template 26 juni sore

June 27, 2024

1 REMEMBER TO CHECK.

Future improvement:

Yang masuk sample dari 186 ke bawah, wajib di double-confirm dgn antconc karena no|none nya salah algoritma :(

nanti, setelah populasi muncul semua, bagi tiga kategori, outliers, first half, second half -> dimitris cuma bagi tiga rata aja

AAAAAAA SALAH ALGORITMA NYAAAAAAAAA, harusnya run through ulang dari nol

[10]:
```
pip install openai --upgrade
```

```
Requirement already satisfied: openai in c:\users\user\anaconda3\lib\site-
packages (1.35.4)
Collecting openai
  Downloading openai-1.35.5-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in
c:\users\user\anaconda3\lib\site-packages (from openai) (4.2.0)
Requirement already satisfied: distro<2,>=1.7.0 in
c:\users\user\anaconda3\lib\site-packages (from openai) (1.8.0)
Requirement already satisfied: httpx<1,>=0.23.0 in
c:\users\user\anaconda3\lib\site-packages (from openai) (0.27.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in
c:\users\user\anaconda3\lib\site-packages (from openai) (1.10.12)
Requirement already satisfied: sniffio in c:\users\user\anaconda3\lib\site-
packages (from openai) (1.3.0)
Requirement already satisfied: tqdm>4 in c:\users\user\anaconda3\lib\site-
packages (from openai) (4.65.0)
Requirement already satisfied: typing-extensions<5,>=4.7 in
c:\users\user\anaconda3\lib\site-packages (from openai) (4.9.0)
Requirement already satisfied: idna>=2.8 in c:\users\user\anaconda3\lib\site-
packages (from anyio<5,>=3.5.0->openai) (3.4)
Requirement already satisfied: certifi in c:\users\user\anaconda3\lib\site-
packages (from httpx<1,>=0.23.0->openai) (2024.2.2)
Requirement already satisfied: httpcore==1.* in
c:\users\user\anaconda3\lib\site-packages (from httpx<1,>=0.23.0->openai)
(1.0.5)
Requirement already satisfied: h11<0.15,>=0.13 in
```

```
c:\users\user\anaconda3\lib\site-packages (from
httpcore==1.*->httpx<1,>=0.23.0->openai) (0.14.0)
Requirement already satisfied: colorama in c:\users\user\anaconda3\lib\site-
packages (from tqdm>4->openai) (0.4.6)
Downloading openai-1.35.5-py3-none-any.whl (327 kB)
   ---------------------------------------- 0.0/327.5 kB ? eta -:--:--
   --- ------------------------------------ 30.7/327.5 kB 1.4 MB/s eta 0:00:01
   ---------------------------------- ----- 286.7/327.5 kB 4.5 MB/s eta 0:00:01
   ---------------------------------------- 327.5/327.5 kB 3.4 MB/s eta 0:00:00
Installing collected packages: openai
  Attempting uninstall: openai
    Found existing installation: openai 1.35.4
    Uninstalling openai-1.35.4:
      Successfully uninstalled openai-1.35.4
Successfully installed openai-1.35.5
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```python
import os
import pandas as pd
from sec_downloader import Downloader
from sec_downloader.types import RequestedFilings

def read_tickers_from_csv(csv_filename):
    df = pd.read_csv(csv_filename)
    return df.set_index('number')['ticker'].to_dict()

def validate_filing_dates(metadatas, start_year=2023, end_year=2014):
    if len(metadatas) != (start_year - end_year + 1):
