text_generation

November 11, 2020

The input data is first sentence segmented by NLTK's `PunktSentenceTokenizer` and then tokenized to words by `TreebankWordTokenizer`, source here. The output is then later detokenized by `TreebankWordDetokenizer` which makes the output look better. I hope this is not considered cheating, since the tokenizers and detokenizers are rule based and don't learn from the data.

Furthermore, the text generation doesn't stop after exactly `N` tokens, but after at least `N` tokens have passed and the last generated token was an end of sentence token.

[2]:
```
from ngram import *
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.tokenize.treebank import TreebankWordDetokenizer
word_detokenize = TreebankWordDetokenizer().detokenize
import random

START_SYMBOL = "<$>"
END_SYMBOL   = "</$>"

def corpus(corpus, nlength, lower, rev):
    with open(corpus, 'r') as f:
        start_n = nlength-1 if not rev else 1
        end_n = 1 if not rev else nlength-1
        data = [word_tokenize(sent) for sent in sent_tokenize(f.read() if not␣
 ↪lower else f.read().lower())]
        random.shuffle(data)
        out = [START_SYMBOL]*start_n + data[0]
        for sent in data[1:]:
            out += [END_SYMBOL]*end_n + [START_SYMBOL]*start_n + sent
        out += [END_SYMBOL]*end_n
    return out

def create_story(corpusname, nlength=2, wordcount=100, rev=False, lower=False):
    data = corpus(corpusname, nlength, lower, rev)
    if rev:
        data = data[::-1]
    ngram = BasicNgram(
        nlength, data,
        start_symbol=START_SYMBOL if not rev else END_SYMBOL,
        end_symbol=END_SYMBOL if not rev else START_SYMBOL,
        pad_left=False,
        pad_right=False
    )
    context = [ngram._start_symbol]*(nlength-1)
    output = []
    i = 0
    while (i < wordcount or context[-1] != ngram._end_symbol):
        word = ngram[tuple(context)].generate()
        context = context[1:] + [word]
        output.append(word)
        i += 1
    output = [x for x in output if x not in {START_SYMBOL, END_SYMBOL}]
    if rev:
        output = output[::-1]
    output = word_detokenize(output).replace(' .', '.')
    if lower:
        output = list(output)
        output[0] = output[0].upper()
        for (i, char) in enumerate(output):
            if char == '.' and i < len(output)-2:
                output[i+2] = output[i+2].upper()
        output = ''.join(output)
    print(output)

create_story('../data/Junglebook.txt', nlength=2)
create_story('../data/King James Bible.txt', nlength=4)
create_story('../data/poetryfoundation_clean.txt', nlength=4, wordcount=50)
create_story('../data/ads.txt', nlength=3, wordcount=30)
create_story('../data/ads.txt', nlength=4, wordcount=30, lower=True)
create_story('../data/ads.txt', nlength=3, wordcount=50, rev=True)
```

Junglebook, no post-editing:

> Kala Nag's elephants swung him sat down on with a tiger will come here–I can't touch cattle because he has kept my own use them one such as soon as he was a hundred years, but unless you are dogs, you have told even I am proud, a line was not sleep to Kaa, transcribe and dropped him. Look out in deep sleep, panting among men will be freely available by Teddy's bridle-wise?"Don't mistake me, as tiny fibers, widow will pay for the young calf (they called the bath-rooms of things made it very easy striking–raining on thy mother and he would shout together.

King James Bible, no post-editing:

> But Deborah Rebekah's nurse died, and all the cattle ringstraked. Some remove the landmarks; they violently take away flocks, and their sockets of brass four; their hooks of silver, sixteen sockets; two sockets under another board. yea, sweeter than honey to my mouth! And there came in two men, sons of Belial shall be all righteous: they shall bring unto the LORD. Then came Peter to him, saying, Let the priests, and they spoil it, so that they wearied themselves to find the door.

