```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
from typing import List, Dict, Any
import time
from datetime import datetime, timedelta
from bs4 import BeautifulSoup
import requests
import json
import os

class HanKyungScraper:
    def __init__(self, start_date: str, end_date: str, output_file: str):
        self.start_date: str = start_date
        self.end_date: str = end_date
        self.output_file: str = output_file if output_file else "result.json"
        self.driver: webdriver.Chrome = self._setup_driver()

    def _setup_driver(self) -> webdriver.Chrome:
        chrome_options = Options()
        chrome_options.add_experimental_option("detach", True)
        chrome_options.add_experimental_option("excludeSwitches", ["enable-logging"])

        service = Service(executable_path=ChromeDriverManager().install())
        driver = webdriver.Chrome(service=service, options=chrome_options)
        return driver

    def scrape_and_save(self) -> None:
        articles: List[Dict[str, Any]] = []
        try:
            articles = self.scrape()
        except KeyboardInterrupt:
            # Ctrl + C 입력으로 인한 인터럽트를 처리
            print("Scraping interrupted by user.")
        except Exception as e:
            # 기타 예외 처리
            print(f"An error occurred: {e}")
        finally:
            # 스크랩된 데이터를 저장
            output_path = os.path.join(os.getcwd(), self.output_file)
            with open(output_path, "w", encoding="utf-8") as f:
                json.dump(articles, f, ensure_ascii=False, indent=4)
            print(f"Data saved to {self.output_file}")
    def scrape(self) -> List[Dict[str, str]]:
        self.driver.get("https://www.hankyung.com/all-news")
        self.driver.implicitly_wait(5)
        self.driver.maximize_window()

        elements = self.driver.find_elements(By.CSS_SELECTOR, ".nav-link")
        economi = elements[14]
        economi.click()

        time.sleep(5)

        previous_date_obj = datetime.strptime(self.start_date, "%Y%m%d") - timedelta(days=1)
        previous_date_str = previous_date_obj.strftime("%Y%m%d")

        while True:
            try:
                date_elements = self.driver.find_elements(By.CSS_SELECTOR, ".txt-date")
                for date_element in date_elements:
                  if date_element.text != "":
                     formatted_date = datetime.strptime(date_element.text, "%Y.%m.
%d").strftime("%Y%m%d")
                     if previous_date_str == formatted_date:
                         raise StopIteration
                more_btn = self.driver.find_elements(By.CSS_SELECTOR, ".btn-more")
                for btn in more_btn:
                    if btn.text == "더보기":
                     btn.click()
                    #  time.sleep(2)
            except StopIteration:
                break
        elements = self.driver.find_elements(By.CSS_SELECTOR, ".news-tit a")

        articles: List[Dict[str, str]] = []
        if elements:
            for element in elements:
                if element.text != "":
                    href = element.get_attribute("href")
                    title = element.text
                    articles.append({
                        "title": title,
                        "href": href
                    })

        scraped_data = self._scrape_articles(articles)
        self.driver.quit()
        return scraped_data

    def _scrape_articles(self, articles: List[Dict[str, str]]) -> List[Dict[str, str]]:
        lst: List[Dict[str, str]] = []
        for article in articles:
            response = requests.get(article["href"])
            soup = BeautifulSoup(response.text, 'html.parser')

            date_list = soup.select(".txt-date")
            if date_list:
                date = date_list[0].text
                date_edit = date_list[1].text if len(date_list) > 1 else ""
                content = soup.select("#articletxt")
                href = article["href"]
                title = article["title"]
                if content:
                    article_text = content[0].get_text(separator="\n", strip=True)

                lst.append({
                    "date": date,
                    "date_edit": date_edit,
                    "href": href,
                    "title": title,
                    "content": article_text
                })
        return lst
```

셀레니움으로 경제 페이지의 모든 기사를 불러오고 정보를 리스트에 담아서

해당 링크로 뷰티풀수프로 파싱합니다. 예외처리까지 모두 구현했습니다.