Store Preprocessing Text Result Methods

Text Preprocessing has become a very important task in mordern computer science field such as Artificial Intelligent and Machine Learning. For example, in 2013, E. Haddi et. al have used multiples Text Prerocessing methods to analyse the sentiment of online movie reviews [1]. However, it is a very expensive process, and it becomes extremely expensive when the dataset become bigger. For example, The data scientists have to process millions of online news articles from multiple online publishers including Los Angeles Times, Routers, and New York Times [2].

There are three main components of Pre-processing including Tokenization, Normalization, and Subsitution [3] and their output results can be reused many times. Howerver, those tasks require very high computational resources. In order minimize the computing cost, we can save the output results to file and and then we can reload them while needed. So that saving results to a file becomes a mandatory feature. There are some methods to address this problems and the methods’ technique can be various which depends on the data structure of the results. The methods can be put into two main categories which are Human-readable (text-based) and Non-human-readable (binary).

Since all the results of Text Preprocessing in this assignment is stored in Objects so the problem is narrowed down to “How to store an Java Object to file” and it can be solved using Object Persistence Methods [4], they are basically a process that transform an Object to a format that can be stored on disk or database, howerver, we are only interested in storing in file due to the scope of this assignment.

The first approach is to use the Build-in Java Serialization which make use of object stream classes including *ObjectInputStream* and *ObjectOutputStream* [5] to convert and write an object to a binary file. The second approach is to tranforms the Objects to a human-readable strings using the *Data Markup Language* such as XML or *Data Serialization Language* such as JSON and YAML and then those strings to file.

For my implementation, I would choose the second approach with JSON, although it may not be as efficient as using Java Serialization in term of run time and space, howerver, it is a multi-platforms language, so that I can reuse the result on another programming language and it is also human-readble which easy for error checking.

By using JSON as the *Data Serialization Language*, for each section, I will convert the Map of line number and line text, the Trie as well as all the IndexTables of words to JSON string and then store them in their associated section file,

# **References**

[1] E. Haddi, X. Liu and Y. Shi, "The Role of Text Pre-processing in Sentiment Analysis", *Procedia Computer Science*, vol. 17, pp. 26-32, 2013.

[2] Y. Shinyama and S. Sekine, "Named entity discovery using comparable news articles," in *Proceedings of the 20th international conference on Computational Linguistics*, 2004, p. 848: Association for Computational Linguistics.

[3] M. Mayo, "A General Approach to Preprocessing Text Data", *Kdnuggets.com*, 2018. [Online]. Available: https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html. [Accessed: 12- Oct- 2018].

[4] C. JMTauro, R. Kumar Sahai and S. Rani A., "Object Persistence Techniques - A Study of Approaches, Benefits, Limits and Challenges", *International Journal of Computer Applications*, vol. 85, no. 5, pp. 19-27, 2014.

[5] "The Java™ Tutorials - Object Streams", *docs.oracle.com*, 2018. [Online]. Available: https://docs.oracle.com/javase/tutorial/essential/io/objectstreams.html. [Accessed: 12- Oct- 2018].

[1] S. man and A. Utama Siahaan, "Huffman Text Compression Technique", *International Journal of Computer Science and Engineering*, vol. 3, no. 8, pp. 103-108, 2016.