Goal: Automatic Text Generation in Bengali (Text type can be News)

Reference Implementation: Automatic Text Generation in Macedonian Using Recurrent Neural Networks

Their process:

* Generated two types of texts: News articles and Poems
* Model trained on a dataset in English, used for Macedonian
* ROUGE-N metric used for performance evaluation

Their steps:

1. Collect data using crawler from websites (based on html tags)
2. Clean the data:
   1. Remove special characters, html tags, Java-script functions, etc.
   2. Remove $, degree, mathematical ops, etc.
   3. Translate English words into ??
   4. Punctuation signs
3. A

My Possible Steps:

1. Collect a lot of articles on a particular topic, suppose, Cricket (part-done)
2. Take article a1
   1. Remove html tags, java-script function, special characters, etc. (done)
   2. Replace numbers with <number>
   3. Add ‘space’ before and after punctuations (for now, only full-stop) to handle them as separate ‘units’
   4. Find names by comparing to a dictionary and replace names with <name>
3. Do Step 2 for all the articles
   1. Encode ‘units’ with numbers using a dictionary (later, word embedding), then use One hot encoding (later vector) for output modeling
   2. Build the LSTM Model (what type of LSTM and parameters)

Environment Setup & Tutorial Codes:

1. <https://towardsdatascience.com/installing-keras-tensorflow-using-anaconda-for-machine-learning-44ab28ff39cb>
   1. Install and system prep
2. <https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/>
   1. Character generation using lstm rnn
   2. <https://github.com/keras-team/keras/blob/master/examples/lstm_text_generation.py>

haven’t checked this one, should have a comprehensive set of implementation

* 1. <https://towardsdatascience.com/yet-another-text-generation-project-5cfb59b26255> char-rnn to generate Trump like tweets

1. <https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/>
   1. Shows the preprocessing and word2vec (bag of words and skipgram) implementation
2. <https://towardsdatascience.com/word-embeddings-exploration-explanation-and-exploitation-with-code-in-python-5dac99d5d795>
3. <https://www.tensorflow.org/tutorials/text/text_generation>
   1. Character wise text generation using GRU RNN
   2. Have not tried yet…
4. [**https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb**](https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb)
   1. Dataset: 5000 lyrics (1M+ words, 5M+ characters) of Mexican Banda Music
      1. Data was dirty, discarded all punctuations, symbols, numbers, etc.
      2. Kept the new lines only and treated them as separate words
      3. Read the corpus ‘word by word’
      4. Filtered out the ‘uncommon words’ (f < 10)
   2. Encoded words with unique integers (0, 1, … N)
      1. Created word sequences of size 10
      2. Created sequences that do not contain the ‘uncommon words’
      3. Created 98%-2% training – test sets from the sequences
   3. Used RNN composed of LSTM units
      1. Two level stacked set of Bidirectional LSTM units
   4. Used Data Generator to ‘fit’ and ‘evaluate’ Keras functions
      1. Batch size set = 32
      2. Used one-hot encoding to encode the training set
      3. Saved checkpoints after each epoch
      4. Calculated accuracy with validation set in every epoch
      5. Implemented early stopping (if no improvement in 5 epochs)
   5. Takes about 2.5 hours per epoch on Core i7, 8 GB RAM
   6. Some sample output generated after each epoch
5. <https://medium.com/@enriqueav/update-automatic-song-lyrics-creator-with-word-embeddings-e30de94db8d1>
   1. Word Embedding Version, will check next
6. Python Bengali tokenization:
   1. <https://pynlp.wordpress.com/2013/12/10/unit-5-part-ii-working-with-files-ii-the-plain-text-corpus-reader-of-nltk/>
   2. <https://stackoverflow.com/questions/42718792/reading-bengali-with-python-natural-language-toolkit>