Soyoung Kim
LING-L545

Report: 01a

1. segmenter.rb' vs. the built-in sentence tokenizer in nltk vs. segmenter.py

a) segmenter.rb:

Sample outcome:

```
Faqja kryesore:
Thanas Qerama:
Thanas Qerama (7 prill, 1945 — 4 prill, 2004), ishte shkrimtar shqiptar, novelist dhe shkrues i tregimeve fanta
stiko-shkëncore, botues i revistës "Horizonti" gjatë viteve 1979—1989, revistë për moshat adoleshente me përmbl
edhje, kuriozitete dhe tregime nga shkencat natyrore dhe historike.
Tituj të veprave.
Filmi "Lundrimi i parë" është bazuar në veprën e tij me të njëjtin titull.
Dhjetori:
Dhjetori është muaji i dymbëdhjetë dhe i fundit në Kalendarin Gregorian dhe ka 31 ditë.
Emri "dhjetor" është përkthim dhe adaptim nga Latinishtja për fjalën "dhjetë" ("decem").
dhjetori ishte muaji i dhjete për Romakët e antikitetit, përpara se të shtoheshin muajt janar dhe shkurt.
Nëntori:
Nëntori është muaji i njëmbëdhjetë në Kalendarin Gregorian dhe ka 30 ditë.
Emri "nëntor" është përkthim dhe adaptim nga latinishtja për fjalën "nëntë" ("novem").
Nëntori ishte muaji i nëntë për romakët e antikitetit, përpara se të shtoheshin muajt janar dhe shkurt.
28 Nëntori është dita e Pavarësisë së Shqipërisë.
Tetori:
Tetori është muaji i dhjetë në Kalendarin Gregorian dhe ka 31 ditë.
Emri "tetor" është përkthim dhe adaptim nga latinishtja për fjalën "tetë" ("octo").
Tetori ishte muaji i tetë për romakët e antikitetit, përpara se të shtoheshin muajt janar dhe shkurt.
Shtatori:
Shtatori është muaji i nëntë në Kalendarin Gregorian dhe ka 30 ditë.
Emri "shtator" është përkthim dhe adaptim nga latinishtja për fjalën "shtatë" ("septem").
Shtatori ishte muaji i shtatë për romakët e antikitetit, përpara se të shtoheshin muajt janar dhe shkurt.
Ne shtator fillon zyrtarisht stina e vjeshtës dhe ndërrohet ora verore në atë dimërore
Gjuha esperanto:
Esperanto është gjuhë artificiale,e krijuar nga Ludwik Lejzer Zamenhof më 1887.
```

The code imports the source code from the pragmatic_segmenter, which seems to have general cleaning and replacing algorithm. It seems that one can also specify the language, if there's a language-specific source code (e.g. Chinese, Burmese, English, etc.). But many languages are not supported and such languages should resort to general/common rules. The code then segments the text in accordance with the imported source code/rules, and create a new line at the end of each sentence by adding \n. It is able to recognize parentheses as a part of a sentence and keeps running even though there is punctuation in there.

Number of sentences detected:

```
(base) sykim@Soyoungs-MacBook-Pro lib % wc -l wiki.rubysegmented
  178556 wiki.rubysegmented
(base) sykim@Soyoungs-MacBook-Pro lib % wc -l wiki.new
   99999 wiki.new
```

Note that there are some unexpected empty lines that are being counted.

Perhaps because I did not specify exception words such as p. sh. (Albanian equivalent of the English e.g.), this segmenter is not able to treat such instances as one individual token — instead, I observed that tokens like p. sh. are treated as two sentences. Conveniently enough, though, if "p. sh." was located in parentheses, segmentation did not occur and the expected segmentation was completed.

One of the less obvious “mistakes” I found was the treatment of titles and subtitles such as [TITLE]:. For example, if the original text has something like “Shtatori:\n Shtatori është…,” the segmenter treats it has two sentences: 1) Shatori:, 2)Shtatori është… I am suspecting that this instance was treated as two sentences because of the line breaker \n. Potential issues for this segmentation method is that, if there’s an original text that has multiple line breakers within a sentence for stylistic purposes, it won’t be able to recognize it as one sentence. In addition, it is debatable whether one should treat titles and subtitles as individual sentences or not, and I believe additional rules or exceptions should be implemented depending on what the user believes to be sentences or what the user is looking for in a corpus.

b) Built-in nltk sentence_tokenize:

Sample outcome:

```
['\nFaqja kryesore:\n\nThanas Qerama:\nThanas Qerama (7 prill, 1945 – 4 prill, 2004), ishte shkri
mtar shqiptar, novelist dhe shkrues i tregimeve fantastiko-shkëncore, botues i revistës "Horizont
i" gjatë viteve 1979–1989, revistë për moshat adoleshente me përmbledhje, kuriozitete dhe tregime
nga shkencat natyrore dhe historike.',
 'Tituj të veprave.',
 'Filmi "Lundrimi i parë" është bazuar në veprën e tij me të njëjtin titull.',
 'Dhjetori:\nDhjetori është muaji i dymbëdhjetë dhe i fundit në Kalendarin Gregorian dhe ka 31 di
të.',
 'Emri "dhjetor" është përkthim dhe adaptim nga Latinishtja për fjalën "dhjetë" ("decem").',
 'dhjetori ishte muaji i dhjete për Romakët e antikitetit, përpara se të shtoheshin muajt janar d
he shkurt.',
 'Nëntori:\nNëntori është muaji i njëmbëdhjetë në Kalendarin Gregorian dhe ka 30 ditë.',
 'Emri "nëntor" është përkthim dhe adaptim nga latinishtja për fjalën "nëntë" ("novem").',
 'Nëntori ishte muaji i nëntë për romakët e antikitetit, përpara se të shtoheshin muajt janar dhe
shkurt.',
 '28 Nëntori është dita e Pavarësisë së Shqipërisë.',
 'Tetori:\nTetori është muaji i dhjetë në Kalendarin Gregorian dhe ka 31 ditë.',
 'Emri "tetor" është përkthim dhe adaptim nga latinishtja për fjalën "tetë" ("octo").',
 'Tetori ishte muaji i tetë për romakët e antikitetit, përpara se të shtoheshin muajt janar dhe s
hkurt.',
 'Shtatori:\nShtatori është muaji i nëntë në Kalendarin Gregorian dhe ka 30 ditë.',
 'Emri "shtator" është përkthim dhe adaptim nga latinishtja për fjalën "shtatë" ("septem").',
 'Shtatori ishte muaji i shtatë për romakët e antikitetit, përpara se të shtoheshin muajt janar d
he shkurt.',
 'Ne shtator fillon zyrtarisht stina e vjeshtës dhe ndërrohet ora verore në atë dimërore\n\nGjuha
esperanto:\nEsperanto është gjuhë artificiale,e krijuar nga Ludwik Lejzer Zamenhof më 1887.',
 'Ludwik, i njohur me nofkën Doktoro Esperanto, pati si qëllim të krijojë një gjuhë të lehtë për
tu mësuar, politikisht neutrale që do të kapërcente kombësitë dhe do të sillte paqe dhe mirëkupti
m ndërmjet njerëzve.',
 'Burgu i Mërgimit:\n\nAstronomia:\nAstronomia, që nga etimologjia do të thotë ""ligji i yjeve"",
(greq.',
 ': astro + nomos) është shkencë që përfshin vrojtimin dhe shpjegimin e dukurive që ndodhin jasht
ë Tokës dhe atmosferës së saj.']
```

This segmenter/tokenizer is based on the model that has been pre-trained on English. The training is based on common abbreviations, collocations, and sentence-initial words. As such, in order for this tokenizer to work on another language, it must be trained on a large set of data in that language (unless it is the language that nltk already has trained the model on). The text I am using, Albanian, is not one of the languages that nltk supports, and so the tokenizer segmented the text based on English rules.

Number of sentences it generated:

```
[19]: len(sent_tokenize_list)

[19]: 152274
```

The number of the segmented sentences is still less than what segmenter.rb produced, mainly because nltk.tokenizer does not treat '[TEXT]: ' as a separate sentence, even though it is marked by \n, whereas segmenter.rb recognizes \n as one of the cues for sentence break. Given the nature of wikipedia articles, there should be many tokens that contain ':\n,' which resulted in the nltk tokenizer to produce less number of sentences. However, as mentioned above, it is debatable that such tokens are individual sentences or not, and the decision really depends on what information is sought by the researcher — e.g. is the person looking for general syntactic patterns/sketch of the corpus or the language, or is the person looking to do information retrieval, etc. While '[TEXT]: ' is clearly not a grammatical sentence, it should also not be counted as a part of the following sentence, given that these are more of the titles or subtitles of the sections. However, also as mentioned above, there are instances in which '[TEXT]: ' or ' :[TEXT]' is indeed a part of the sentence, and so the nltk's approach that uses training based on collocation seems plausible.

