1 FINANCE 751 Technical Note

This technical note informs software installation, transcript extraction, implementation and comparison methodologies to ascertain measures of corporate culture in NZX50 companies, and six Australian commercial banks, using 258 earnings call transcripts. This note describes the steps taken to implement Option-2, replicating the corporate culture results. Additionally, this technical note is for a MacOS operating system and assumes basic proficiency in package management and Python programming.

1.1 Installation

This section informs the installation of required software to facilitate analysis.

1. Install several software packages to run the StanfordCoreNLP to measure corporate culture from text files and develop transcript processing code. Anaconda is a distribution of the Python programming language, simplifying package management to develop code in the Python language. Microsoft Visual Studio Code is an integrated development environment, suitable for application building. The combination of both Anaconda and Microsoft Visual Studio Code enable programming environments to process the transcripts.
2. Secondly, clone the remote repository implementing the method described by Li et al., (2021) to your local directory. Read the instructions carefully for correct installation. In particular, change the **os.environ ["CORENLP_HOME"]** variable in **global_options.py** file to the installation location of the **stanford-corenlp-full-2018-10-05** directory on your local device. Install the required python packages excluded from Anaconda using the pip package and requirements.txt with the **pip install requirement.txt** command in the terminal. Some packages require specific versions. If you need to revert to a previous version, use the terminal command **pip install PackageName==Version** to revert to a previous version.
3. Test the correct installation of the StanfordCoreNLP using the document text files from remote repository by following the ReadMe instructions. Progress to transcript extraction after the successful execution of the StanfordCoreNLP. Otherwise, review the above installation process before progressing.

1.2 Transcript Extraction

This section informs extracting earnings call transcripts from Capital IQ.

1. Review the firm_id column in the 1.firm_score.xlsx sheet from Option-1 to identify the companies related to the 258 earnings call transcripts.
2. Navigate to Capital IQ, selecting the companies tab, followed by the transcripts link.
3. In the search criteria company search bar, type in and select each of the unique companies listed in the firm_id column mentioned above. The selected entities will update to list sixteen companies as Telecom Corp of New Zealand Ltd changed rebranded to Spark new Zealand Limited.
4. Change the time frame from 01/01/2009 to 01/10/2021 to ensure you include all 258 transcripts listed in the 1.firm_score.xlsx spreadsheet and select search in middle-right of the webpage.
5. Select all transcripts on the page by ticking the top tick box middle-left of the webpage.
6. Click the options dropbox middle-left of the webpage, and select Download in .Zip file to download all selected documents into a .Zip file.
7. Scroll to the bottom of the page to select the next subset of transcripts.
8. Repeat steps five through seven to downloaded all transcripts in .Zip files.
9. Create a new local directory titled 'transcripts', unzip all .Zip files, moving all transcripts to this newly created directory.
10. Review the transcripts in the 'transcripts' directory. The filenames align with the filename column in the 1.firm_score.xlsx spreadsheet. There will be multiple transcripts with the same name e.g., Air New Zealand Limited - ShareholderAnalyst Call.pdf. Consult filename and calltime columns in 1.firm_score.xlsx to identify the correct transcripts according to date e.g., 201510 is October 2015 , deleting the incorrect duplicates. After, the subset of 258 transcripts will exist amongst the full set in the transcript directory.

1.3 Implementation

This section highlights the code to process earnings call transcripts, execute the StanfordCoreNLP and compare the results. The implementation was partitioned into three Python functions within the finance-751-cmcd398.py script (1.5.3). This section provides a high level overview of the code with further details described in the code comments. Transcripts have a common structure. The first three pages are front-matter. The last page is the legal disclaimer. Some transcripts don't have Q&A sections while others have multiple. The transcripts without Q&A sections are isolated and excluded during processing. Transcripts with multiple Q&A sections are manually condensed prior to processing but excluded during comparison.

1.3.1 Variables

The definition of several variables and arrays take place prior to implementation.