        return False
    expected_years = {str(year) for year in range(end_year, start_year + 1)}
    actual_years = {metadata.report_date[:4] for metadata in metadatas}
    return expected_years == actual_years

# Initialize API connection
dl = Downloader("Afi Testing", "afiinvesting@gmail.com") #input your name and␣
 ↪your email
print("Connection successful\n")

# Load tickers from CSV
ticker_mapping = read_tickers_from_csv('numbered_from_smallest_modified.csv')␣
 ↪#create your own CSV, I've attached mine so you can just follow
print("Tickers loaded from CSV:", ticker_mapping)
print("\n")

# Input numbers and get corresponding tickers
numbers = [int(num.strip()) for num in input("Enter the number(s) corresponding␣
 ↪to the ticker(s), separated by comma: ").split(",")]
```

```
tickers = [(num, ticker_mapping.get(num)) for num in numbers]

# Check for missing tickers
missing_tickers = [num for num, ticker in tickers if not ticker]
if missing_tickers:
    print(f"Ticker(s) not found for number(s): {', '.join(map(str,␣
 ↪missing_tickers))}\n")

# Process each ticker
tickers_not_covering_years = []
for num, ticker in tickers:
    if ticker:
        metadatas = dl.get_filing_metadatas(
            RequestedFilings(ticker_or_cik=ticker, form_type="10-K", limit=10)␣
 ↪#change the form_type, read the documentation
        )

        if validate_filing_dates(metadatas):
            output_dir = f"downloaded_filings_{ticker}"
            os.makedirs(output_dir, exist_ok=True)

            for metadata in metadatas:
                html = dl.download_filing(url=metadata.primary_doc_url).decode()
                report_date = metadata.report_date
                file_name = f"{ticker}_{report_date}.txt"
                file_path = os.path.join(output_dir, file_name)

                with open(file_path, 'w', encoding='utf-8') as file:
                    file.write(html)

                print(f"Downloaded {file_name}")
            print()
        else:
            print(f"The filings for ticker {ticker} (number {num}) do not cover␣
 ↪the required range of years from 2023 to 2014.\n")
            tickers_not_covering_years.append((num, ticker))

if tickers_not_covering_years:
    print("The following tickers do not cover the required range of years from␣
 ↪2023 to 2014:")
    for num, ticker in tickers_not_covering_years:
        print(f"Ticker {ticker} (number {num})\n")

print("All filings downloaded and saved.\n")
```

```
Connection successful
```

```
Tickers loaded from CSV: {1: 'NIDB', 2: 'CRSB', 3: 'BKSC', 4: 'CARV', 5: 'VBFC',
6: 'QNTO', 7: 'PFBX', 8: 'NWPP', 9: 'IROQ', 10: 'FSRL', 11: 'AUBN', 12: 'SFBC',
13: 'UBOH', 14: 'UWHR', 15: 'FUSB', 16: 'CMTV', 17: 'HMNF', 18: 'PFLC', 19:
'CSBB', 20: 'UBFO', 21: 'FMBM', 22: 'PROV', 23: 'SBFG', 24: 'OVBC', 25: 'ASRV',
26: 'CIZN', 27: 'FKYS', 28: 'FXNC', 29: 'OPOF', 30: 'UNB', 31: 'BFIN', 32:
'FFNW', 33: 'RVSB', 34: 'SFDL', 35: 'LARK', 36: 'PLBC', 37: 'PEBK', 38: 'VABK',
39: 'NKSH', 40: 'EMYB', 41: 'QNBC', 42: 'MBCN', 43: 'EFSI', 44: 'FCCO', 45:
'FRAF', 46: 'TSBK', 47: 'OVLY', 48: 'CZWI', 49: 'FNRN', 50: 'HWBK', 51: 'FNCB',
