Assignment 04: A Simple Glosser for Indonesian

November 28, 16:00 – December 05, 14:00

Before you start, there are some preparatory steps to understand:

1. Your work for this assignment builds on a simple dictionary which contains Indonesian words and their English translations in the format {str:str}. The dictionary is stored in the file `ind_eng_dict.pickle`. In order to read the the dictionary from the file, you can use the helper function `read_dict(file_name)`.

2. In order to check whether your functions work, you can use the variables that are already created for you in the `main` block. These two variables are also used as arguments in the examples.

```
example = "ayah saya tidak di rumah sekarang."
ind_eng_dict = read_dict("ind_eng_dict.pickle")
```

3. In previous assignments, you were writing functions from scratch. This time, you will have to make some modifications in the existing code in Task 3. This emulates a common real-life scenario where you will not write code from scratch, but interact with and build on code written by other people.
   There are comments in the task file about which code you should be editing (e.g., lines that contain the `pass` placeholder) and which code you should not modify (e.g., the helper functions already created to help you). Please pay attention to these instructions!

1 Translating from Indonesian to English [2 points]

Write a function `translate_ind_eng(sentence, lang_dict)` which 'translates' Indonesian input sentences into English by producing a string of English tokens, each of which represents the meaning of one Indonesian token in the input. Whenever there is a punctuation mark (comma, dot, exclamation mark, question mark or semicolon), you need to reproduce it in the translation. If there is a word, you have to replace it by the translation(s) from the dictionary. Your function should take a string `sentence` and the dictionary `lang_dict` as arguments and return a string.

Example:

```
>>> translate_ind_eng(example, ind_eng_dict)
'father I/my not in/at house/home now.'
```

2 Creating a nested data structure [2 points]

Write a function `build_data_structure(sentence, lang_dict)` which tokenizes the sentence and creates a data structure containing the tokens of the Indonesian sentence together with their English

glosses. This function should ignore punctuation, i.e., punctuation tokens will not be represented in the structure. Your function should take the string `sentence` and the dictionary `lang_dict` as arguments and return a list of tuples of the following format:

(Indonesian_word, [list_of_English_translations]).

You could use a helper function `tokenize(sentence)` like in Assignment 03 for the tokenization.

Example:

```
>>> build_data_structure(example, ind_eng_dict)
[("ayah", ["father"]), ("saya", ["I", "my"]),("tidak", ["not"]),
("di", ["in", "at"]), ("rumah", ["house", "home"]), ("sekarang", ["n
```

3 Getting the possible glosses [3 points]

Finish the function `get_possible_glosses(sentence, lang_dict)` which creates a gloss in the form of matrix where different translation variants appear in different rows of the same column. It should take the string `sentence` and a dictionary `lang_dict` as arguments and it should return a matrix with glosses.
Some parts of the function are already implemented, you only need to finish it. Places where you should write your code contain the keyword `TODO`. The work can be split into several subtasks:

1. Retrieve separately the Indonesian words and their English translations from the data structure that is returned by the function from Task 2.
2. Create a matrix for storing possible translations. The matrix is already initialized for you but you need to determine the necessary number of rows and columns.
   (a) Get the number of columns. The number of columns is equal to the length of the list of Indonesian words. For example, our example sentence has 6 words (punctuation is ignored).
   (b) Get the number of rows. The number of rows is equal to one plus the maximum number of possible translations. For example, in our example sentence, the words *saya*, *di* and *rumah* have 2 possible translations, which is the maximum, so you will need 3 rows.
3. Fill in the matrix. In each column, you need to iterate through the list of translations and, depending on the number of possible translations for each word, fill in the cells. The layout of the desired matrix for the example is like this:

"ayah"	"saya"	"tidak"	"di"	"rumah"	"sekarang"
"father"	"I"	"not"	"in"	"house"	"now"
""	"my"	""	"at"	"home"	""

Use the predefined helper function `print_possible_glosses(possible_glosses)` to check the output.

Example:

```
>>> pg = get_possible_glosses(example, ind_eng_dict)
>>> print_possible_glosses(pg)
ayah    saya    tidak   di      rumah   sekarang
father  I       not     in      house   now
        my              at      home
```

4 Generating all possible translations [3 points]

Write a method `get_all_translations(sentence, lang_dict)` which generates all combinatorically possible translations (in mathematical terms, the Cartesian product of all gloss lists) of the sentence from Indonesian to English. It should take the string `sentence` and a dictionary `lang_dict` as arguments and it should return a list of strings. Build on the data structure from Task 2, and `itertools`!

Example:

```
>>> get_all_translations(example, ind_eng_dict)
["mother and father I not in house now",
 "mother and father I not in home now",
 "mother and father I not at house now",
 "mother and father I not at home now",
 "mother and father my not in house now",
 "mother and father my not in home now",
 "mother and father my not at house now",
 "mother and father my not at home now"]
```

5 Tests [ungraded]

That's it! before submitting, don't forget to test your methods one last time using `test_ex_04.py`!