Just like segmenter.rb, abbreviations such as 'p. sh.' cannot be captured, primarily because the tokenizer does not have any information of Albanian words. However, while segmenter.rb was able to at least recognize that the abbreviations or clipped words at the beginning of the parenthesized phrases are not sentence breaks, nltk tokenizer doesn't seem to be able to make such general predictions:

```
(greq.',
 ': astro + nomos) është shkencë që përfshin vrojtimin dhe shpjegimin e dukurive që ndodhin jasht
ë Tokës dhe atmosferës së saj.',
```

c) segmenter.py:

Sample outcome:

```
 Astrofizika.
 Astrofizika dhe astronomia janë aplikimet e teorisë dhe metodave fizike për studimin e strukturës yjore, evolu
cionit yjor, origjinës së sistemit diellor, dhe problemeve të lidhura me kozmologjinë.
Për shkak se astrofizika është një subjekt shumë i gjerë, astrofizikantët zakonisht aplikojnë shumë disiplina t
ë fizikës, përfshirë mekanikën, elektromagnetizmin, mekanikën statistike, termodinamikën, mekanikën kuantike, r
elativitetin, fizikën bërthamore, dhe fizikën atomiko-molekulare.
 Astrofizika u zhvillua nga shkenca e vjetër e astronomisë.
Astronomët e civilizimeve të hershme bërën vëzhgime metodike të qiellit, kjo duket nga artifaktet e shumta astr
onomike të kohëve të hershme të gjetura në kultura të ndryshme.
Pas shekujsh zhvillimi nga astronomët Babiloniane dhe Greke, astronomia perëndimore u fut në një periudhë letar
gjike për katërmbëdhjetë shekuj deri në ardhjen e Nicolaus Copernicus i cili modifikoi sistemin Ptolemaik duke
vendosur diellin në qendër të universit.
Observimet e detajuara të Tycho Brahes çuan në Ligjet e Keplerit të lëvizjes planetare.
Në të njëjtën kohë teleskopi i Galileut ndihmoi në zhvillimin e shkencës moderne.
Teoria e Njutonit e gravitetit universal dha një bazë dinamike për ligjet e Keplerit.
Në fillim te shekullit te 19-te., shkenca e mekanikes qiellore arriti një stad shumë të zhvilluar në duart e Le
onhard Euler, J.
L.
Lagranzhit, P.
S.
Laplasit, dhe të tjerëve.
```

Number of sentences generated: I couldn't count the overall sentence numbers as it cannot process some empty lines and spaces. (Also see my comments in the .py file that I separately uploaded).

Also it looks like it adds an empty space wherever it detects some empty symbols, resulting in an indent-like space in front of some sentences (should probably be resolved by text-processing).

once the text is processed and cleared it seems to be able to tokenize the sentences pretty accurately — the challenge is that you have to add all possible exceptions such as e.g. and the initial of the names. Thus, if I were to run this script with one missing exception or run it on a new text file, chances are that it will incorrectly segment unknown abbreviations/exceptions into separate sentences.

2. Tokenization - maxmatch

Implementation of maxmatch - please see maxmatch.py
Instructions on how to use it:
The .py file already had the Japanese dictionary extracted from the CONLUU file. If one must use a different dictionary (or a language), then they should go back to the code and delete the the first few lines where I extracted the dictionary list, and delete the hashmark in line 41. The dictionary or word list must be pre-processed so it only contains words/tokens only.

Given that you are using the built-in dictionary, you can type in something like echo “これはテストです” | python3 maxmatch.py or python3 maxmatch.py text.txt.

Aside from whether Japanese particles and case markers such as “は” “で” and “も” should be treated as separate words (which pertains to the dictionary and the wordlist, not the algorithm issue), the algorithm seems to work very well. Below is the sample WER calculation for the string “これはテストです。記念に一緒に写真でもとりましょう。”, which only returns 13.33% error.

```
(base) sykim@149-161-172-120 ling-l545 % python3 wer.py reference.txt hypothesis.txt
REF: ['これ', 'は', 'テスト', 'です', '。', '記念', 'に', '一緒', 'に', '写真', 'で', 'も', 'とり', 'ましょう', '。']
HYP: ['これ', 'は', 'テスト', 'です', '。', '記念', 'に', '一緒', 'に', '写真', 'で', 'もと', 'り', 'ましょう', '。']
EVA:                                                  S     S
WER: 13.33%
```

The specific error comes from the treatment of “でも-とり-ましょう” as “で-もと-り-ましょう.” This is perhaps because “もと” can be found in the dictionary while “とり” isn’t, because in most written materials it should be written with its corresponding Kanji, as “撮りましょう。” Indeed, when you revise the sentence so that it contains the Kanji and run MaxMatch again, the word boundaries are captured correctly:

```
[44]: print(MaxMatch(s,d))

      ['これ', 'は', 'テスト', 'です', '。', '記念', 'に', '一緒', 'に', '写真', 'で', 'も', '撮り', 'ましょう', '。']
```

However, in many casual written conversations and children’s books you will most likely find “写真でもとりましょう” rather than “撮りましょう.” Japanese is a peculiar case in this sense,

as users can alternate between Kanji and Kana, but it does not change the fact that the dictionary and the Maxmatch algorithm would work better on certain types of literature than others.