Find the Intruder

Table of contents

The Problem

The 1,300 newspaper articles on Georgia lynchings were manually entered into a database[i], cross-referencing each of the 392 Georgia lynching events to the appropriate newspaper articles. A further 4,000 newspaper articles published across America were added to the database. Upon close inspection of the cross-references between newspaper articles and lynching events, several errors were discovered, with some newspaper articles "intruding" into the wrong story (i.e., lynching event). As different lists of errors were compiled, the question was raised: Is there a way to check automatically for cross-referencing errors? A way to obtain a list of newspaper articles that have nothing to do with the event to which they were cross-referenced as data sources for the event?

The Computational Solution

FindNotRelatedDocuments.py – To understand that question, suppose that you have n documents in a cluster (e.g., a lynching event). You want to search through all n documents to see whether they talk about the same event on the basis of social actors and NER values for Location, Date, Person, Organization. The logic is similar to the summary_checker.py, but contrary to the summary_checker.py, the script does not check each document against a standard document (the "compilation"). Rather, to identify the intruder, the script checks each of the n documents in an event against each of all other n-1 documents in the event. And for each document it builds an index of relativity to be used comparatively with all the documents in the cluster. We calculated the relativity index by using a modified version of Jaccard index: The length of the intersection between social actors in the current document and those in the folder divided by the sum of the length of this intersection and the social actors in the current document. Documents with index of relativity significantly lower than the rest of the cluster are signaled as not likely to belong to the cluster.

The script, also written in Python 3, takes the list of social actors obtained from social_actors.py and checks the values in the list against the nouns found in all the documents in the cluster. Nouns for each document are tagged using Stanford CoreNLP (https://stanfordnlp.github.io/CoreNLP/) (POSTAG, Part Of Speech Tag, NN, noun singular, NNP, proper noun singular, NNS, noun plural, and NNPS, proper noun plural). These nouns are then checked to see if they are in the WordNet social actor list. With Stanford CoreNLP the script also extracts NER values (Named Entity Recognition). For each document, the script builds an index of relativity based on the social actors and NER values of Location, Date, Person, Organization mentioned in the document. We calculated the relativity by using a modified version of Jaccard index: The length of the intersection between social actors in the current document and those in the folder divided by the sum of the length of this intersection and the social actors in the current document. The index is used comparatively with all the documents in the cluster; documents with low index of relativity are not likely to belong to the cluster.

Two different types of output in csv format are generated: 1. The validity.csv file that lists all missing cases (Fig. #); 2. The validity_freq.csv file with five rows of each type of error (social actors and NER person, organization, date, and location) and three columns for the frequency of compilations with errors in the category, a list of compilation filenames for the row type of error, and a list of source filenames for row type of error (Fig. #).

Figure # – validity.csv file output listing all cases of compilation error by error type (missing social actors and missing NER values)

Figure # – validity_freq.csv file output of frequencies of compilation errors by error type (missing social actors and missing NER values)

The data in the validity_freq.csv file are used to display in Excel column and pie charts; the two exported lists of compilation and source filenames for each type of error (social actors and NER values) are displayed when hovering over each column (see Fig. # and #).

Figure # – Column chart of frequencies of compilation errors by error type (missing social actors and missing NER values)

Figure # – Pie chart of frequencies of compilation errors by error type (missing social actors and missing NER values)

Input

The tool requires in input the folder with the documents in plain text format to be evaluated, the folder where it can save the created index and the output file name.

Output

The output is a text file containing the candidate documents for each set of duplicates and the list of its duplicates. The features extracted by Lucene and used to compare the documents are also listed as well as the similarity percentage.

[i] We used Franzosi"s PC-ACE program as described in Franzosi (2010) and available at www.pc-ace.com.