Poem dataset, taken from here. Newlines and fine detokenization post-edited:

```
because this was all right
I'm taking some time out
from being alive with daughters

It's OK
I'm impersonating a kiss of lilacs
a murder of chance every mouse in the dust
where I’d huddle at the foot
of his name

on thursday
he’ll endure
```

The trigrams from the commercials transcripts dataset delivered suprisingly fluent results, no post-editing:

> Respect yourself, call your Allstate agent can help you. Look to IBM - serving businesses of all wheel drive standard. What can be a little less time at the front.

Lowercasing the data provided somewhat less coherent output, but it made the model less overfit on the data, allowing to use a quadgram model. Capitalized with a script.

> Now at home. A shower that can dissolve a day. Stop, before the dodge you want goes to somebody else.

Throughout this task we assumed that text should be generated left to right. But what if we reversed the direction? The result, without post-editing, is actually not that bad:

> Designed to reduce pain, poor coloring. The world’s fastest-reacting suspension system. Five sprays for instant relief that won’t replace fast acting inhalers for sudden symptoms and should not be able to afford a necklace made of sand?

In the following snippets, the ngram models from different corpora are swichted after every word. Since the models don’t implement any smoothing, there needs to be some solution for the case where the first model generating some context which the second model never saw. I experimented first with ending the sentence and starting a new one, which provided somewhat poetric looking outputs. The following solutions just takes the next ngram model in line (after some number of unsuccessful attempts).

```
[3]: def create_story_roll(corpusnames, nlength=2, wordcount=100):
         datas = [corpus(corpusname, nlength, False, False) for corpusname in␣
      ↪corpusnames]
         ngrams = [
             BasicNgram(
                 nlength, data,
                 start_symbol=START_SYMBOL, end_symbol=END_SYMBOL,
                 pad_left=False, pad_right=False
             )
             for data in datas
```

```
    ]
    ngramcontexts = [set(ngram.contexts()) for ngram in ngrams]
    context = [ngrams[0]._start_symbol]*(nlength-1)
    output = []

    def present_in_all(word):
        return all([word in context for context in ngramcontexts])

    succ_i = 0
    while (succ_i < wordcount or context[-1] != ngrams[0]._end_symbol):
        current_ngram = ngrams.pop(0)
        ngrams.append(current_ngram)
        word = START_SYMBOL + START_SYMBOL # one token, does not exist anywhere␣
↪in the data
        tolerance = 20
        skip_ngram_loop = False
        while not present_in_all(tuple(context[1:]+[word])):
            tolerance -= 1
            if tolerance == 0:
                skip_ngram_loop = True
                # context = [ngrams[0]._start_symbol]*(nlength-1)
                break
            word = current_ngram[tuple(context)].generate()
        if skip_ngram_loop:
            continue
        context = context[1:] + [word]
        output.append(word)
        succ_i += 1
    output = [x for x in output if x not in {START_SYMBOL, END_SYMBOL}]
    print(word_detokenize(output).replace(' .', '.'))

create_story_roll(
    ['../data/ads.txt', '../data/King James Bible.txt'],
    nlength=2, wordcount=30
)
# This may take a few seconds to complete
create_story_roll(
    ['../data/ads.txt', '../data/poetryfoundation_clean.txt'],
    nlength=3, wordcount=30
)
```

```
Okay. That's where our people as I am a daughter. No one likes to be rushed. I
love you for ourselves by living in one ear and let your children are not.
```

Commercials and King James Bible switching bigram model with no post-editing:

> For in the army. And because his childhood and the ship our God I am rich more children of the house. Let me with the word for the knowledge and love which go into

> one man or my daughter, too mighty, saying I said.

Commercials and Poems switching trigram model with only new lines added:

```
The same way again.
I will go to bed.

Discover the magic of the earth
for some people are still in it.
```

I find this quite touching, especially the last sentence. No large ngrams were present in the original data.