Assignment4
Error Correction

Introduction

- This assignment want you to correct senteces with spelling errors or gramatical errors.
- There are three tasks to be achieved, and each task has one unfinished function. The first two both generates some candidates from the input sentence, and then use a ready-made language model “kenlm” to choose the best one.
- Other part has been done in the ***correct_error.py***, please complete it to finish this assignment.

Task 1 - Spelling Check

- You need to complete a function named ***spelling_check***.
- The function checks the spelling and corrects errors.
- The function finds out tokens that don’t exist in big.txt and generates candidates of these tokens by ***edits2*** function with ***suggest*** function, and then use kenlm to choose best one.
- Let’s understand these functions first!

Spelling Corrector - edits1, edits2 and suggest

- This part has been completed by Peter Norvind and put on his website, but we made some changes.
- Three of the most important functions are ***edits1***, ***edits2*** and ***suggest***.
- edits1:

```
def edits1(word):
    "All edits that are one edit away from `word`."
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)]
    deletes    = [L + R[1:]               for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in letters]
    inserts    = [L + c + R               for L, R in splits for c in letters]
    return set(deletes + transposes + replaces + inserts)
```

Spelling Corrector - edits1, edits2 and suggest

- edits2: It does what edits1 does twice

```
def edits2(word):
    "All edits that are two edits away from `word`."
    return (e2 for e1 in edits1(word) for e2 in edits1(e1))
```

- suggest:

  we modify it from correction and candidates by Peter. It returns top-5 words as suggestion. The rank is based on the probability of word frqurency in big.txt.

kenlm

- Kenlm is a language model inference by Kenneth Heafield.
- The fundamental method used in kenlm is similar to the method we learned last week. However, there’re more detailed processing in kenlm, so the result of kenlm will be better.
- We use kenlm in this assignment to choose the best candidate.
- The trained kenlm model will be provided for you.
- Also, the longer the sentene is, the lower the LM score is. Hence, we need to do smoothing on it. Please use the code:

  model.score({your_sent}, bos=True, eos=True) / len({your_sent})

  to get the result.

Example

- Input: **nobady knows thats thing**
- Process:
  - Find out the token that doesn’t exist in big.txt (The tokens exist in big.txt have been stored in WORDS for you): **nobady, thats**
  - Use suggest function to generate the candidates of tokens above:

    nobady: [nobody]

    thats: ['that', 'this', 'what', 'has', 'than']

Example

- Process(cont.):

  nobady: [nobody]

  thats: ['that', 'this', 'what', 'has', 'than']

  - Generate candidates with all combinations:

    nobody knows that thing

    nobody knows this thing

    nobody knows what thing

    nobody knows has thing

    nobody knows than thing

  - Use kenlm to get the best result wtih highest score
- Output: ***nobody knows that thing***

Task 2 - Preposition and Article Check

- In this task, you need to complete ***prep_check***.
- The function checks the prepositions and articles and corrects these errors.
- There are two sets threcording all prepositions and articles that should be considered.

  They are preps and atcs.

  atcs = {"", "the", "a", "an"}

  preps = {"", "about", "at", "by", "for", "from", "in", "of", "on", "to", "with"}

Example

- Input: **at an afternoon**
- Process:
  - If the token is a preposition or an article, please consider all tokens in sets to generate candidates for all combinations:
    afternoon, at afternoon, on afternoon, about afternoon, in afternoon, ……
    an afternoon, at an afternoon, on an afternoon, about an afternoon, in an afternoon, ……

Example

- Process(cont.):
  - Use kenlm to get the best result wtih highest score

- Output: ***in the afternoon***

Task 3 - Combination of Task 1 & 2

Complete the ***process_sent*** function that considers spelling, prepositions, and articles at the same time.

Some Instructions

- Install kenlm:

  pip install https://github.com/kpu/kenlm/archive/master.zip

- If you are a windows user, please follow the installation guide in the annoncement on eeclass.
- Put ***big.txt*** and kenlm model ***bnc.prune.arpa*** in the same path as ***correct_error.py*** which have all been uploaded to eeclass.

Input and Expected Output [Task 1]

- Input: he sold everythin escept the housee
- Output:

  Text: he sold everythin escept the housee

  Candidates: (Because there’re two many candidates, we set only printing 30.)

  he sold everything except the house

  ……

  ……

  Number of Candidate: 50

  Result: he sold everything except the house

Input and Expected Output [Task 2]

- Input: look on an picture in the right
- Output:

  Text: look on an picture in the right

  Candidates: (Because there're two many candidates, we set only printing 30.)

  look picture right

  ……

  ……

  Number of Candidate: 1936

  Result: look at the picture on the right

Input and Expected Output [Task 3]

- Input: we descuss a possible meamin by that
- Output:

  Text: we descuss a possible meamin by that

  Result: we discuss the possible meaning of that

TAs' Note

- Remember to make an appoiment with TA to demo/explain your implementation before 10/14 15:30.
- You should also save your file as {student_id}.py and submit it to eeclass.
- Late submission is not allowed.