text_generation

November 10, 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.

I would wish to claim some extra points for working with reverse n-grams, multiple external datasets and creating and switching ngram model. I also experimented with lower-casing the input data.

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

def create_story(corpusname, nlength=2, wordcount=100, rev=False, lower=False):
    data = corpus(corpusname, lower)
    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
```

```
    )
    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:

> And Jacob awaked out of his mouth goeth a sharp sword, that our wives and our children's: now then, whatsoever God doeth, it shall be cast the same in hope, that he shut up the kingdom to Israel? And the Lord shall cover him all the while that it is an heave offering: and every laver was four cubits: and the men of Keilah deliver me up into the hand of her enemies. The LORD is far from me. For if thou wert cut out of the king of Samaria.

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.

> Display the flag proudly. Gmac not only has money available, but at interest rates that make sense. They call me up.

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:

> Unlike ordinary toothpaste, you discern not only help you make it the Best Luxury SUV in 2014. You'll easily see the signs of Alzheimer's Center on site for those at least 18 who have felt this beautiful. No cars.

[ ]:
```python
def create_story_roll(corpusnames, nlength=2, wordcount=100):
    datas = [corpus(corpusname) for corpusname in corpusnames]
    ngrams = [BasicNgram(nlength, data, start_symbol=START_SYMBOL,␣
 ↪end_symbol=END_SYMBOL) for data in datas]
    ngramcontexts = [set(ngram.contexts()) for ngram in ngrams]
    context = [ngrams[0]._start_symbol]*(nlength-1)
    output = []
    i = 0
    def present_in_all(word):
        for context in ngramcontexts:
            if word not in context:
                return False
        return True
    while (i < wordcount or context[-1] != ngrams[0]._end_symbol):
        current_ngram = ngrams.pop(0)
        ngrams.append(current_ngram)
        word = START_SYMBOL + START_SYMBOL # does not exist anywhere in the data
        tolerance = 100
```

```
        while not present_in_all(tuple(context[1:]+[word])):
            tolerance -= 1
            if tolerance == 0:
                context = [ngrams[0]._start_symbol]*(nlength-1)
            word = current_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}]
    print(word_detokenize(output).replace(' .', '.'))

create_story_roll(
    ['../data/ads.txt', '../data/King James Bible.txt'],
    nlength=2, wordcount=30
)
create_story_roll(
    ['../data/ads.txt', '../data/poetryfoundation_clean.txt'],
    nlength=3, wordcount=30
)
```

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

> Our hands, and easy, sealing the wheel over ten of us, our sister, and enjoy pleasure. I run after all that simple pass on this thing: What I do them.

Commercials and Poems switching trigram model with no post-editing:

> Leave a And best. The great ones who look up to the world to contemplate And it isn't matter which Leave a Leave a Leave your shoes.