Assignment 2: Natural Language Processing

2ID90 Q3, 2015-2016

March 11, 2016

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 not alter correct words of course. 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 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. Regarding the latter, if, for instance, in a dataset 200 times 'a' was mistakingly typed instead of 'e', the file contains the line a|e 200. [2] 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.

3 *Requirements and assumptions*

- The assignment is done in pairs and submitted through Peach. Your submission consists of exactly 4 java files `SpellChecker.java`, `CorpusReader.java`, `ConfusionMatrixReader.java`, and `SpellCorrector.java`. Optionally you may want to use an improved version of the auxilliary files `samplecnt.txt` and `samplevoc.txt`, which then should be submitted too.

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

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

- The submitted code is assumed to be Java 8.
- All words that are input to your program are in lower case and there is no interpunction or any other kind of character possible, except for the single apostrophe (′) like in "don′t" or "can′t". Hence, the alphabet for the words consists of 27 characters.
- All misspelled words have a Damerau-Levenshtein distance of at most 1, which means that each input word to your program needs only be altered by at most 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 Peach assumes that each of your output sentences is preceded by the string “`Answer: `“[3].
- Peach will run two tests to see if your program outputs the right format and can handle two simple sentences.

4 *Grading*

The assignment is to be graded on three criteria: the report (5 pnts), the test results (3 pnts) while the remaining points (2 pnts) are reserved for additional features of your solution when clearly described in the report and supported by your implementation (e.g. advanced smoothing, back-off, performance improvements, improvement of the bigrams counts, etc.).

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. 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 the main method as given, a sentence can be given as an argument to the `spellCorrector` method of the

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

`SpellCorrector sc` directly. In the final version the sentence needs to be read from standard input. The class `CorpusReader` provides auxilliary functionality. Its smoothening method must be filled to obtain decent spelling correction. The class `ConfusionMatrixReader` doesn't need to be changed. The overall structure in the class `SpellCorrector` is there, but code is missing at most places.

1. In the file `CorpusReader.java` you need to adapt the smoothening method `getSmoothedCount.` The simplest, but not the best solution, is to use add-one smoothening.

2. In the file `SpellCorrector.java` three methods need to be extended:

    (a) The method correctPhrase deals with the correction at the sentence level according to the noisy-channel model combined with bigram information. 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.`

    (b) The method `getCandidateWords` is intended, in view of the assumptions, to collect all words from the vocabulary that have edit-distance 1 to a word from the given sentence. For each collected word the noisy channel probability must be computed and returned. `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.

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

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.