condition] date
```

Some examples:

```
[MON] [DEC] [False] [196] 3-12-1962
[THU] [DEC] [True] [204] 3-12-2048
[WED] [JAN] [False] [181] 10-1-1810
```

- **day condition: input.** A three letter token, with square brackets around it, depicting the day that the generated date should match. Eg: `[WED]` means that the output date should occur on a Wednesday. For this condition to be **PASSED**, the date has to match a wednesday in any month or year.
- **month condition: input.** A three letter token, with square brackets around it, depicting the month that the output date should occur in. Eg: `[JAN]` means that the output date should be in January. For this condition to be **PASSED**, the date has to occur in January, of any year and on any day.
- **leap year condition: input.** Either a `[False]` or a `[True]` token, depicting whether the output year should be a leap year (True) or not (False). For this condition to be **PASSED**, the date has to occur in a leap year, regardless of the decade, month, or day. Leap year definition can be found here.
- **decade condition: input.** A three letter token, with square brackets around it, depicting the decade that the output date should occur in. Eg: `[192]` means that the output date can be any date from `1-1-1920` to `31-12-1929`. For this condition to be **PASSED**, the date has to occur in the given decade, regardless of the day, month, or being in a leap year.
- **date: output.** The only output. A date string of the format `dd-mm-yyyy` (day, month, year). It should match all the previous conditions.

Provided Files:

1. `repo/model/eval.py :` functions to evaluate your output.
2. `repo/data/datagen.py :` python file to generate data
3. `repo/data/data.txt :` all data generated using datagen.py (running datagen.py will just regenerate it)
4. `repo/data/example_input.txt :` example file with input conditions only, without output.

Hints:

1. The hardest part about this problem will probably be figuring out the **problem formulation** and corresponding network **architecture**. It is an important step for any

deep learning project. Take your time reading the problem description and reading about possible models online.

2. It is better to implement a **custom tokenizer**, and maybe effective to change the order of the tokens for some architectures. Think about this: What are easier tokens to figure out given the input conditions ? digit by digit, in the date.
3. **Data imbalance:** For some conditions, there is much less data than others. You can handle this using data imbalance handling techniques. It is not essential, but will improve your accuracy. Do not start with it, but maybe finish with it.
4. This is a generation, not a classification problem. Accuracy is not the best way to monitor your model output since several answers are correct. What is a better way you can use to monitor your model while training ?

Submission:

Submission is done as a reply to the mail we sent you.

Requirements:

1. A private cloned repository containing your solution, with inviting hazemessamm and wmustafaawad as readers to your private repo.
2. A pdf or .docx named "`dl_problem_your_name`" file containing briefing about your methodology, your analysis of the outputs, training and test loss graphs, and any figures you see relevant.

How to:

1. Clone the repo and start implementing the code in your own "**Private**" repo. Please do not make your solutions public till at least one month from now. Share your repo with the following github accounts: hazemessamm and wmustafaawad.
2. Your repository is expected to have:
    a. The model's training, inference, and evaluation code in a folder called "model". It should also contain the model's training weights.
    b. The inference code should be located in a file of path: `repo/model/predict.py`, and can be run using the following command: `python predict.py -i $path_to_input_file -o $path_to_output_file` , producing a file with the predictions in `$path_to_output_file`. An example input file is present under the path `repo/data/example_input.txt`. The output format should match the `data.txt` format exactly (conditions + output), in the same order as the `example_input.txt`.
3. conda environment spec file to allow replication of your results if needed
4. "`dl_problem_your_name`" report file is an important part of the grading, make sure to reflect on your choices and analysis briefly. The document should not be more than 5 pages by any means, one page could be good enough. Think of it as a document to communicate your implementation (without the code), and why it works, to a fellow deep learning engineer.

Constraints:

1. Your submission should be in python 3.6+.
2. If you are going to use TensorFlow 2.x, you should implement the training loop from scratch (do not use model.fit())
   Hint: you can use tf.GradientTape() to record the operations and get the gradients (Check this link).
3. Your code is required to only work for dates in the range [`1-1-1800` to `31-12-2200`]. Do not overwhelm yourself by covering all possible dates in history.
4. Provide a reference to any code snippets you copied during coding, even and specially from stack overflow, using links in comments (`# link to code`).

Evaluation Criteria:

1. **Evaluation Metric** (similar to eval.py but not identical to it).
2. **Report: Briefness & Clarity –** Can we make the same conclusions you made, fast ?
3. **Problem formulation** (make sure you explain it well in the report file, including tokenization, loss function, architecture ..etc)
4. **Code readability:** This is necessary for us to be able to evaluate your code. Do not submit all your code in one notebook. You are a software engineer, structure your code in folders, files, and classes when needed.
5. **Code Correctness:** Your logic should be sane, assumptions you make should be explained in the report.
6. **Results readability:** Can we visually and logically validate that your results are ok ? Provide some output examples, maybe also provide examples where your model failed and your reflection on them.
7. **Percentage of original code**: While not essential, it is important to be able to see your coding quality and skills.
8. **Bonus:** Deep Learning and coding best practices: shuffling data, test data, manual seed to allow for experiment replication, type hinting ..etc.

2. Technical Essay Questions (20 Points)

1. A deep learning developer is working for NETFLIX on predicting whether the user will interact with a recommended movie title or not. She trained the model, achieving 90% accuracy on the test data. The model was deployed in production to suggest movie titles for users, and the accuracy was found to be quite close to 90% (88%). This is good news, she marks the project as done, and moves to a new project.
   After one year, the backend engineers report a problem: the model is producing much lower accuracy than it used to – it's now accurate 60% of the time only. She is doubtful that the deployed code might be corrupted, but she runs the deployed code, and to her surprise, it produces 90% accuracy on the test set she worked with one year ago. She also verified that it is the exact same code she worked with one year ago.

She comes to you asking about why this happened, and what she could do about it. What is your best guess as to what happened ? What would you ask her to check to verify if your guess is true ? How can she fix this problem if your guess was right ?

2. A deep learning developer was working on a GAN model for MNIST digit generation. The goal was to have a stable decrease in the generator loss, and generate diverse photos of digits similar to those in the original data. The generator loss was steadily decreasing as desired, so he thought that his model was doing great, and kept training it till the loss was flat for a good number of iterations and showed no more decrease. He then stopped training and started generating images.
   To his surprise, he found out that the model was generating the number 8 only, regardless of the input. Whatever values he changed the input or its standard deviation to, the model produced only very slight variations of the number 8, ignoring all other digits (0 to 9) that were found in the dataset. He comes to you asking about a solution to the situation. What's your best guess about what has happened ? How did the generator loss decrease and was still generating the same outputs at the same time ? What would you edit about his training process or code to avoid what happened ?

Submission:

Submission is done as a reply to the mail we sent you.
Submit a .pdf or .docx file, one page, of your answers, in the same repo as the coding question. Name it: `essay_answers_your_name`

Inquiries

- For technical inquiries about the task, please contact: hazem@proteinea.com