52: 'FUNC', 53: 'BPRN', 54: 'PKBK', 55: 'ISBA', 56: 'EBMT', 57: 'EVBN', 58:
'ATLO', 59: 'CVLY', 60: 'NWFL', 61: 'PWOD', 62: 'WSBF', 63: 'TBNK', 64: 'FSFG',
65: 'LCNB', 66: 'ESSA', 67: 'ACNB', 68: 'CWBC', 69: 'CFFI', 70: 'FDBC', 71:
'CZNC', 72: 'BCML', 73: 'WNEB', 74: 'UNTY', 75: 'CHMG', 76: 'MCBC', 77: 'NRIM',
78: 'ISTR', 79: 'NBN', 80: 'FNLC', 81: 'FSBW', 82: 'CZFS', 83: 'CBAN', 84:
'ORRF', 85: 'FCBC', 86: 'FMAO', 87: 'MVBF', 88: 'HBCP', 89: 'FBIZ', 90: 'FGBI',
91: 'FRBA', 92: 'BSRR', 93: 'PFIS', 94: 'BMRC', 95: 'WTBA', 96: 'BCBP', 97:
'CIVB', 98: 'FRST', 99: 'AMBZ', 100: 'BHB', 101: 'SFST', 102: 'AROW', 103:
'FLIC', 104: 'CCBG', 105: 'HBIA', 106: 'SMBC', 107: 'EBTC', 108: 'HIFS', 109:
'HTBI', 110: 'SMBK', 111: 'THFF', 112: 'CNND', 113: 'EQBK', 114: 'FMNB', 115:
'INBK', 116: 'HTBK', 117: 'IBCP', 118: 'MPB', 119: 'FMCB', 120: 'TFIN', 121:
'MBWM', 122: 'CATC', 123: 'NFBK', 124: 'CAC', 125: 'OSBC', 126: 'FBAK', 127:
'CCNE', 128: 'CTBI', 129: 'GSBC', 130: 'SHBI', 131: 'FRBK', 132: 'GABC', 133:
'FISI', 134: 'CHCO', 135: 'TRST', 136: 'WABC', 137: 'MOFG', 138: 'PGC', 139:
'LKFN', 140: 'RBCAA', 141: 'PFBC', 142: 'HFWA', 143: 'WASH', 144: 'CASH', 145:
'HAFC', 146: 'FMBH', 147: 'CPF', 148: 'TBBK', 149: 'UVSP', 150: 'TMP', 151:
'HBNC', 152: 'FBMS', 153: 'SYBT', 154: 'SBSI', 155: 'NIC', 156: 'FFIC', 157:
'QCRH', 158: 'PFC', 159: 'SRCE', 160: 'PEBO', 161: 'HMST', 162: 'STBA', 163:
'PRK', 164: 'CNOB', 165: 'TCBK', 166: 'NBHC', 167: 'CFFN', 168: 'LOB', 169:
'BRKL', 170: 'FCF', 171: 'EGBN', 172: 'FBNC', 173: 'BUSE', 174: 'BANF', 175:
'VBTX', 176: 'BHLB', 177: 'FBK', 178: 'FFIN', 179: 'NBTB', 180: 'OCFC', 181:
'DCOM', 182: 'SASR', 183: 'PFS', 184: 'NWBI', 185: 'EFSC', 186: 'SBCF', 187:
'IBOC', 188: 'CBU', 189: 'BANR', 190: 'CVBF', 191: 'SFBS', 192: 'HTH', 193:
'TOWN', 194: 'RNST', 195: 'FFBC', 196: 'WSBC', 197: 'FRME', 198: 'TRMK', 199:
'PPBI', 200: 'IBTX', 201: 'HOPE', 202: 'INDB', 203: 'HTLF', 204: 'AX', 205:
'WSFS', 206: 'AUB', 207: 'CUBI', 208: 'WAFD', 209: 'HOMB', 210: 'CATY', 211:
'BOH', 212: 'ABCB', 213: 'UCBI', 214: 'SFNC', 215: 'FULT', 216: 'GBCI', 217:
'TCBI', 218: 'UBSI', 219: 'FIBK', 220: 'CBSH', 221: 'HWC', 222: 'BKU', 223:
'BANC', 224: 'PB', 225: 'ASB', 226: 'UMBF', 227: 'SSB', 228: 'FNB', 229: 'PNFP',
230: 'CADE', 231: 'ONB', 232: 'BOKF', 233: 'CFR', 234: 'COLB', 235: 'WTFC', 236:
'SNV', 237: 'VLY', 238: 'EWBC', 239: 'WAL', 240: 'WBS', 241: 'FHN', 242: 'CMA',
243: 'SBNY', 244: 'NYCB', 245: 'RF', 246: 'KEY', 247: 'HBAN', 248: 'MTB', 249:
'SIVBQ', 250: 'FRCB', 251: 'FCNCA', 252: 'FITB', 253: 'CFG', 254: 'TFC', 255:
'PNC', 256: 'USB', 257: 'WFC', 258: 'C', 259: 'BAC', 260: 'JPM'}


Enter the number(s) corresponding to the ticker(s), separated by comma:  260

Downloaded JPM_2023-12-31.txt
Downloaded JPM_2022-12-31.txt
```

```
Downloaded JPM_2021-12-31.txt
Downloaded JPM_2020-12-31.txt
Downloaded JPM_2019-12-31.txt
Downloaded JPM_2018-12-31.txt
Downloaded JPM_2017-12-31.txt
Downloaded JPM_2016-12-31.txt
Downloaded JPM_2015-12-31.txt
Downloaded JPM_2014-12-31.txt

All filings downloaded and saved.