1. Set strings describing the relative paths for the 1.firm_score.xlsx file, transcript directory, selected transcript directory to move 258 transcripts of interest, transcript directory for processed transcripts after removing Q&A sections, documents.txt file, documents_ids.txt, and processed text directory.
2. Review each transcript in the 1.firm_score.xlsx filename column to record the page number for the first page of the Q&A section, appending each value to the end of an array. If no Q&A section exists, record a value of 4. **The preservation of order is imperative with the position of the page number matching the position of the filename in the filename list from 1.firm_score.xlsx**.
3. Set an array listing the set of company ids from the 1.firm_score.xlsx spreadsheet aligning with an array listing the cumulative position of the final transcript corresponding to the company id. For example, Air New Zealand (ANZ) has 11 transcripts. Auckland International Airport (AIA) has 13 transcripts. Therefore, ANZ and AIA have values of 11 and 24, respectively, in the cumulative position array.
4. Set strings describing the relative paths for output files, results spreadsheet, and firm scores outputs from the StanfordCoreNLP.
5. Set binary variables (TRUE or FALSE) to control the execution of the below functions.

1.3.2 Prepare_documents.py

This function isolates the Q&A sections of each transcript, converts each transcript to a line in a text file, and returns the document text file and identification. The following sequence of functions are nested within, called on in the order below.

1. **get_transcripts** extracts a list of filenames from the 1.firm_score.xlsx spreadsheet, transferring transcripts of interest to the transcripts selected directory.
2. **remove_transcript_metadata** deploys the pdfrw package to extract each page of the Q&A section per transcript, using the array denoting the starting page number for the Q&A section, creating a processed transcript stored in the transcripts processed directory.
3. **create_ids** creates various forms of identification in data frames for comparison while excluding transcripts without Q&A sections.
4. **create_documents_text** deploys the pdfminer package to convert each processed transcript into a single line of text, appending each line to the document.txt file to use as an input for the StanfordCoreNLP.

1.3.3 Perform_stanford_nlp.py

This function executes each one of the five Python functions integral to StanfordCoreNLP in the following order. The provision of two separate dictionaries (NZD/AUS and US) informs analysis.

1. **parse.py** to parse the raw documents.
2. **clean_and_train.py** to clean, remove stopwords, and named entities in the parsed documents text file.
3. **create_dict.py** to create the expanded dictionary.
4. **score.py** to score the document. This implementation uses the TF-IDF weights used in the article.
5. **aggregate_firms.py** to aggregate the scores to the firm-time level.

Complete steps one, two and three. Next, replace the expanded_dictionary.csv in the dict directory with the AUS/NZD trained dictionary. It is possible to manually edit these dictionaries in attempts to improve scores. However, the provided dictionaries trained to ascertain the original scores. Therefore, the provided dictionaries were left unchanged in replicating scores. Next, Run score.py and aggregate_firms.py, saving the scores_TFIDF.csv as an xlsx file to the comparisons directory. Repeat steps four and five with the US dictionary.

1.3.4 Compare_results.py

This function combines a formatted 1.firm_scores.xlsx document with the TF-IDF output scores from perform_stanford_nlp.py by merging data frames on document identification in order to make comparisons. Compare_results.py must be repeated for both dictionaries. After, combine both comparison spreadsheets to compare results from both sets of dictionaries, deleting duplicate values.

1.4 Comparison

This section compares our replication of the measures for corporate culture across the five values (Innovation, Integrity, Quality, Respect, Teamwork) using NZ/AUS and US dictionaries. We acknowledge the provided scores have slightly shorter document lengths, likely from different pdf to text conversion methodologies. Our analysis detected a few abnormalities in the aforementioned subset of transcripts, omitting the majority of Q&A sections (1.5.1), in addition to a subset of transcripts not including Q&A sections but trained on presentation sections. The author's emphasize the presentation sections in transcripts are likely not a true reflection of company culture as edited by corporate lawyers and PR personal. Subsequently, we exclude these transactions during processing.

