046747 Bonux EX6

Guidelines

1. Submit your files using Moodle system.
2. You are allowed to submit in **pairs**. if you choose to do so, **both students have to submit**.
3. In order to submit your solution please submit the following files:
   (a) `ex_3.py` - Python 3.7+ code file(attach all your src code).
   (b) `output.txt` - A txt file with your predictions.
   (c) `ex_3.pdf` - A pdf file, described in section 3.
   (d) `kenlm.arpa` - the generated language model. described in section1.2.

   Follow the instructions and submit all files needed for you code to run.

Good Luck!

1 Automatic Speech Recognition

In this exercise, you will implement your first ASR system!
The dataset you are given is called `TIDIGITS` where a sequence of **up to 7** digits is pronounced in each recording. For simplicity, this corpus's vocabulary is only composed of digits(0-9).

In class, you have learned about the key components of an ASR system - An acoustic model and a Language model.

1.1 Acoustic Model

For training an acoustic model without an aligned transcription, you should use the `CTC` loss over the **letters** composing the pronunciation of each digit.
In other words :

```
The digit codes in a filename indicate the digit sequence spoken and can be
decoded as follows:
z -> zero        3 -> three        7 -> seven
o -> oh          4 -> four         8 -> eight
1 -> one         5 -> five         9 -> nine
2 -> two         6 -> six

e.g. a file named '7o19a.wav' is transcribed into -> SEVEN OH ONE NINE.
                  '249z9a.wav' is transcribed into -> TWO FOUR NINE ZERO NINE.
                  '4b.wav' is transcribed into -> FOUR.
Note: the last letter in each filename is a|b and can be ignored.
```

The training set contains 12549 recordings produced by men, women, boys, and girls.
Implementation details are up to you - implement as you wish!

1.2 Language Model

The decoding process combines the acoustic model probabilities with a language model to get the most probable transcription.

As seen in class, there are many types of decoders. In this assignment, we will focus on `Greedy_decoder` and a `CTC_decoder`. Both decoders are presented in this tutorial(click). Note - some features require updating packages. In order to evaluate the decoding components, you will have to decode ~~four~~ **three** times for the following configurations.

- Greedy_decoder without a language model.
- ~~Greedy_decoder with a language model.~~
- CTC_decoder without a language model.
- CTC_decoder with a language model.

Note: You can use the decoders in the tutorial mentioned above.
We recommend building an n-gram LM with KenLM. Attach the generated LM file with the `.arpa` extension to your submission. (optional) You are provided with `lexicon.txt` and `train_transcription.txt` for your convenient.

For each configuration, report the Word Error Rate(WER) and Character Error Rate (CER) for different beam sizes[1, 50,500] (**beam size is only relevant for the ctc_decoder**).
Overall Given your trained acoustic model, you should include the ~~24~~ **14** reported values in your report - in a 2-column table (cols are WER and CER). `n_rows` is ~~12~~ **7** - 2 ctc_decoding conf X 3 beam_size options. + 1 greedy decoder.
PyTorch metrics implementations(optional): WER and CER

2 Testing

Once your system is trained, pick the best configuration for you from section 1.2 and generate predictions for the given test set. You should write them to a file named `output.txt` (should also be submitted) in the following format:

```
test_0.wav - 526883z
test_1.wav - 1
test_2.wav - 4o629
test_3.wav - 31
..
test_12546.wav - 28

format : <test_wav_name> - <prediction>
Your output.txt content can be in any order.
```

3 Report

You should submit a file called `ex_3.pdf` which includes the following:

- WER and CER for the configurations described in 1.2.
- A description of your acoustic model.
- instructions for running your code.

Good Luck!