Tutorial 1

Python
Tokenisation and Segmentation
Indexing and Search
Evaluation

Python

- Many ways to learn – here are some resources:
  - https://docs.python.org/3/tutorial/
  - http://www.slideshare.net/MattHarrison4/learn-90

- We'll visit a Sheffield tutorial on this

Tokenisation

- Goal: convert stream of bytes to tokens
- Simple method: Python's .split() function
- Let's get some data and play with it!
  - import nltk
  - nltk.download()
- Go to *Corpora*, find the *Gutenberg* corpus
- Let's load it
  - from nltk.corpus import gutenberg
  - text = gutenberg.raw("carroll-alice.txt")
  - print(“\n”.join(gutenberg.fileids()))

Whitespace Tokenisation

- Assume that tokens are whitespace-delimited
- Implement a tokeniser using split()
- Write a program that:
  - Loads and tokenises the text
  - Prints the number of tokens it contains

Word Tokenisation

- Now use NLTK's word_tokenize() function to find token boundaries
  - nltk.download()
  - Get punkt – a tokenisation package (for words and sentences)
- Write a program
  - That reads and tokenises the text
  - Prints the number of tokens it contains.
- How does this compare to using whitespace tokenisation? Why?

Chinese Word Segmentation

- Chinese has no explicit word boundaries

中文句子由连续的一系列单词组成

- Greedy Matching
  - Use a list of known words
  - Find longest possible matches

Chinese Word Segmentation

- Algorithm:
  - Start at the beginning of the sentence
  - Find longest sequence of (up to *n*) consecutive characters in the list
  - If a match is found, assume that's the next word
    - Store it
    - Move on
  - Otherwise, assume a single-character word
  - Continue until the end of the sentence

Chinese Word Segmentation

- Download
  - Unsegmented text
  - Word list

- Evaluate performance using the evaluation script
  - What effect does tuning n have?

Indexing

- Goal is to build a word-to-document reference
- We can store this with shelve
  - import sys
  - index_filename = sys.argv[1]
  - import shelve
  - with shelve.open(index_filename, 'c') as index:
    - index['x'] = [1,2,3]
    - index.sync()
- This means we only need to do our indexing once

Indexing

- Write a program that:
  - Takes an index filename
  - Opens the gutenberg corpus
  - Goes through the documents in the corpus, tokenising each one
    - for document_id in gutenberg.fileids()
      - tokens = nltk.word_tokenize(gutenberg.raw(document_id))
  - For each token, adds an entry in the index containing that document's ID
    - if term not in index:
      - Index[term] = []
    - index[term].append(document_id)

Searching

- Open the index
- Take a query string
  - sys.stdin.readline()
  - .strip()
- Tokenise the query string
- Find matching documents
  - Look up the entries in the shelved index
    - Lists are not unique
    - set(list)

Ranking

- Update the indexing program to include TF
  - term_count = document_tokens.count(term)
  - index[term].append((document_id, term_count))
- DF is just len(index[term])
- TF.IDF = TF / DF
  - Variants:
    - +1 smoothing to IDF
    - Take logs on both sides
- Calculate TFIDF for all documents in list, then rank
- Congratulations! Your own pre-Google search engine

Evaluation

- Evaluating IR:
  - How much did you find?
  - How much did you find, that shouldn't have been there?
  - How much did you miss?

- Bonus exercise:
  - Repeat the above, for Chinese or Russian!
  - Xinhua, Russia Today

Assignment

- Read in a collection; run some queries over it
- Total 500~1500 words
  - Discuss performance
  - Identify problems
  - Suggest methods for improvement
- Include a copy of your source code
  - For an A grade, extend the system beyond this specification, and describe your extension
  - Due October 6
  - Mail me: **leonderczynski@gmail.com**