1.4.1 Accuracy Measures

Absolute and percentage differences between our replication and the provided results are displayed in the 751-comparison.xlsx workbook. However, we utilize the following equations to measure the accuracy of our replication across companies, values, and total results.

$$\text{Individual} = 1 - \frac{\sum_i |\text{New}_{i,j,k} - \text{Old}_{i,j,k}|}{\sum_i \text{Old}_{i,j,k}} \forall j,k \quad (1)$$

$$\text{Total} = 1 - \frac{\sum_i \sum_j \sum_k |\text{New}_{i,j,k} - \text{Old}_{i,j,k}|}{\sum_i \sum_j \sum_k \text{Old}_{i,j,k}} \quad (2)$$

$$\text{Company} = 1 - \frac{\sum_i \sum_k |\text{New}_{i,j,k} - \text{Old}_{i,j,k}|}{\sum_i \sum_k \text{Old}_{i,j,k}} \forall j \quad (3)$$

$$\text{Value} = 1 - \frac{\sum_i \sum_j |\text{New}_{i,j,k} - \text{Old}_{i,j,k}|}{\sum_i \sum_j \text{Old}_{i,j,k}} \forall k \quad (4)$$

$$i \epsilon \{1, ..., N\} \quad (5)$$

$$j \epsilon \{\text{Air New Zealand,...,Westpac Banking Corporation}\} \quad (6)$$

$$k \epsilon \{\text{Innovation,Integrity, Quality, Respect, Teamwork}\} \quad (7)$$

1.4.2 Results

Individual, **Company**, **Value**, and **Total** measure the accuracy of our replication for a specific company and value, company across all values, value across all companies, and across all values and companies respectively. The accuracy results are displayed in a matrix (1.5.2). There are a few abnormalities. The value Teamwork for Goodman Property Trust is NA as both values in the original results are zero. Our replication for Teamwork using the NZD/AUS dictionary, and Respect using the US dictionary, deviate relatively from provided figures in our replication. The later driven by material differences in Infratil's replication (-89%) and 80% accuracy for Westpac Banking Corporation. The remaining results from the Respect value using the US dictionary are above $\tilde{8}0\%$. However, all Teamwork results using the NZD/AUS dictionary are above 83%, not raising cause for concern. The **Company** level of accuracy is above 90% for all Company IDs. Each **Value** level of accuracy is above 90% except for the Respect value measured by the US dictionary ($\tilde{8}0\%$). Finally, the **Total** level of accuracy is 93%. Discrepancies may be caused by small differences in documents lengths, or abnormalities when parsing documents using StanfordCoreNLP. In summary, our results are highly accurate and satisfactory across Company IDs and Values, providing supporting evidence our replication is successful.

References

Li, K., Mai, F., Shen, R., & Yan, X. (2021). Measuring corporate culture using machine learning. *The Review of Financial Studies*, *34*(7), 3265–3315.

1.5 Appendix

1.5.1 Transcripts with Multiple Q&A Sections

The following transcripts have multiple Q&A sections. The sections are consolidated into one section by deleting the presentation material in between the Q&A sections. However, they are excluded from comparison calculation as Helen only uses the last Q&A section. We took the perspective the last section alone does not proxy for the entire Q&A sections in the transcript. Therefore, not suitable for measuring corporate culture given the document lengths.

- Australia and New Zealand Banking Group Limited - ShareholderAnalyst Call.pdf,
- Bank of Queensland Ltd. - ShareholderAnalyst Call.pdf
- Commonwealth Bank of Australia - ShareholderAnalyst Call.pdf
- Infratil Limited - AnalystInvestor Day.pdf
- Infratil Ltd. - AnalystInvestor Day.pdf
- National Australia Bank Limited - ShareholderAnalyst Call.pdf

