RSM 8301
ASSIGNMENT 4

In this assignment, you will use what you learnt from the NLP class to solve several tasks. It is always the best experience to have hands-on coding implementation to reinforce the knowledge of NLP. The assignment's topic is to analyze public companies' earnings call transcripts and ultimately develop an investment strategy based on the data. The coding environment setup is introduced in the attached document "*Assignment Coding Environment Instruction.docx*". You will need submit your codes and results to complete the assignment.

1. **Data Preprocessing**:
   It is important for NLP models to reduce the size of the vocabulary through data preprocessing and cleaning, especially for word frequency methods. The common data preprocessing steps include:
   - ➢ Convert text to lowercase;
   - ➢ Remove punctuation;
   - ➢ Remove frequent words in the corpus (called stop words), such as "this" "that", and so on.;
   - ➢ Remove numbers;
   - ➢ Perform stemming to reduce inflected or derived words to their root form.

   Please apply the above text cleaning steps to the statements made by Netflex's CEO in the 2020 Q1 earning call session. The data is prepared in "RSM8301-A4-Q1.ipynb". Process the data and save the clean text in "RSM8301-A4-Q1.csv". You need submit the notebook including your code together with the final CSV file.

2. **TF-IDF**
   We have learned how TF-IDF is calculated to represent a document. Let's have a hands-on exercise to implement the TF-IDF formulas. You are given a list of 1,000 tokens' document occurrence counts in the file "*token_doc_counts_a4_q2.csv*". In this file, each row contains a token name and the number of documents that contains this token in the corpus. We obtained the top 1,000 most frequent tokens from the corpus. The corpus includes 362,330 documents which are the earning call statements from 2018 to 2021.
   A document is prepared in "RSM8301-A4-Q2.ipynb". Use the token document counts to generate a normalized TF-IDF vector for the document (please refer to the formulas in page 11 of NLP slide1). You need submit the notebook including your code together with the final CSV file.

3. **Word2Vec**
   The Word2Vec model can capture semantic relationships between words by representing them as dense vector embeddings in a continuous vector space. It is a powerful tool to determine analogous words by comparing the similarity between their embedding vectors. To measure the similarity between two vectors, Cosine Similarity is a common metric. It calculates the cosine of the angle between the vectors, which determines whether the vectors are pointing in roughly the same direction or not. The formula of Cosine Similarity is as follows:

$$CosSim(A,B) = \frac{A \cdot B}{|A||B|}$$

   Where $A \cdot B$ is the dot product of the vectors A and B; $| * |$ is the Euclidean norm of a vector.
   In the notebook "RSM8301-A4-Q3.ipynb", we trained a Word2Vec model by using the corpus of S&P500 earning call transcripts. As learned from the class, we can use the Word2Vec model as a dictionary which maps the word to its embedding vector. Use cosine similarity to calculate the

similarities among the following words:

['fall', 'loss', 'reduction', 'success', 'process', 'pleased', 'confident']

Which two words are most similar? What is their similarity score?

4. Sentence BERT and LLM RAG

The Sentence-BERT (SBERT) model [1] is a modification of the pre-trained BERT model that uses an additive learnable pooling layer to derive semantically meaningful sentence embeddings. SBERT is suitable for semantic textual similarity tasks and, therefore, a good candidate for building a document retrieval tool to support LLM Retrieval Augmented Generation (RAG). In this question, you are asked to build a SBERT based RAG to find the answers for a list of questions from Wells Fargo & Co's (WFC) 2023 January earning call.

Use the RAG framework learned in the class to execute the following steps:

1. Segment the earning call transcript into documents. Tips: SBERT works better with short-length documents; the recommended length of each document is 100 to 150 tokens.
2. Use SBERT to encode the documents and the question. Search for the top 5 most similar documents and compose a context.
3. Create a prompt using the question and context. Send the prompt to OpenAI ChatGPT and gather the answer.

You will use the notebook "RSM8301-A4-Q4.ipynb" to implement the first two steps, and then use ChatGPT to summarize the answer. The questions are as follows:

a. What are the AI tools and Technology innovations WFC introduced?
b. What is the year over year active account increase of WFC's Intuitive Investor tool?
c. How much has WFC's headcount been reduced since 2020?
d. How much deposit decline did WFC experience? And what is the average deposit cost?

You need to submit the notebook including your code together with the entire ChatGPT conversation transcripts for the answers.

5. Investment Strategy

Through the above questions, you will get familiar with our Rotman-NCS python package. For more information, you can always read the PyPI page (rotman-ncs · PyPI), source code (rotman_ncs (github.com)) or contact me through email (mailto:colinzeyu.wang@utoroto.ca).

In this question, you are encouraged to develop an investment strategy based on the earning call dataset. We split the data into training and testing sets. You will use NLP and ML classification techniques learned from the class to train a model by the training data. The model takes the earning call transcripts as inputs and decides the trading action - buy, sell or no action, as output.

We set a baseline strategy with a purely random selection, which trades randomly after each earning call. Its annual simple return is -0.5% when executed within the testing period. Your grade for this question will depend on your strategy's annualized simple return:

- ➢ 5 Points if your strategy beats the baseline strategy.
- ➢ 10 Points if your strategy achieves a 5% annual return.
- ➢ 15 Points if your strategy achieves a 10% annual return.

You are provided with three notebooks (see in "*Assignment Coding Environment Instruction.docx*") which implements strategies based on TF-IDF, word embedding and BERT models. You can start with these examples and change the hyperparameters, such as holding period, statement weights and so on. You can also choose to build your own strategy from scratch, collect extra text data (such as press releases), or explore more NLP techniques. Your strategy will be evaluated by utilizing ` ***ncs.run_strategy***` function which takes an action file and holding period after the earning call (details in the notebook examples).

You need submit:

1. The notebook for your strategy training.
2. An action file which includes call_uid and the corresponding action integer (1: buy; 0: no action; -1: sell) for the testing dataset. random_action.csv is the action file for the baseline strategy.
3. Your selection of holding period after the earning call (1, 5, or 10 days)

Reference

1. Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.