COM3110/4115/6115: Lab Class

Segmenting Words in Chinese Text

In this exercise you will build a simple Chinese word segmenter.

1. Introduction:

Chinese has a *logographic* writing system in which individual symbols, or *characters*, may represent whole words, or even phrases. Some words, however, are represented by *sequences of several characters*, and many characters that can stand alone as one word can also be part of the character sequences representing other words. Furthermore, Chinese writing has no explicit marking of word boundaries, c.f. the use of spaces in English. Thus, a Chinese sentence is written as a consecutive string of characters, such as:

> 中文句子由连续的一系列单词组成。
> (TRANSLATION: a Chinese sentence is formed by a consecutive string of words)

The above sentence would usually be segmented into words as follows:

中文	句子	由	连续	的	一系列	单词	组成
Chinese	sentence	cause/with	consecutive	adjective marker	a series of	word	formed

Human readers can easily identify how the characters group into words, based on their meaning. This task is more difficult for a machine, however, but is required before various other tasks can be done, e.g. counting word occurrences or building language models. Hence, this task – known as *Chinese word segmentation* – is an important step in Chinese text processing.

2. Greedy match algorithm

The *greedy match* algorithm, also called the *maximum match* algorithm, is a simple technique for Chinese word segmentation. The algorithm starts at the beginning of a sentence and attempts to find the longest word from a predefined word list (dictionary) that starts at that position. It then moves forward to the position at the end of that word, and proceeds from there. If no match is found in the word list, the algorithm simply treats the single next character as the next word. Analysis shows that most Chinese words are 5-characters long or less, so we can set this as the maximum word length, to limit the strings that must be considered during search.

For example, consider segmenting the sentence: 中文句子由连续的一系列单词组成
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

The algorithm starts at the left, i.e. position 0. The group of length 5 here ("中文句子由") is not in the dictionary, nor those of length 4 ("中文句子") or 3 ("中文句"), but that of length 2, i.e. "中文" (Chinese), *is* in the dictionary, and so is accepted as the first word. The algorithm then advances to the end of this word, i.e. position 2, and repeats until there is no more input.

3. Python Unicode notes ...

Within a program, Unicode characters and strings can be manipulated directly, but when Unicode characters are stored in files or displayed on a terminal they must be encoded as *one or more bytes*. Some encodings (such as ASCII and Latin-2) use a single byte, and can only support a small subset of Unicode, enough for a single language. Other encodings (such as UTF-8) use multiple bytes, so they can represent the full range of Unicode symbols. We will be using UTF-8 to encode our Chinese text. Fortunately, Python (3) is fully able to handle Unicode text, but it may be necessary to specify the particular encoding of a file when a filestream is opened for reading from, or writing to, that file. We can do so using the "`encoding`" keyword argument of the `open` command, as in the following example:

```
with open("filename.txt", "r", encoding = "utf-8") as myfile:
    for line in myfile:
        ... # process your text as you would normally
```

In fact, UTF-8 may well be the default encoding on your system, but it is good practice to be explicit about the encoding in use.

4. Exercise

Download the file `chinese_segmentation_resources.zip` from the module homepage and unzip it to get the following files:

```
chinesetext.utf8
chinesetext_goldstandard.utf8
chinesetrad_wordlist.utf8
eval_chinese_segmentation.py
chinese_segmentation_STARTER_CODE.py
```

The file `chinesetext.utf8` contains Chinese text (one sentence per line) that has been drawn from the Sinica Treebank corpus of traditional Chinese (as made available in the NLTK). (If you are unable to read this text directly, you might try opening the file in a browser, and selecting the translate option, to see an English translation.) The file `chinesetext_goldstandard.utf8` contains a version of the same text, but with gold-standard word boundaries marked, by the addition of spaces.

Rename the code file `chinese_segmentation_STARTER_CODE.py`, and then add your own code, so as to load the dictionary of Chinese words (`chinesetrad_wordlist.utf8`), and to use that in your own implementation of the *greedy match algorithm*, writing the results to an output file, as UTF-8, with spaces added for word boundaries.

You can the score the performance of your system by using the evaluation script supplied, calling this (e.g.) as in the following example:

```
python eval_chinese_segmentation.py chinesetext_goldstandard.utf8 MYRESULTS.utf8
```