1.5.2 Result Matrix

Firm	Document Length	Innovation (ANZ)	Integrity (ANZ)	Quality (ANZ)	Respect (ANZ)	Teamwork (ANZ)	Innovation (US)	Integrity (US)	Quality (US)	Respect (US)	Teamwork (US)	**Company**
Air New Zealand Limited	96%	95%	91%	97%	96%	92%	93%	94%	95%	90%	93%	**97%**
Auckland International Airport Limited	96%	95%	94%	95%	96%	87%	94%	91%	93%	92%	93%	**98%**
Australia New Zealand Banking Group Limited	96%	93%	94%	93%	94%	93%	94%	90%	94%	84%	95%	**96%**
Bank of Queensland Limited	96%	94%	93%	95%	93%	83%	95%	91%	93%	93%	89%	**97%**
Bendigo and Adelaide Bank Limited	96%	93%	95%	94%	94%	94%	94%	94%	94%	89%	95%	**97%**
Commonwealth Bank of Australia	96%	92%	92%	94%	92%	93%	93%	92%	92%	88%	93%	**96%**
Contact Energy Ltd	94%	90%	93%	93%	94%	93%	91%	93%	91%	88%	91%	**94%**
Fisher Paykel Healthcare Corporation Limited	95%	93%	92%	94%	92%	84%	94%	92%	93%	84%	93%	**97%**
Fletcher Building Ltd	96%	93%	94%	95%	94%	96%	92%	93%	94%	95%	95%	**96%**
Goodman Property Trust	96%	93%	91%	93%	97%	95%	94%	90%	90%	89%	#N/A	**97%**
Infratil Limited	95%	92%	94%	93%	94%	95%	94%	90%	92%	-89%	93%	**94%**
Kiwi Income Property Trust	95%	96%	95%	95%	81%	95%	96%	87%	93%	96%	95%	**98%**
National Australia Bank Limited	96%	94%	94%	95%	94%	94%	94%	93%	94%	89%	94%	**97%**
Spark New Zealand Limited	95%	94%	96%	93%	96%	88%	94%	96%	92%	89%	95%	**96%**
Telecom Corp of New Zealand Ltd	97%	95%	92%	94%	96%	84%	93%	88%	94%	85%	95%	**97%**
Vector Limited	94%	92%	93%	92%	92%	90%	91%	92%	92%	87%	92%	**95%**
Westpac Banking Corporation	96%	96%	94%	94%	94%	95%	94%	91%	94%	79%	93%	**98%**
Value	**96%**	**94%**	**93%**	**94%**	**94%**	**91%**	**94%**	**92%**	**93%**	**80%**	**93%**	**93%**

