CS-677 Project Proposal

Word2Vec is a natural language processing(NLP) algorithm developed by Google which in essence, converts a vocabulary based on a corpus of text documents into vector space where each word/term of the vocabulary is denoted by a vector of k-dimensions. Theoretically, the size of the vector space is n-dimensions where n is the size of unique words/term count in the vocabulary. This vector space provides data scientists with a mathematical way to interpret the semantic meanings of words. Albeit papers showing vector space sizes in ranges of 300 to 400 being as effective as n unique words depending on the corpus, the sheer size of the vectors still proves a computational hassle during NLP analysis.

When working at McKinsey and Company alongside their analytics team on an NLP project, we wanted to model topics in a text corpus and came across the idea of clustering and trying to interpret Word2Vec vector space. This idea came out of words relative meaning being modeled as vectors and therefore similar words would be clustered and un-similar words would be further apart. However, the development time to research and implement this function with Word2Vec was unfavorable and instead, we chose to use the Latent Dirichlet allocation to model topics and never got the chance to implement Word2Vec. However, I have always wondered what the results of clustering on mathematical semantic interpretations of words would be. Thus, this purpose of this project is to perform the k-means clustering algorithm on Word2Vec vector space of a large vocabulary, over various vector sizes to represent the vocabulary. In the interest of the course and optimizing parallelization over many cores, I will explore this problem over various values of k for k-means. In order to compute this, I will use data sets provided by previously trained Word2Vec models provided by Google, research papers, universities, companies, etc. and if needed the ability to create my own data set is simple.

K-means traditionally takes the number of clusters desired and the data set to perform k-means on. The algorithm follows a two-step process of designating each data point in Euclidean space to the nearest center point of a cluster, also known as a centroid. The next step is to update the centroids by taking the mean center of all points designated to that centroid. This process is repeated until satisfaction. However, k-means is more traditionally used on data sets based in 2 or 3 dimensions. For every comparison to find the nearest centroid or to recalculate the mean of clusters, the computation will be between vectors of size greater than 300 ranging into the thousands. Therefore, this problem is perfect for massive parallelization with a simple algorithm providing repetitive computations over a large dataset that can spread across a large number of threads. In order to parallelize k-means, I would parallelize the computation of the distance between vectors and centroids by tiling on the vector length. Moreover, I would parallelize the recalculation of the centroids through a similar process. Thus, this approach would encounter multiple iterations of memory loading per comparison which would lead to the need to optimize memory loads across threads. The memory life cycle of the algorithm would require the host to copy the vector space of Word2Vec to the device(s) as vectors and the device would have to return to the host what cluster each vector belongs to.

This project interests my curiosity on what this process will return. However, the challenge of performing a common algorithm in computer science over an extended and large vector space poses an even more interesting challenge. Of the challenges, optimizing memory access will be the most difficult due to the sheer size of data needed to perform comparisons required in clustering.