Lab 1 – Performing Text Normalization

This page describes performing the lab and the outline of the pages below. I apologize in advance if the lab was not performed as the instruction intended, I may have misinterpreted or misunderstood the instruction. This was my first-time using Google Colab and I already enjoy it a lot.

In Google Colab, I added text between certain codes and describing what I was doing. After I was done in Google Colab I exported the .ipynb file as a pdf file with code and text, which is why the following pages are in landscape mode.

My though process for this lab was as follows. Download the dataset from Kaggle and use the English folder only. Since the there were lots of files in the English folder, I decide to choose the first file (output-00000-of-00100) and use it to normalize it. I attached a separate text file (TextNormalized_OutputFile.txt) containing the normalized results. But for the demonstration of the lab, I used the top five examples of the test file, and performed text normalization and inverse text normalization as shown in the tutorial file.

Please let me know if you need the .ipynb file or anything else. Thank you.

To perform the following lab, I took Kaggle's API and downloaded the datasets to google colab, and extracted it. Below are the following commands I performed to get the dataset and extract it to a new folder.

Side Note: This is my first time using google colab and it has been a while since I used Python. Also, I apologize in advance if I misunderstood the lab instruction and performed it incorrectly.

```
! pip install -q kaggle
```

```
from google.colab import files

files.upload()
```

```
! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/
```

```
! chmod 600 ~/.kaggle/kaggle.json
```

```
! kaggle datasets download richardwilliamsproat/text-normalization-for-english-russian-and-polish
```

```
! mkdir Text_Normalization_For_Languages
```

```
! unzip /content/text-normalization-for-english-russian-and-polish.zip -d /content/Text_Normalization_For_Languages
```

```
! tar zxvf /content/Text_Normalization_For_Languages/en_with_types.tgz -C /content/Text_Normalization_For_Languages
```

Here I performed the necessary installations to use NeMo and imported it to python.

```
## Install NeMo, which installs both nemo and nemo_text_processing package
BRANCH = 'r1.14.0'
!python -m pip install git+https://github.com/NVIDIA/NeMo.git@$BRANCH#egg=nemo_toolkit[nlp]

# install Pynini for text normalization
! wget https://raw.githubusercontent.com/NVIDIA/NeMo/main/nemo_text_processing/install_pynini.sh
! bash install_pynini.sh
```

```
# try to import of nemo_text_processing an other dependencies
import nemo_text_processing
import os
```

In the next few lines of code, I essentially performed the following bulletin steps together:

- 2.1 Run TN on input string.
- 2.2 Run TN on list of input string.
- 2.3 Evaluate text pairs.

First I connected the path to the en_with_types test folder and choose output-00000-of-00100 as the file to perform the text normalization on. This file has a lot of examples in this format:

> " DATE 2006 two thousand six
> LETTERS IUCN i u c n
> PLAIN Red <self>
> PLAIN List <self>
> PLAIN of <self>
> PLAIN Threatened <self>
> PLAIN Species <self>
> PUNCT . sil
> <eos> <eos> "

Therefore, I performed the following in the code below to get it into a un normalized sentence form, before proceeding to normalizing it.

```
from nemo_text_processing.text_normalization.data_loader_utils import load_files, training_data_to_sentences
#Set the file path from google collabs files section.
Input_File = f'/content/Text_Normalization_For_Languages/en_with_types/output-00000-of-00100'

#Here we have all un normalized and text normalized data of the input file above. However it is very large to show on here.
eval_data = load_files([Input_File])
sentences_un_normalized, sentences_normalized, sentences_class_types = training_data_to_sentences(eval_data)
unNormalized = list(sentences_un_normalized) #Get the un normalized sentences only and place them in a list
```

```
# Create text normalization instance.
from nemo_text_processing.text_normalization.normalize import Normalizer
normalizer = Normalizer(input_case='cased', lang='en')
```

```
[NeMo I 2023-01-27 23:43:01 tokenize_and_classify:87] Creating ClassifyFst grammars.
```

Next, I performed normalization on the first five un normalized example sentences of the input file above.

```
#Here we run normalization on first 5 examples (elements) strings from the un normalized list


for i in unNormalized[:5]:
  written = i
  print("This sentence is un normalized: ", written)
  normalized = normalizer.normalize(written, verbose=True, punct_post_process=True)
  print("This sentence is normalized: ", normalized)
  print()
```

```
This sentence is un normalized:  brillantaisia is a genus of plant in family acanthaceae
tokens { name: "brillantaisia" } tokens { name: "is" } tokens { name: "a" } tokens { name: "genus" } tokens { name: "o
This sentence is normalized:  brillantaisia is a genus of plant in family acanthaceae
```

```
This sentence is un normalized:  2006 iucn red list of threatened species
tokens { date { year: "two thousand six" } } tokens { name: "iucn" } tokens { name: "red" } tokens { name: "list" } to
This sentence is normalized:  two thousand six iucn red list of threatened species

This sentence is un normalized:  circa survive draw influences from soft rock post hardcore experimental rock emo prog
tokens { name: "circa" } tokens { name: "survive" } tokens { name: "draw" } tokens { name: "influences" } tokens { nam
This sentence is normalized:  circa survive draw influences from soft rock post hardcore experimental rock emo progres

This sentence is un normalized:  circa survive opened for my chemical romance's 2007 shows in worcester and australian
tokens { name: "circa" } tokens { name: "survive" } tokens { name: "opened" } tokens { name: "for" } tokens { name: "m
This sentence is normalized:  circa survive opened for my chemical romance's two thousand seven shows in worcester and

This sentence is un normalized:  it has a greyish green densely hairy stem and alternate palmately lobed leaves
tokens { name: "it" } tokens { name: "has" } tokens { name: "a" } tokens { name: "greyish" } tokens { name: "green" }
This sentence is normalized:  it has a greyish green densely hairy stem and alternate palmately lobed leaves
```

