Conference Call Code Description

February 11, 2021

Abstract

This file gives descriptions of the conference call codes and the locations of raw data.

Contents

1 Data Location

The processed conference call data ready to use are in "/project/kh_mercury_1/CallCsv" and "/project/kh_mercury_1/csv". The matched conference calls with GVkey information are in the csv files starting with "CC_List". There are 20 files by year and a total file. The gvkey files could be linked to the main conference call text by using report id as an identifier.

To improve the speed of any text identification process, we automatically separate the csv files into 50 groups. These groups are stored in "/project/kh_mercury_1/WorkTemp" from csv1 to csv50. The output folder is "/project/kh_mercury_1/CriCount" with also 50 sub folders for each group (you could establish an output folder with similar structure).

2 Code Description

There are two Python codes related to the main process of identifying keywords, and an auxiliary Python code to help generate the output file. In this process, we also used a STATA do file to help us drop duplicates and make other clean. The code files are

keyword_ident_1.py:

The input of the code is the input path, the output path and the txt file with all the keywords. Note as mentioned above, the raw csvs and the output should be structured by 50 groups (you may want to increase or decrease the group). The input path and output path specify the group number, and are passed into the code through submit script file (.sh). The output of this code would be an aggregate file for each group recording the report number with identified keywords.

keyword_ident_2.py:

The input is the aggregate files generated in the first step for every group, and the csv files path. The function of this code is to extract the paragraph from every conference call that contains a specific keyword (which gives one observation we want to capture). The code will extract the csv file where the conference call is and get needed paragraph from the call script. The output file would be a excel file with all the paragraphs.

mergeclean.do:

This auxiliary code takes the output of the identification part, and first do some clean for misidentification of several keywords (abbreviation) including occ, capm, etc. Cases are that these are identified also when they appear in another complete word as adjacent letters, for example, irrational for "irr". The code also merges the gvkey file with the output file by using report number. Then the code drops duplicates in paragraphs based on the following rules:

1. If the paragraph has more than one keyword, and one of them is "interest rate" or "cost of debt", drop the observation with "interest rate" or "cost of debt".

2. If not, do a random drop and keep only one observation for the same paragraph.

The output file is an excel with paragraphs and gvkey information, sorted by keywords and other characteristics.

makebold.py:

This simple functional code serves to make all the keywords inside a paragraph bold for manual identification. Also, it helps to separate the single file into small files with 500 observations each.