```

[1]:
```python
import os
import re
import pandas as pd
import openai
import json

# Step 3: Set up your API key
os.environ['OPENAI_API_KEY'] =␣
 ↪'sk-proj-BZ4y1s6uJOHWlzuXMFZ9T3BlbkFJiDofcONiyyflWjbTMN9n'
openai.api_key = os.getenv('OPENAI_API_KEY')

# Function to read tickers from CSV
def read_tickers_from_csv(csv_filename):
    df = pd.read_csv(csv_filename)
    return df.set_index('number')['ticker'].to_dict()

# Read tickers from the CSV file
ticker_mapping = read_tickers_from_csv('numbered_from_smallest_modified.csv') ␣
 ↪# Adjust the path as necessary
print("Tickers loaded from CSV:", ticker_mapping)

# Input the number corresponding to the ticker
numbers = input("Enter the number(s) corresponding to the ticker(s), separated␣
 ↪by comma: ")
numbers = [int(num.strip()) for num in numbers.split(",")]

# Get the ticker corresponding to the input number
tickers = [ticker_mapping.get(num) for num in numbers]

# Check if the ticker exists
for ticker in tickers:
    if not ticker:
        print("Ticker not found.")
    else:
        # Define Fintech Keywords
```

```
        fintech_keywords = [
            "Technology", "Digital Banking", "Network", "Internet Banking",
            "Online Services", "FinTech", "AI", "Blockchain", "E-payment",␣
↪"Mobile Banks"
        ]

        # Define Negative Patterns
        negative_patterns = re.compile(r'\b(no|none)\b', re.IGNORECASE)

        print("Step 1: Keywords and patterns defined.")

        def load_text(file_path):
            with open(file_path, 'r', encoding='utf-8') as file:
                return file.read()

        print("Step 2: Function to load text defined.")

        # Function to extract sentences containing fintech-related keywords
        def extract_sentences_with_keywords(text, keywords):
            sentences = re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s',␣
↪text)
            keyword_sentences = [sentence.strip() for sentence in sentences if␣
↪any(keyword in sentence for keyword in keywords)]
            return keyword_sentences

        print("Step 3: Function to extract sentences with fintech keywords␣
↪defined.")

        # Function to classify sentences using OpenAI GPT-4 model
        def classify_sentences_with_gpt(sentences):
            results = []
            for sentence in sentences:
                prompt = f"""
                Please classify the following sentence based on its relevance␣
↪to fintech.
                Sentence: {sentence}
                Respond in json format with three keys "classification",␣
↪"probability score" and "Reasons".
                """
                response = openai.ChatCompletion.create(
                    model="gpt-3.5",  # Adjust model name if using a different␣
↪model
                    prompt=prompt,
                    max_tokens=150,
                    n=1,
                    stop=None,
```

```
                    temperature=0.7
                )
                response_text = response['choices'][0]['text'].strip()
                result = json.loads(response_text)

                result_dict = {
                    'sentence': sentence,
                    'classification': result['classification'],
                    'probability score': result['probability score'],
                    'Reasons': result['Reasons']
                }
                results.append(result_dict)

            return results

        print("Step 4: Function to classify sentences with ChatGPT defined.")