Next, I will perform some evaluation commands to see the runs prediction and measure sentence accuracy of the un normalized sentences and normalized sentences of the five examples above.

```
# Run prediction
sentences_prediction = normalizer.normalize_list(unNormalized[:5])
print(sentences_prediction)
```

```
100%|██████████| 1/1 [00:00<00:00, 41.69it/s]

100%|██████████| 1/1 [00:00<00:00, 44.06it/s]

100%|██████████| 1/1 [00:00<00:00, 20.65it/s]

100%|██████████| 1/1 [00:00<00:00, 24.04it/s]

100%|██████████| 1/1 [00:00<00:00, 44.98it/s]
['brillantaisia is a genus of plant in family acanthaceae', 'two thousand six iucn red list of threatened species', 'c
```

```
# Measure sentence accuracy
from nemo_text_processing.text_normalization.data_loader_utils import evaluate

sentences_accuracy = evaluate(
            preds=sentences_prediction, labels=sentences_normalized[:5], input=unNormalized[:5]
        )
print("- Accuracy: " + str(sentences_accuracy))
```

```
     inpu: "2006 iucn red list of threatened species"
     gold: "two thousand six i u c n red list of threatened species"
     pred: "two thousand six iucn red list of threatened species"
     - Accuracy: 0.8
```

As we can see above, the second example of our un normalized list has 0.8 our of 1.0 accuracy, because the difference between "i u c n" and "iucn". The correct one is the sentence with "i u c n". Moreover, the other four examples were correct and got an accuracy of 1.0.

Next, I will perform the Inverse Text Normalization of the five examples above.

```
# Create inverse text normalization instance
from nemo_text_processing.inverse_text_normalization.inverse_normalize import InverseNormalizer
inverse_normalizer = InverseNormalizer(lang='en')
```

```
     [NeMo I 2023-01-28 01:18:01 tokenize_and_classify:64] Creating ClassifyFst grammars.
```

```
# Run Inverse TN on the five examples.
for i in sentences_normalized[:5]:
  spoken = i
  un_normalized = inverse_normalizer.inverse_normalize(spoken, verbose=True)
  print(un_normalized)
  print()
```

```
tokens { name: "brillantaisia" } tokens { name: "is" } tokens { name: "a" } tokens { name: "genus" } tokens { name: "o
brillantaisia is a genus of plant in family acanthaceae

tokens { cardinal { integer: "2000" } } tokens { telephone { number_part: "6iucn" } } tokens { name: "red" } tokens {
2000 6iucn red list of threatened species

tokens { name: "circa" } tokens { name: "survive" } tokens { name: "draw" } tokens { name: "influences" } tokens { nam
circa survive draw influences from soft rock post hardcore experimental rock emo progressive rock art rock and pop mus

tokens { name: "circa" } tokens { name: "survive" } tokens { name: "opened" } tokens { name: "for" } tokens { name: "m
circa survive opened for my chemical romance's 2007 shows in worcester and australian and new zealand shows as well

tokens { name: "it" } tokens { name: "has" } tokens { name: "a" } tokens { name: "greyish" } tokens { name: "green" }
it has a greyish green densely hairy stem and alternate palmately lobed leaves
```

Every thing looks good above, except the 2006 has been missread; hence, it returned an incorrect un normalized sentence.

Lastly, I will preform the Audio-based Text Normalization.

In the following I will try and generate five normalization options of each example.

```
# import non-deterministic WFST-based TN module
from nemo_text_processing.text_normalization.normalize_with_audio import NormalizerWithAudio
```

```
# Initialize normalizer, this may take some time to generate the extended grammars.
# Thus, we recommend to cache the grammars by specifying a cache directory
normalizer = NormalizerWithAudio(
        lang="en",
        input_case="cased",
        overwrite_cache=False,
        cache_dir="cache_dir",
    )
```

```
Created cache_dir/en_tn_post_processing.far
[NeMo I 2023-01-28 01:31:35 tokenize_and_classify_with_audio:101] Creating ClassifyFst grammars. This might take some
```

```
Created cache_dir/_cased_en_tn_False_deterministic.far
[NeMo I 2023-01-28 01:39:52 tokenize_and_classify_with_audio:229] ClassifyFst grammars are saved to cache_dir/_cased_e
Created cache_dir/en_tn_False_deterministic_verbalizer.far
[NeMo I 2023-01-28 01:39:53 verbalize_final:76] VerbalizeFinalFst grammars are saved to cache_dir/en_tn_False_determin
```

```
# Create up to 5 normalization options for the five un normalized examples above.
for i in unNormalized[:5]:
  print(normalizer.normalize(i, n_tagged=5, punct_post_process=True))
  print()
```

```
brillantaisia is a genus of plant in family acanthaceae
{'brillantaisia is a genus of plant in family acanthaceae'}


2006 iucn red list of threatened species
{'two oh oh six iucn red list of threatened species', 'two thousand and six iucn red list of threatened species', 'twe


circa survive draw influences from soft rock post hardcore experimental rock emo progressive rock art rock and pop mus
{'circa survive draw influences from soft rock post hardcore experimental rock emo progressive rock art rock and pop m


circa survive opened for my chemical romance's 2007 shows in worcester and australian and new zealand shows as well
{"circa survive opened for my chemical romance's two thousand seven shows in worcester and australian and new zealand


it has a greyish green densely hairy stem and alternate palmately lobed leaves
{'it has a greyish green densely hairy stem and alternate palmately lobed leaves'}
```

After performing the task above, we note that there were more normalization options for examples 2 and 4.

Overall, I had a great time working in google colab and attempting this lab. In just one lab, I have already learned a lot!