Figure 1: Results Matrix

1.5.3 Python

```
# Descriptions
# This script/function implements the StanfordNLP to score corporate culture, replicating the production of inputs in Option 1 as outputs
# Inputs for Option 1 include:
# 1.  Firm_score.xlsx contains five scores estimated with two different dictionaries for all calls. Scores ended with 1 (for example, integrity1) are estimated with the dictionary trained on the 258 call transcripts included in this sample. Scores ended with 2 (for example, integrity2) are estimated with the dictionary from the original paper (Table IA3 in the Internet Appendix). Other variables include document_id (used in your coding), filename (file name used by CapitalIQ), firm_id (firm name) and call time (year and month of the call).
# 2.  Expanded_dict1.csv is the culture dictionary trained with the 258 call transcripts (the new dictionary).
# 3.  Expanded_dict2.csv is the culture dictionary from the original paper (the original dictionary).
# 4.  Word_contributin_TFIDF1.csv (Word_contributin_TFIDF2.csv) contains word contribution based on TFIDF score estimated with the new dictionary (the original dictionary).
# 5.  The Li, Mai, Shen and Yan (2021) paper and the Internet Appendix of this paper.
#
# Author: Connor McDowall
# Date: 25th August 2021

# Imports
# Transcript Processing Modules
import pandas as pd
from pathlib import Path
import shutil as sh
from pdfrw import PdfReader, PdfWriter
import pdfminer as pdfm
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfdocument import PDFDocument
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.pdfpage import PDFPage
from pdfminer.pdfparser import PDFParser
import io
import datefinder as dtf
# General Python Modules
import datetime
import functools
import logging
import sys
import math
import os
import pickle
import gensim
import itertools
from pprint import pprint
from collections import Counter, defaultdict, OrderedDict
from tqdm.auto import tqdm
from typing import Dict, List, Optional, Set
from multiprocessing import Pool
from operator import itemgetter
from tqdm import tqdm as tqdm

# StanfordNLP Specific Functions
from culture import culture_models, file_util, preprocess, culture_dictionary, preprocess_parallel
from stanfordnlp.server import CoreNLPClient
import global_options
import parse
import clean_and_train
import create_dict
import score
import aggregate_firms

# Functions
def get_transcipts(firm_score_xlsx, transcript_directory,transcript_selected):
    """Locates and isolates transcripts for processing

    Args:
        firm_score_xlsx (xlsx): Excel file containing the initial list of transcripts
        transcript_directory (str): Source of all transcripts
        transcript_selected (str): Destination for transcripts of interest

    Returns:
        transcript_list (list): List of transcript filenames
        calltimes (list): List of calltimes
    """
    # Get list of filenames
    firms_df = pd.read_excel(firm_score_xlsx)
    firms_df=firms_df.dropna()
```

```
    firms_df.columns = firms_df.iloc[0]
    firms_df = firms_df.drop(2)
    firms_df = firms_df.reset_index(drop=True)
    transcript_list = firms_df['filename'].tolist()
    # Get list of calltimes for the firm ID
    calltimes = firms_df['calltime'].tolist()
    # Copy file into selection if exists
    files_found = 0
    files_to_find = len(transcript_list)
    missing_files_list = []
    for filename in transcript_list:
        transcipt_x = Path(transcript_directory +'/'+filename)
        if transcipt_x.is_file():
            transcipt_y = Path(transcript_selected +'/'+filename)
            sh.copy(transcipt_x,transcipt_y)
            files_found = files_found + 1
        else:
            missing_files_list.append(filename)
    missing_files = files_to_find - files_found
    if missing_files > 0:
        print('You are missing the following transcripts...')
        print(missing_files_list)
    else:
        print('All transcripts found')
    return transcript_list,calltimes

def create_ids(transcript_list,qa_num, company_ids_set, company_ids_order, documents_ids_text, calltimes):
    """Creates document identification, updates transcript list to only include transcript lists
    with Question and Answer Sections, and creates dataframe to compare results.

    Args:
        transcript_list (list): List of transcript filenames
        qa_num (list): List of page numbers denoting the start of question and answer sections
        company_ids_set (list): List of company names
        company_ids_order (list): List of numbers referencing number of file relating to one company
        documents_ids_text (str): Directory to store document id list as a text file
        calltimes (list): List of calltimes

    Returns:
        updated_transcript_list (list): List of updated filenames
        updated_document_ids (list): List of updated document ids
