Practical 1: T9 Texting

Text input on mobile phones can be done by the T9 system, by which each digit is mapped to a set of possible letters. See also Section 3.4 of the NLPP book. Where a sequence of input digits could correspond to more than one word, the T9 system proposes the most likely matching word, possibly taking into account one or more of the previous words.

Here we'll be creating our own word guesser. There should be a function that takes a string of digits. Each digit corresponds to three or four letters, as in T9. From all strings of letters that are consistent with the input string of digits, the function outputs the most likely string, according to a bigram model. The context of a word is not taken into consideration. Nor is it checked whether the guessed words actually exist (e.g. in a dictionary). We moreover ignore capitalisation (either ignore words not consisting entirely of lower-case letters, or convert all words to lower-case).

Do training of the bigram model according to one of the corpora in NLTK. Before you do this, I recommend you study the examples in the NLPP book relevant to N-grams. (E.g. Section 1.3 and Chapter 5.) However, the book is mostly concerned with word bigrams, whereas this practical is about letter bigrams.

The code (in Python) should be uploaded onto MMS. There should be good code commenting and an adequate programming style. Add a README file that tells me how to run the code. A report is optional; see below.

Grading and extensions

Grading is according to the school handbook. The basic requirements above earn you up to grade 17 if all is done well. Higher grades require additional work. Suggested extensions are:

- Evaluation of accuracy, by measuring how often a word can be guessed correctly. Be careful to keep training and test data disjoint. Results of the evaluation can be written up in a report.
- Further analysis of errors, represented for example in terms of confusion matrices.
- Generalisation of the code to N-grams. Additional evaluation, to be added to the report, may compare the accuracy for different values of N.
- Consideration of smoothing, interpolation or backoff.

Submission

This practical is worth 50% of the coursework for this module. The deadline is as indicated on MMS.

Additional hints

- Start-of-input and end-of-input symbols are needed for correct implementation of N-gram models.
- One way of finding the most likely word is to first generate all words satisfying the input string of digits, and then to compute the probability of each such word.
- Write many small functions solvings parts of the problem and test each of them using the Python interpreter before writing larger functions.