        # Function to clean negative occurrences
        def clean_negative_occurrences(text):
            return re.sub(negative_patterns, '', text)

        print("Step 5: Function to clean negative occurrences defined.")

        # Processing files and capturing context
        def process_files(directory, keywords):
            results = []
            print("Step 6: Processing files and capturing context.")

            # Check if directory exists
            if not os.path.exists(directory):
                print(f"Error: The directory '{directory}' does not exist.")
                return results

            for filename in os.listdir(directory):
                if filename.endswith(".txt"):
                    file_path = os.path.join(directory, filename)
                    print(f"Processing file: {file_path}")
                    text = load_text(file_path)
                    text_cleaned = clean_negative_occurrences(text)
                    sentences_with_keywords =␣
↪extract_sentences_with_keywords(text_cleaned, keywords)

                    # Classify sentences with OpenAI GPT-4
                    if sentences_with_keywords:
                        sentence_classifications =␣
↪classify_sentences_with_gpt(sentences_with_keywords)
```

```
                        # Store the results
                        base_filename = os.path.splitext(filename)[0]
                        current_ticker, report_date = base_filename.split('_')
                        for classification in sentence_classifications:
                            result = {
                                'Company': current_ticker,
                                'Report Date': report_date,
                                'Sentence': classification['sentence'],
                                'Classification':␣
↪classification['classification'],
                                'Probability Score':␣
↪classification['probability score'],
                                'Reasons': classification['Reasons']
                            }
                            results.append(result)
                        print("Processing completed.")

            return results

        print("Step 7: Function to process files and capturing context defined.
↪")

        # Function to save results to CSV
        def save_to_csv(results, ticker, output_filename):
            df = pd.DataFrame(results)
            df.to_csv(output_filename, index=False)
            print(f"Data successfully saved to {output_filename}")

        print("Step 8: Function to save results to CSV defined.")
        print("\n")

        # Set the directory dynamically based on the ticker
        directory = f"downloaded_filings_{ticker}"

        # Process the text files
        results = process_files(directory, fintech_keywords)

        if results:
            # Save the results to a CSV file with a dynamic name based on the␣
↪ticker symbol
            csv_filename = f"fintech_analysis_{ticker}.csv"
            save_to_csv(results, ticker, csv_filename)
```

```
Tickers loaded from CSV: {1: 'NIDB', 2: 'CRSB', 3: 'BKSC', 4: 'CARV', 5: 'VBFC',
6: 'QNTO', 7: 'PFBX', 8: 'NWPP', 9: 'IROQ', 10: 'FSRL', 11: 'AUBN', 12: 'SFBC',