        updated_firm_id (list): List of updated firm ids
        output_df (dataframe): Dataframe with document information
    """
    # Initial lists
    document_ids = []
    firm_id = []
    # Updated lists
    updated_document_ids = []
    updated_firm_id = []
    updated_transcript_list = []
    updated_calltimes = []
    # Assigns document id
    idx = 0
    for i in range(len(transcript_list)):
        document_ids.append(str(i + 1)+'.F')
        if i < company_ids_order[idx]:
            firm_id.append(company_ids_set[idx])
        else:
            idx = idx +1
            firm_id.append(company_ids_set[idx])
    # Updates lists to remove entries with no question and answer sections
    for j in range(len(qa_num)):
        if qa_num[j] != 4:
            updated_document_ids.append(document_ids[j])
            updated_firm_id.append(firm_id[j])
            updated_transcript_list.append(transcript_list[j])
            updated_calltimes.append(calltimes[j])
    # Creates document_id text file
    with open(documents_ids_text, "w") as file:
        # Clear the file
        file.truncate(0)
        for element in updated_document_ids:
            file.write(element + "\n")
        file.close()
    # Creates a dataframe with updated transcript list
    output_df = pd.DataFrame(list(zip(updated_document_ids, updated_transcript_list, updated_firm_id)),
                columns =['document_id', 'filename', 'firm_id'])
```

```
    # Creates id2firsm csv
    for i in range(len(updated_calltimes)):
        val = updated_calltimes[i]
        new_val = int(str(val)[:4])
        updated_calltimes[i] =  new_val
    id2firms_df = pd.DataFrame(list(zip(updated_document_ids,updated_firm_id,
    updated_calltimes)),
                columns =['document_id', 'firm_id','time'])
    print(id2firms_df.head())
    id2firms_df.to_csv('data/input/id2firms.csv')
    return updated_transcript_list, updated_document_ids, updated_firm_id, output_df

def remove_transcript_metadata(transcript_list,qa_num,transcript_selected,
    transcript_processed):
    """Removes front matter, table of contents, call participants, and copyright
    disclaimer
    to process transcripts to a format suitable for combination. This is possible as
    the
    format is consistent for all earnings call transcripts.

    Args:
        transcript_list (list): List of transcript filenames
        qa_num (list): List of page numbers denoting the start of question and answer
     sections
        transcript_selected (str): String of selected transcipt directory
        transcript_processed (str): String of processed transcript directory
    """
    # Count for
    i = 0
    # Create copy, remove pages, and move to processed directory
    for filename in transcript_list:
        # Defines pdfs
        input_pdf = Path(transcript_selected +'/'+filename)
        output_pdf = Path(transcript_processed +'/'+filename)
        # Defines objects
        reader_input = PdfReader(input_pdf)
        writer_output = PdfWriter()
        for page_x in range(len(reader_input.pages)):
            # Adds pages excluding sections prior to Q&A section and legal disclaimer
            if page_x >= qa_num[i]-1 and page_x < (len(reader_input.pages)-1):
                writer_output.addpage(reader_input.pages[page_x])
        writer_output.write(output_pdf)
        i = i + 1
    return

def create_documents_text(transcript_list,transcript_processed, text_processed,
    documents_text):
    """Creates documents.txt file for the Stanford NLP

    Args:
        transcript_list(str): List of processed transcipts
        transcript_processed (str): String of processed transcript directory
        text_processed (str): Directory to store text file
        documents_text (str): Directory for documents.txt file

    Returns:
        documents_test_list (list): Returns a list of processed transcript document
    strings
    """
    # Adapted from https://towardsdatascience.com/pdf-text-extraction-in-python-5
    b6ab9e92dd
    # Erase object contents to reset the textfile
    with open(documents_text, "r+") as file:
        file.truncate(0)
        file.close()
    # Creates empty list
    documents_test_list = []
    # Begin extracting files
    for file_name in transcript_list:
        file_pdf = Path(transcript_processed +'/'+file_name)
        file_text = io.StringIO()
        with open(file_pdf, 'rb') as in_file:
            parser = PDFParser(in_file)
            doc = PDFDocument(parser)
            rsrcmgr = PDFResourceManager()
            device = TextConverter(rsrcmgr, file_text, laparams=LAParams())
            interpreter = PDFPageInterpreter(rsrcmgr, device)
            for page in PDFPage.create_pages(doc):
                interpreter.process_page(page)
        # Extract text to and remove characters
        textname = Path(text_processed +'/output.txt')
        with open(textname, "w") as file:
            file.write(file_text.getvalue())
```

```
            file.close()
        # Print the lines
        with open(textname, "r+") as file:
            line = file.read().replace("\n", " ")
            file.truncate(0)
            file.close()
        # Write line to the documents file
        with open(documents_text, "a") as file:
            file.write(line)
            if file_name != transcript_list[-1]:
                file.write("\n")
            file.close()
        # Create list of texts and dates
        documents_test_list.append(line)
    return documents_test_list

def prepare_documents(firm_score_xlsx, transcript_directory, transcript_selected, transcript_processed, text_processed, documents_text, documents_ids_text, qa_num, company_ids_set,company_ids_order):
    """ Isolate transcripts of interest, process Q&A sections, and create document files

    Args:
        firm_score_xlsx (xlsx): Excel file containing the initial list of transcripts
        transcript_directory (str): Source of all transcripts
        transcript_selected (str): Destination for transcripts of interest
        transcript_processed (str): Directory for processed transcripts
        text_processed (str): Directory to store text file
        documents_text (str): Directory for documents.txt file
        documents_ids_text (str): Directory to store document id list as a text file
        qa_num (list): List of page numbers denoting the start of question and answer sections
        company_ids_set (list): List of company names
        company_ids_order (list): List of numbers referencing number of file relating to one company

    Returns:
        documents_test_list (list): Returns a list of processed transcript document strings
        document_ids (list): List of document ids
        firm_id (list): List of firm ids
        output_df (df): Dataframe with document information
    """
    # Prepares the documentation
    # Get list of transcripts
    transcript_list, calltimes = get_transcipts(firm_score_xlsx, transcript_directory, transcript_selected)
    # Isolates Q&A sections while removing legal disclaimers
    remove_transcript_metadata(transcript_list,qa_num,transcript_selected, transcript_processed)
    # Creates supplementary identification (Changed here to remove files without text files)
    transcript_list, document_ids, firm_id, output_df = create_ids(transcript_list, qa_num, company_ids_set, company_ids_order, documents_ids_text, calltimes)
    # Creates the documents.txt file, documents ids, firm_ids, and dataframe of outputs
    documents_test_list = create_documents_text(transcript_list,transcript_processed, text_processed, documents_text)
    # Saves csv for comparison
    dataframe_file = Path('data/input/results.csv')
    output_df.to_csv(dataframe_file)
    return documents_test_list, document_ids, firm_id, output_df

def perform_stanford_nlp():
    """Executes Stanford NLP algorithm on processed documentation via
    """
    print("Implementing Stanford NLP...")
    # Creates variables and directories in global options
    exec(open("global_options.py").read())
    # Step 1: Use `python parse.py` to use Stanford CoreNLP to parse the raw documents.
    exec(open("parse.py").read())
    # Step 2: Use `python clean_and_train.py` to clean, remove stopwords, and named entities in parsed `documents.txt`
    exec(open("clean_and_train.py").read())
    # Step 3: Use `python create_dict.py` to create the expanded dictionary.
    exec(open("create_dict.py").read())
    # Step 4: Use `python score.py` to score the documents.
    exec(open("score.py").read())
    # Step 5: Use `python aggregate_firms.py` to aggregate the scores to the firm-time level.
    exec(open("aggregate_firms.py").read())
    return
```

```
293
294 def compare_results(results,output_scores):
295     """Creates comparison excel sheets with helens results
296
297     Args:
298         results (str): Directory to the document id files
299         output_scores (str): Directory for scoring sheets
300     """
301     # Load in the results
302     output_df = pd.read_csv(results)
303     # Set directories
304     tf = 'firm_scores_TF.csv'
305     tfidf = 'firm_scores_TFIDF.csv'
306     wfidf = 'firm_scores_WFIDF.csv'
307     helen_results = 'outputs/scores/firm_score_helen.xlsx'
308     firm_scores_tf = Path(output_scores+'/'+tf)
309     firm_scores_tfidf = Path(output_scores+'/'+tfidf)
310     firm_scores_wfidf = Path(output_scores+'/'+wfidf)
311     helen_results = Path(helen_results)
312     # Read csv and excel files
313     firm_scores_tf_df = pd.read_csv(firm_scores_tf)
314     firm_scores_tfidf_df = pd.read_csv(firm_scores_tfidf)
315     firm_scores_wfidf_df = pd.read_csv(firm_scores_wfidf)
316     helen_results = pd.read_excel(helen_results)
317     # Merge results with dataframes for comparison
318     target_df = firm_scores_tfidf_df
