Assignment 2: Natural Language Processing

2ID90 Q3, 2016-2017

March 17, 2017

1 A rudimentary spell checker

The purpose of this assignment is to design a basic spell checker in Java. The program you are going to write (or rather complete), must be able to correct misspelled words in the context of a sentence, but should, of course, not alter correct words. Being correct in the assignment means two things: the word belongs to the reference vocabulary, and the word should fit semantically in the enclosing sentence. To make things easier, we consider lower case text only, without interpunction, and use a single space character as white space between words.

As an example sentence, one may think of "i am loking for a new car" which should be corrected to "i am looking for a new car" and not to "i am locking for a new car". Also the sentence "i am booking for a new car" is not correct, although all its constituent words are, and should be corrected to "i am looking for a new car" again.

2 Materials

You are given the following: 1) a data set, 2) a start-up Netbeans project and 3) a test file. The data set consists itself of 4 files, including (i) the full corpus `sample.ascii.txt` from which the vocabulary and the unigrams and bigrams are extracted, (ii) the vocabulary `samplevoc.txt`, (iii) unigrams and bigrams[1] with counts in `samplecnt.txt` (which are contaminated), and finally (iv) a file `confusion_matrix.txt`, that has counts how often particular spelling errors are made[2,3]. The Netbeans-project provides code to read the files of the dataset and provides the methods that are supposed to return the correct phrase or word, given a possibly misspelled phrase. The test file `test-sentences.txt` contains some example phrases that the program should work on for testing purposes.

[1] Including start of sentence (SoS) and end of sentence (EoS) indicators.

[2] If, for instance, in a dataset 200 times 'a' was mistakingly typed instead of 'e', the file contains the line a|e 200.

[3] For more explantion, see article "A Spelling Correction Program Based on a Noisy Channel Model" by Kernighan, Church, and Gale.

3 Requirements and assumptions

- The assignment is done in pairs and submitted through Momotor. Your submission consists of exactly 5 java files `SpellChecker.java`, `CorpusReader.java`, `ConfusionMatrixReader.java`, `SpellCorrector.java, and WordGenerator.java`. Optionally, you may want to use an improved version of the auxilliary files, like `samplecnt.txt` and `samplevoc.txt`, which then should be submitted too.
- The submitted code is assumed to be Java 8.

- All sentences that are input to your program contain only lower case characters a-z. There is no interpunction. Hence, the alphabet for the words consists of 26 characters.
- All misspelled words have a Damerau-Levenshtein distance of 1, which means that for correction they need 1 insertion, deletion, transposition, or substitution.
- All words that should be in your output on a given test phrase are in the vocabulary. A phrase can have' 0, 1 or 2 misspelled words, but misspelled words are never consecutive. Keep into account that each word in a phrase is either correct or wrong.
- The evaluation in Momotor assumes that each of your output sentences is preceded by the string `"Answer: "`[4].
- Momotor will run two early-out tests to see if your program 1) outputs the right format and 2) can handle two simple sentences.

[4] That is, the string `"Answer:"` without the quotes and followed by a single space.

4 *Grading*

The assignment is to be graded on three criteria: the report, the test results, and additional implemented and documented features, like advanced smoothing, performance improvements, improvement of the bigrams counts, etcetera.

For the report, it is important that the structure of the report is clear. It should be clear from the report how the main problem of the assignment is divided into smaller subproblems and how these are solved. It should at least contain a description of how the main problem is divided into subproblems, how these subproblems are solved, why some choices were made, what kind of input the program would not work on and why and a brief description on the program workings. A template report will be provided. Please note that the report is the most important part of your grade.

Following the general rules, your report does not exceed 8 pages, excluding an optional small appendix. The evaluation of your code will be obtained from running a number of sentences with misspellings and real word errors (extending the sentences in the test file `test-sentences.txt` provided).

5 *Base code*

A NetBeans project is provided for which a number of methods needs to be completed. In general, you need to do the following.

1. `SpellChecker::main`[5] is the `main` method in which an object `of the class SpellCorrector` is created and used to correct a single sentence.
2. The class `CorpusReader` provides auxilliary functionality. Its smoothing method must be completed to obtain decent spelling correction.

[5] It can be easily adapted to do either local testing or testing in Momotor

3. The class `ConfusionMatrixReader` itself does not need to be changed.

4. In the `method CorpusReader::getSmoothedCount` you need to implement smoothing.[6]

5. `You need to complete` the method `SpellCorrector::correctPhrase, which` deals with the correction at the sentence level according to the noisy-channel model combined with bigram information.[7]

6. The method `SpellCorrector::getCandidateWords` collects not only all words from the vocabulary that have edit-distance 1 to a word from the given sentence, but also their noisy channel probability[8]. For each collected word this probability is to be computed in the method `WordGenerator::probability`.

7. You may want to tune constants, like NO_ERROR and LAMBDA, to improve the reach of your program.

[6] The simplest, but not the best solution, is to use add-one smoothing.

[7] For the channel probability you may want to calibrate the weight of the prior, e.g. using a construction like `likelihood * Math.pow(prior,LAMBDA) * SCALE_FACTOR`.

[8] `That` is, the conditional probability of a presumably incorrect word calculated from the confusion matrix and the type of error that was made: deletion, insertion, substitution, or transposition.

6 Closing remark

Submitted files will be checked immediately for eligibility which may cause delay in the feedback and may lead to possible congestion close to the submission deadline. Therefore, it is advised to hand in your submission in time.