13: 'UBOH', 14: 'UWHR', 15: 'FUSB', 16: 'CMTV', 17: 'HMNF', 18: 'PFLC', 19:
'CSBB', 20: 'UBFO', 21: 'FMBM', 22: 'PROV', 23: 'SBFG', 24: 'OVBC', 25: 'ASRV',
```

```
26: 'CIZN', 27: 'FKYS', 28: 'FXNC', 29: 'OPOF', 30: 'UNB', 31: 'BFIN', 32:
'FFNW', 33: 'RVSB', 34: 'SFDL', 35: 'LARK', 36: 'PLBC', 37: 'PEBK', 38: 'VABK',
39: 'NKSH', 40: 'EMYB', 41: 'QNBC', 42: 'MBCN', 43: 'EFSI', 44: 'FCCO', 45:
'FRAF', 46: 'TSBK', 47: 'OVLY', 48: 'CZWI', 49: 'FNRN', 50: 'HWBK', 51: 'FNCB',
52: 'FUNC', 53: 'BPRN', 54: 'PKBK', 55: 'ISBA', 56: 'EBMT', 57: 'EVBN', 58:
'ATLO', 59: 'CVLY', 60: 'NWFL', 61: 'PWOD', 62: 'WSBF', 63: 'TBNK', 64: 'FSFG',
65: 'LCNB', 66: 'ESSA', 67: 'ACNB', 68: 'CWBC', 69: 'CFFI', 70: 'FDBC', 71:
'CZNC', 72: 'BCML', 73: 'WNEB', 74: 'UNTY', 75: 'CHMG', 76: 'MCBC', 77: 'NRIM',
78: 'ISTR', 79: 'NBN', 80: 'FNLC', 81: 'FSBW', 82: 'CZFS', 83: 'CBAN', 84:
'ORRF', 85: 'FCBC', 86: 'FMAO', 87: 'MVBF', 88: 'HBCP', 89: 'FBIZ', 90: 'FGBI',
91: 'FRBA', 92: 'BSRR', 93: 'PFIS', 94: 'BMRC', 95: 'WTBA', 96: 'BCBP', 97:
'CIVB', 98: 'FRST', 99: 'AMBZ', 100: 'BHB', 101: 'SFST', 102: 'AROW', 103:
'FLIC', 104: 'CCBG', 105: 'HBIA', 106: 'SMBC', 107: 'EBTC', 108: 'HIFS', 109:
'HTBI', 110: 'SMBK', 111: 'THFF', 112: 'CNND', 113: 'EQBK', 114: 'FMNB', 115:
'INBK', 116: 'HTBK', 117: 'IBCP', 118: 'MPB', 119: 'FMCB', 120: 'TFIN', 121:
'MBWM', 122: 'CATC', 123: 'NFBK', 124: 'CAC', 125: 'OSBC', 126: 'FBAK', 127:
'CCNE', 128: 'CTBI', 129: 'GSBC', 130: 'SHBI', 131: 'FRBK', 132: 'GABC', 133:
'FISI', 134: 'CHCO', 135: 'TRST', 136: 'WABC', 137: 'MOFG', 138: 'PGC', 139:
'LKFN', 140: 'RBCAA', 141: 'PFBC', 142: 'HFWA', 143: 'WASH', 144: 'CASH', 145:
'HAFC', 146: 'FMBH', 147: 'CPF', 148: 'TBBK', 149: 'UVSP', 150: 'TMP', 151:
'HBNC', 152: 'FBMS', 153: 'SYBT', 154: 'SBSI', 155: 'NIC', 156: 'FFIC', 157:
'QCRH', 158: 'PFC', 159: 'SRCE', 160: 'PEBO', 161: 'HMST', 162: 'STBA', 163:
'PRK', 164: 'CNOB', 165: 'TCBK', 166: 'NBHC', 167: 'CFFN', 168: 'LOB', 169:
'BRKL', 170: 'FCF', 171: 'EGBN', 172: 'FBNC', 173: 'BUSE', 174: 'BANF', 175:
'VBTX', 176: 'BHLB', 177: 'FBK', 178: 'FFIN', 179: 'NBTB', 180: 'OCFC', 181:
'DCOM', 182: 'SASR', 183: 'PFS', 184: 'NWBI', 185: 'EFSC', 186: 'SBCF', 187:
'IBOC', 188: 'CBU', 189: 'BANR', 190: 'CVBF', 191: 'SFBS', 192: 'HTH', 193:
'TOWN', 194: 'RNST', 195: 'FFBC', 196: 'WSBC', 197: 'FRME', 198: 'TRMK', 199:
'PPBI', 200: 'IBTX', 201: 'HOPE', 202: 'INDB', 203: 'HTLF', 204: 'AX', 205:
'WSFS', 206: 'AUB', 207: 'CUBI', 208: 'WAFD', 209: 'HOMB', 210: 'CATY', 211:
'BOH', 212: 'ABCB', 213: 'UCBI', 214: 'SFNC', 215: 'FULT', 216: 'GBCI', 217:
'TCBI', 218: 'UBSI', 219: 'FIBK', 220: 'CBSH', 221: 'HWC', 222: 'BKU', 223:
'BANC', 224: 'PB', 225: 'ASB', 226: 'UMBF', 227: 'SSB', 228: 'FNB', 229: 'PNFP',
230: 'CADE', 231: 'ONB', 232: 'BOKF', 233: 'CFR', 234: 'COLB', 235: 'WTFC', 236:
'SNV', 237: 'VLY', 238: 'EWBC', 239: 'WAL', 240: 'WBS', 241: 'FHN', 242: 'CMA',
243: 'SBNY', 244: 'NYCB', 245: 'RF', 246: 'KEY', 247: 'HBAN', 248: 'MTB', 249:
'SIVBQ', 250: 'FRCB', 251: 'FCNCA', 252: 'FITB', 253: 'CFG', 254: 'TFC', 255:
'PNC', 256: 'USB', 257: 'WFC', 258: 'C', 259: 'BAC', 260: 'JPM'}

Enter the number(s) corresponding to the ticker(s), separated by comma:  260

Step 1: Keywords and patterns defined.
