Assignment 06:
Tokenizing and Glossing a Thai Text

handed out: January 19, 20:00
to be submitted by: January 26, 20:00

Linguistic background: Thai is an isolating language, meaning that many processing tasks can be performed on it without having to deal with morphology. However, Thai orthography only very rarely leaves spaces between words, such that what appears as a word in a Thai text is typically an entire clause.

ดาวอังคารเป็นดาวเคราะห์ลำดับที่สี่จากดวงอาทิตย์

In order to build a glosser that works in roughly the same way as the one for Indonesian from Assignment 04, you will have to tokenize the input into words based on the principle of either the shortest or the longest possible lookup. Moreover, since we are now working with a non-Latin script, an additional annotation layer in terms of a phonetic transcription needs to be processed and displayed.

Task 1: Reading the dictionary from a file [1 points]

Write a function `read_file_to_dict(input_file)` which reads in the contents of an input file in such a way that you can retrieve the contents of columns 2, 3, and 4 by the (unique) values in column 1. When processing a line, you should check whether it contains four columns of data, and only then add an entry to the dictionary. Your function should take a string `input_file` as an argument, and return a dictionary.

Example:

```
>>> file_name = "th-eng-dict.tsv"




>>> th_eng_dict = read_file_to_dict(file_name)
>>> print(th_eng_dict["อัน"])
('an', 'RPRN', 'which')
```

Task 2: Simple tokenization [1 points]

Write a function `simple_tokenizer(th_dict, sentence)` which is a combination of simple tokenizer, transliterator, part-of-speech tagger and glosser, and analyses the example text based on the dictionary data. The idea is to always look for the shortest possible chunk of characters starting at the current position which can be found in the dictionary. For instance, on the example text you would start by first looking up "ด", then "ดา", then "ดาว", and so on. As soon as an entry is found, the equivalent portion of the input is consumed, and the annotations from the dictionary added to an object representing the sentence analysis. Processing proceeds at the first position after the consumed input. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of tuples in the following format:

(thai_word, translation, category, english_glosses)

Example:

```
>>> print(simple_tokenizer(th_eng_dict, "'ดาวอังคารเป็น"))
[('ดาว', 'daaw', 'N', 'star; spot'), ('อังคาร', 'angkhaan', 'NE', 'Mars'),
 ('เป็น', "bpe'n", 'VI', 'be')]
```

Task 3: Getting tokens by simple tokenization [1 points]

Write a function `th_tokens(th_dict, sentence)` which returns the list of tokens for a Thai sentence. It should use `simple_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.
Example:

```
>>> print(th_tokens(th_eng_dict, "ดาวอังคารเป็น"))
['ดาว', 'อังคาร', 'เป็น]
```

Task 4: Getting transliterations based on simple tokenization [1 points]

Write a function `th_translit(th_dict, sentence)` which returns the list of tokens transliterations for a Thai sentence. It should use `simple_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.
Example:

```
>>> print(th_translit(th_eng_dict, "'ดาวอังคารเป็น"))
['daaw', 'angkhaan', "bpe'n"]
```

Task 5: Getting part-of-speech tags based on simple tokenization [1 points]

Write a function `th_pos(th_dict, sentence)` which returns the list of POS tags for tokens of a Thai sentence. It should use `simple_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.
Example:

```
>>> print(th_pos(th_eng_dict, "'ดาวอังคารเป็น"))
['N', 'NE', 'VI']
```

Task 6: Getting glosses based on simple tokenization [1 points]

Write a function `th_glosses(th_dict, sentence)` which returns the list of English glosses for tokens of a Thai sentence. It should use `simple_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.
Example:

```
>>> print(th_gloss(th_eng_dict, "'ดาวอังคารเป็น"))
['star; spot', 'Mars', 'be']
```

Task 7: Greedy tokenization [2 points]

Write a function `greedy_tokenizer(th_dict, sentence)` which is a combination of a greedy tokenizer, transliterator, part-of-speech tagger and glosser, and analyses the example text based on the dictionary data. The idea now is to always find the longest chunk of characters starting at the current position which can still be found in the dictionary (”greedy search”). Your function should take the dictionary `th_dict` and the string `sentence` as arguments and return a list of tuples of the following format:

(thai_word, translation, category, english_glosses).

Example:

```
>>> print(greedy_tokenizer(th_eng_dict, "'ดาวอังคารเป็น")))
[('ดาวอังคาร', 'daaw angkhaan', 'NE', 'Mars (planet)'),
 ('เป็น', "bpe'n", 'VI', 'be')]
```

Task 8: Getting tokens by greedy tokenization [1 points]

Write a function `th_tokens_greedy(th_dict, sentence)` which returns the list of tokens for a Thai sentence. It should use `greedy_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.

Example:

```
>>> print(th_tokens_greedy(th_eng_dict, "ดาวอังคารเป็น"))
['ดาวอังคาร', 'เป็น']
```

Task 9: Getting transliterations based on greedy tokenization [1 points]

Write a function `th_translit_greedy(th_dict, sentence)` which returns the list of tokens transliterations for a Thai sentence. It should use `greedy_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.
Example:

```
>>> print(th_translit_greedy(th_eng_dict, "ดาวอังคารเป็น"))
['daaw angkhaan', "bpe'n"]
```

Task 10: Getting part-of-speech tags based on greedy tokenization [1 points]

Write a function `th_pos_greedy(th_dict, sentence)` which returns the list of POS tags for tokens of a Thai sentence. It should use `greedy_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.
Example:

```
>>> print(th_pos_greedy(th_eng_dict, "ดาวอังคารเป็น"))
['NE', 'VI']
```

Task 11: Getting glosses based on greedy tokenization [1 points]

Write a function `th_glosses_greedy(th_dict, sentence)` which returns the list of English glosses for tokens of a Thai sentence. It should use `greedy_tokenizer(th_dict, sentence)`. Your function should take a dictionary `th_dict` and a string `sentence` as arguments, and return a list of strings.
Example:

```
>>> print(th_gloss_greedy(th_eng_dict, ""ดาวอังคารเป็น"))
['Mars (planet)', 'be']
```

That's it! before submitting, don't forget to test your methods one last time using `test_ex_06.py`!

Total points: 12