319     user_results_df = pd.merge(output_df,  target_df,how = 'left',on = output_df.
    index)
320     comparison_df = pd.merge(user_results_df,helen_results,how = 'left',on = ['
    document_id'])
321     print('Please enter a filename')
322     filename = input()
323     # Save comparison csv
324     file_string = 'outputs/comparisons'+'/'+filename+'.xlsx'
325     comparison_df.to_excel(file_string)
326     return
327
328 # Inputs - established all the directories for the locations
329 # Inputs for processing
330 firm_score_xlsx = 'data/input/option-1/1.firm_score.xlsx'
331 transcript_directory = 'data/input/transcripts'
332 transcript_selected = 'data/raw/selected_transcripts'
333 transcript_processed = 'data/processed/processed_transcripts'
334 text_processed = 'data/processed/processed_text'
335 documents_text = 'data/input/documents.txt'
336 documents_ids_text = 'data/input/document_ids.txt'
337 # Creates array of pages numbers indicating the start of the Q&A section for each PDF
338 # Note: This is labourous but necessary. Values of 4 indicate no Q&A section in the
    document,
339 # starting at the presentation section
340 air_nz_num=[8,10,10,7,8,10,8,11,8,8,8]
341 aia_num = [4,4,12,12,12,9,10,15,11,10,10,10,10] # Changed to 4
342 anz_num =
    [14,6,10,11,11,13,11,13,11,10,11,13,13,8,7,8,7,10,11,12,13,11,12,10,11,11,13,12,11,12,8]
343 bql_num =
    [24,12,10,11,11,12,11,11,14,11,9,16,12,13,16,14,13,12,13,10,10,11,12,12,12,13,8]
344 bab_num = [4,10,10,12,11,10,10,10,14,15,10,10,10,12,10,10,12,12,15,15,9,10]
345 cba_num =
    [5,11,11,12,12,11,12,11,10,10,4,11,11,10,11,11,11,10,12,12,11,12,12,12,6,6,6,7,8]
     # Changed to 4 (29)
346 ce_num = [8,4] # Changed to 4
347 fph_num = [9,8,9,8,9,8,8,7,8,8,8]
348 fbu_num = [12,10,11,10,9,10,10,9,10,11,9]
349 gpt_num = [10,9]
350 il_num = [15,15,15,15,13,14,13,12,13,15,15,16,13]
351 kip_num = [11,10]
352 nab_num =
    [12,4,4,13,12,14,10,18,15,9,10,11,11,12,13,13,10,12,15,10,9,10,10,12,11,9,8,15,7,7,6]
     # Changed to 4 (31)
353 spk_num = [16,12,12]
354 tnz_num = [15,14,11,16,14,12,13,9,16]
355 vec_num = [9,12,9,9,10,9,9,8,12,11,10,9]
356 wpc_num =
    [12,19,12,14,14,13,13,11,12,11,11,12,11,11,16,14,12,12,12,11,11,12,10,11,7,8,7,7,7]
357 # Combines the arrays
358 qa_num = [air_nz_num,
359             aia_num,
360             anz_num,
361             bql_num,
362             bab_num,
363             cba_num,
364             ce_num,
```

```
            fph_num,
            fbu_num,
            gpt_num,
            il_num,
            kip_num,
            nab_num,
            spk_num,
            tnz_num,
            vec_num,
            wpc_num]

qa_num = air_nz_num+aia_num+anz_num+bql_num+bab_num+cba_num+ce_num+fph_num+fbu_num+gpt_num+il_num+kip_num+nab_num+spk_num+tnz_num+ vec_num + wpc_num
# Sets list for company ids
company_ids_set = ['Air New Zealand Limited','Auckland International Airport Limited','Australia New Zealand Banking Group Limited', 'Bank of Queensland Limited','Bendigo and Adelaide Bank Limited','Commonwealth Bank of Australia', 'Contact Energy Ltd','Fisher Paykel Healthcare Corporation Limited','Fletcher Building Ltd','Goodman Property Trust','Infratil Limited','Kiwi Income Property Trust','National Australia Bank Limited','Spark New Zealand Limited','Telecom Corp of New Zealand Ltd','Vector Limited','Westpac Banking Corporation']
company_ids_order = [11,24,55,82,104,133,135,146,157,159,172,174,205,208,217,229,258]
# Inputs for comparison
output_scores = 'outputs/scores'
results = 'data/input/results.csv'
output_word_contributions = 'outputs/scores/word_contributions'
firm_scores_tf = 'outputs/scores/firm_scores_TF.csv'
firm_scores_tfidf = 'outputs/scores/firm_scores_TFIDF.csv'
firm_scores_wfidf = 'outputs/scores/firm_scores_WFIDF.csv'
#########################################################
# Function Calls
# Set binary variables to control function calls
transcript_preparation = False
stanford_nlp_implementation = False
results_comparison = True
# Executes functions based on binary variables
if transcript_preparation == True:
    # Prepare the documents
    print("Preparing documents...")
    documents_test_list, document_ids, firm_id, output_df = prepare_documents(firm_score_xlsx, transcript_directory, transcript_selected, transcript_processed, text_processed, documents_text, documents_ids_text, qa_num, company_ids_set, company_ids_order)
if stanford_nlp_implementation == True:
    # Implements Stanford NLP
    perform_stanford_nlp()
if results_comparison == True:
    print('Comparing results...')
    compare_results(results,output_scores)
# Note: Australia and New Zealand Banking Group Limited - ShareholderAnalyst Call.pdf, Bank of Queensland Ltd. - ShareholderAnalyst Call.pdf
# Commonwealth Bank of Australia - ShareholderAnalyst Call.pdf, Infratil Limited - AnalystInvestor Day.pdf, Infratil Ltd. - AnalystInvestor Day.pdf
# National Australia Bank Limited - ShareholderAnalyst Call.pdf
```