Step 2: Function to load text defined.
Step 3: Function to extract sentences with fintech keywords defined.
Step 4: Function to classify sentences with ChatGPT defined.
Step 5: Function to clean negative occurrences defined.
Step 7: Function to process files and capturing context defined.
Step 8: Function to save results to CSV defined.
```

```
Step 6: Processing files and capturing context.
Processing file: downloaded_filings_JPM\JPM_2014-12-31.txt
```

```
---------------------------------------------------------------------------
APIRemovedInV1                            Traceback (most recent call last)
Cell In[1], line 149
    146 directory = f"downloaded_filings_{ticker}"
    148 # Process the text files
--> 149 results = process_files(directory, fintech_keywords)
    151 if results:
    152     # Save the results to a CSV file with a dynamic name based on the␣
 ↪ticker symbol
    153     csv_filename = f"fintech_analysis_{ticker}.csv"

Cell In[1], line 115, in process_files(directory, keywords)
    113 # Classify sentences with OpenAI GPT-4
    114 if sentences_with_keywords:
--> 115     sentence_classifications =␣
 ↪classify_sentences_with_gpt(sentences_with_keywords)
    117     # Store the results
    118     base_filename = os.path.splitext(filename)[0]

Cell In[1], line 66, in classify_sentences_with_gpt(sentences)
     60 for sentence in sentences:
     61     prompt = f"""
     62     Please classify the following sentence based on its relevance to␣
 ↪fintech.
     63     Sentence: {sentence}
     64     Respond in json format with three keys "classification",␣
 ↪"probability score" and "Reasons".
     65     """
---> 66     response = openai.ChatCompletion.create(
     67         model="gpt-3.5",  # Adjust model name if using a different mode
     68         prompt=prompt,
     69         max_tokens=150,
     70         n=1,
     71         stop=None,
     72         temperature=0.7
     73     )
     74     response_text = response['choices'][0]['text'].strip()
     75     result = json.loads(response_text)

File ~\anaconda3\Lib\site-packages\openai\lib\_old_api.py:39, in␣
 ↪APIRemovedInV1Proxy.__call__(self, *_args, **_kwargs)
     38 def __call__(self, *_args: Any, **_kwargs: Any) -> Any:
---> 39     raise APIRemovedInV1(symbol=self._symbol)
```

```
APIRemovedInV1:

You tried to access openai.ChatCompletion, but this is no longer supported in␣
 ↪openai>=1.0.0 - see the README at https://github.com/openai/openai-python for␣
 ↪the API.

You can run `openai migrate` to automatically upgrade your codebase to use the␣
 ↪0.0 interface.

Alternatively, you can pin your installation to the old version, e.g. `pip␣
 ↪install openai==0.28`

A detailed migration guide is available here: https://github.com/openai/
 ↪openai-python/discussions/742
```

[9]:

```
  Cell In[9], line 1
    openai migrate
           ^
SyntaxError: invalid syntax
```

[ ]: