Text Technologies – Exercise 3

s0840449

1 Design

Preprocessing: Two lines were removed from each speech: the header line and the speaker's title. Line length was used to detect if the latter existed - less than 8 words (chosen by manual testing) indicated a title line. The text was also processed to remove punctuation and any empty lines.
Exact Duplicate Detection: Implemented using adler32[1] fingerprints of documents stored in an inverted index - both choices for efficiency. Duplicates were found by iterating over the index and outputting results that contain more than one speech. The algorithm found all of the exact duplicates in the training set, and all results found in the real data were duplicates too.
Near Duplicate Detection: Implemented using the simhash fingerprint[2]. This was created by: removing 'stop-words'[3] from the speech; hashing each remaining word using md5; replacing '0's in the result with '-1's; multiplying each hash by the frequency of the word; and finally doing a column-wise sum and converting back to binary. The bit-size of the simhash was set to 128.

To compare documents, both the 'L groups of k bits' (with L = 32, k = 4) and the brute force comparison methods were implemented. The former matched two documents if they appeared in at least 27 groups together, the latter if the hamming distance was less than 5.
Plateau-based Duplicate Detection: Implemented using Finn's method[5]. The document was converted to a vector X; $x_i = 1$ if x_i was a tag (non-number), 0 otherwise. The vector was then converted to a list of 'runs' of tags and non-tags[1], and three other arrays found: $b_i \in B$ was the number of words (tag or non-tag) that came before run i; $t_{b,i} \in T_b$ was the number of tags that came before run i; and $t_{a,i} \in T_a$ was the number of tags that came after run i. The best indices were then found by calculating the following for all run indices (i, j), where $i < j$ and r_k stands for the index of the start of run k in X:

$$\sum_{p=1}^{r_i-1} x_p + c \sum_{p=r_i}^{r_j} (1 - x_p) + \sum_{p=r_j+1}^{n} x_p = t_{b,i} + 100 * tokens\_between_{i,j} + t_{a,j}$$

This algorithm is $O(n^2)$, as $tokens\_between_{i,j}$ is just the difference of the tokens before j and the tokens before i, which can each be calculated as the number of words before j and i (b_j, b_i), minus the number of tags before j and i ($t_{b,j}, t_{b,i}$). Only plateaus with at least 3 numbers in them were taken. Near duplicate detection was then applied on the document plateaus to find duplicates.

2 Duplicates

Aside from exact duplication, speeches were plagiarised by: inserting or removing a few words or a sentence ('I would **like to** stress' or 'flexibility **of this proposal** to the Member States'); inserting or removing whitespace; and replacing words with others with similar meaning, ('technique' with 'strategy' or 'see' with 'have'). On a few occasions sentences were moved from one place in the document to another (such as in 200516.txt & 9014773.txt). Interestingly I found only pairs of duplicated documents; there were no instances of a third person copying an already copied speech.

[1] Runs were used as there is never any reason to stop in the middle of a set of tags or non-tags, meaning they can be consolidated into a single section.

References

[1] P. Deutsch, J.-L. Gailly, *ZLIB Compressed Data Format Specification version 3.3*, Network Working Group Request for Comments (RFC) 1950, May 1996

[2] M. Charikar, *Similarity Estimation Techniques from Rounding Algorithms*, Proc. of the 34th Annual ACM Symposium on Theory of Computing, STOC (2002).

[3] NLTK Stopwords Corpus (Porter et al). See http://nltk.googlecode.com/svn/trunk/doc/book/ch02.html

[4] Python standard library md5 implementation. See http://docs.python.org/library/hashlib.html

[5] A. Finn, N. Kushmerick, B. Smyth, *Fact or Fiction: Content Classification for Digital Libraries*, 2001.