READ ME FILE

Title: Firms' Price-setting Behaviour: Insights from Earnings Calls

Authors: Callan Windsor and Max Zang

Description

This 'read me' file contains general instructions on how to replicate the results presented in RDP 2023-06, while preserving our contractual obligations with Refinitiv to only share, distribute and redistribute insubstantial portions of information and/or derived data in a non-systematic manner.

Coding languages

- Python: If you do not have Python installed, a fully working free scientific distribution, which includes all of the necessary packages, can be found here: https://www.anaconda.com/download/. The code has been written in Python 3.7.
- StataMP 16 (64-bit)

Questions, comments and bug reports can be sent to windsorc and zangm at domain rba.gov.au.

Quick guide

- Download the zipped file 'rdp-2023-06-supplementary-information'.
- Unpack into desired directory.

Folder contents

- 'rdp-2023-06-read-me.pdf' is this read me file.
- 'rdp-2023-06-graph-data.xlsx' provides the data used to plot figures in the main paper in an excel format. All of the graph data is publicly available, with the exception of the NAB data in Figure 7 due to third party provider restrictions.
- 'dictionary.xlsx' provides the list of keywords and their associated qualifiers. These keywords underpin the construction of the 10 input cost indices. The file also includes the list of keywords and the associated qualifiers for the consumer demand and final price indices. For each index there are two tabs, one containing the list of negative sentiment qualifiers and the other containing the list of positive sentiment qualifiers.
- *FirmLevelIndexConstruction* is a folder containing all of the code necessary to construct our firm-level indices from the earnings call transcripts. See below for details.
- *RegressionAnalysis* is a folder containing all of the code necessary to replicate the Granger causality tests reported in the paper and the regression analysis. See below for details.

1. Sub folder: *FirmLevelIndexConstruction*

- 'analytical_database.xlsx' provides the transcript level keyword counts for each dictionary topic and the transcript level paragraph counts per each topic as classified by the zero-shot model.
- *events* is a folder containing metadata for earnings call events held by ASX-listed companies. This metadata file identifies the list of transcripts to be downloaded for analysis. It also contains necessary information for index construction, such as event time and sector classifications.

- *Keywords* is a folder containing keyword files formatted specifically to be read by 'RefinitivTranscriptAnalyser.py' for keyword counting by topic.
- *Listings* is a folder containing a csv file that records a list of ASX-listed companies. The list can be generated using the Refinitiv API by running functions in 'RefinitivListingFetcher.py', which also calls the csv file 'gic_label_map'.
- *Transcripts* is a folder containing transcript files for earnings call events. Each file is named by its 'Transcript Id' that identifies the transcript in association with an earnings call event. Files in this folder are called sequentially by 'RefinitivTranscriptAnalyser.py', and 'zero-shot-classifier.ipynb' in the process of building dictionary and zero-shot indices.
- *preds_pos*, *preds_neg* are folders containing results of zero-shot topic predictions of transcripts.
- All other files are either regular Python files (.py) or interactive Python notebook files (.ipynb). These files are described below. To easily view these files, .html versions are available in the folder *rdp_python_notebooks_html*.

Python files

Database construction

The following .py files are necessary to extract the earnings call transcripts by sending requests to the Refinitiv Street Events Application Programming Interfaces (APIs). Our access to these APIs are governed by a commercial agreement with Refinitiv.

- 'RefinitivConnection.py': authenticates credentials and establishes connection to the Refinitiv API, generates access token.
- 'RefinitivEarningsFetcher.py': searches for events by company identifiers (ASX codes), generates a table containing earnings call event ids and associated metadata. Among the metadata are earnings call transcript IDs, which are identifiers for downloading transcripts.
- 'RefinitivEarningsParser.py': functions for reading transcripts (in xml format). Functions for parsing transcripts into sentences.
- 'RefinitivListingFetcher.py': searching and downloading list of ASX listed companies.
- 'RefinitivResultsFetcher.py': functions to search and download filings/reports such as annual reports. Not related to accessing earnings calls.
- 'RefinitivTranscriptAnalyser.py': counts keywords given keyword files (csv files containing topic keyword and qualifying keyword pairs). Two counting functions included, counting of aggregate topics and counting of sub-topics.
- 'RefinitivUtils.py': miscellaneous helper functions.

Building sentiment indices

Sentiment indices can be built by executing the following notebooks in the sequence listed below.

- 'auto_transcript_update.ipynb': this notebook discovers and downloads earnings call transcripts. It searches for earnings call events held by ASX-listed companies in a given period and saves the events' metadata as a csv file in the *events* folder. It downloads transcripts of these earnings call events and saves the transcript files inside the *Transcripts* folder.

- ‘zero-shot-classifier.ipynb’: this notebook runs zero-shot classification over transcripts and saves classification outcomes as csv files.
- ‘analytical_database.ipynb’: this notebook creates transcript-level topic counts and saves the result as a database table. For dictionary keyword topics, the database records the number of topic keyword matches for each instance of transcript document. For zero-shot topics, the database records the number of paragraphs predicted to be related to a topic per instance of transcript document.
- ‘index_builder.ipynb’: this notebook builds price, cost, demand and other indices from ‘analytical_database.xlsx’.

2 Sub folder: *RegressionAnalysis*

- ‘GrangerCausality.ipynb’: this Python notebook conducts the Granger causality tests reported in the paper according to the Toda-Yamamoto procedure. All of the code is documented in Markdown cells.
- ‘GrangerCausality.html’: a .html version of the Python notebook above for easy viewing.
- ‘RegressionAnalysis.do’: this Stata .do file contains all of the code to replicate the panel data regression analysis reported in the paper. All the code is documented.

11 September 2023