Homework 8:
NLTK

Benjamin Roth, Marina Sedinkina
Symbolische Programmiersprache

Due: Thursday Januar 16, 2020, 16:00

In this exercise you will:

- use NLTK to analyze text and perform its morphological analysis
- implement a language guesser

Exercise 1: Lexical information and Morphological Analysis [10 points]

Take a look at `hw08_nltk/analyze.py`. In this exercise you will have to implement some methods in class Analyzer, that can analyze any text. On the course homepage, you can find the file `ada_lovelace.txt`. Download it into the `data/` folder of your project and use the class Analyzer to analyze it and perform its morphological analysis.

This homework will be graded using unit tests by running: `python3 -m unittest -v hw08_nltk/test_analyzer.py`

Implement the following methods:

Exercise 1.1: Lexical information

- `__init__`(self, path) - should read the file text, create the list of of words (use `nltk.word_tokenize` to tokenize the text), and calculate frequency distribution of words (use `nltk.FreqDist`)
- numberOfTokens(self) – should return the number of tokens in the text
- vocabularySize(self) – returns the size of the vocabulary.
- lexicalDiversity(self) – returns the lexical diversity of the text.
- getKeywords(self) - returns words as possible key words, that are longer than seven characters and occur more than seven times (sorted alphabetically)
- numberOfHapaxes(self) – returns the number of hapaxes in the text
- avWordLength(self) – returns the average word length of the text.

Exercise 1.2: Morphological Analysis

An important problem in computational linguistics is morphological analysis. This consists of breaking down a word into its component pieces, for example *losses* might be broken down as *loss + es*. In English, morphology is relatively simple and is mostly comprised of prefixes and suffixes. To get an idea of what suffixes are common in English (and thus could be morphemes), we can look at the frequencies of the last n characters of sufficiently long words.

Implement additional methods in class Analyzer, that can perform morphological analysis:

- topSuffixes (self) - returns top 10 most often seen suffixes of length 2. We define a n-character suffix as the last n characters of any word of length 5 or more, thus ignore any word shorter than five characters
- topPrefixes(self) - returns top 10 most often seen prefixes of length 2. We define a n-character prefix as the first n characters of any word of length 5 or more, thus ignore any word shorter than five characters
- tokensTypical(self) - returns first 5 tokens of the (alphabetically sorted) vocabulary that contain both often seen prefixes and often seen suffixes in the corpus

Exercise 2: Language Guesser [4 points]

Implement a language guesser that determines the language it thinks the text is written in. The decision should base on the frequency of individual words in each language. Take a look at the file `hw08_nltk/model_lang.py`. In this exercise you will have to complete some methods to make it work.

This homework will be graded using unit tests by running: `python3 -m unittest -v hw08_nltk/test_lang_guesser.py`

1. Complete the class method `build_language_models(self)`. This method should return a conditional frequency distribution where:
   - the languages are the conditions
   - the values are the frequency distribution of lower case words in corresponding language
2. Complete the class method `guess_language(self,language_model_cfd, text)`:
   - it should calculate the overall score of a given text based on the frequency of words accessible by `language_model_cfd[language].freq(word)`.
   - it should return the most likely language for a given text according to the scores