Problem: Extracting documents containing key words “pre frontal cortex” and “posterior parietal cortex” from the database <https://www.semanticscholar.org/> along with summarising anatomical relation between these two brain regions.

Probable steps:

1. Screen all articles using word match “pre frontal cortex” AND “posterior parietal cortex”.

The mining of given data can be performed by creating Document Term Matrix. This matrix will contain rows and columns, where each row will represent a separate entry and columns will represent all the variables (here the variables are 2 in number majorly but we need to consider their abbreviations and substitution words for eg. PFC for pre fontal cortex) . The value of each entry in the matrix can be calculated using Term Frequency-Inverse Document Frequency (tf-idf) scheme. Mining approach can also be extended with Latent semantic analysis. In this approach, matrices containing all other variables per one single variable (other variables could be all variable associated with posterior parietal cortex) can be constructed. We can then use singular value decomposition method to reduce the number of rows while preserving the similarity among columns. The entries in the rows can then be compared by taking dot product between the row vectors. These way similar and dissimilar words can be identified for each variable (for all the given variables with respect to each other) and clustered together.

1. Focusing on Results and discussion part of the articles founded in the above step.
2. Finding relation-

To find the relationship between two areas, we can transform each word (“pre frontal cortex” AND “posterior parietal cortex”) into Document Term Matrix. This matrix will contain 1s and 0s, where 1 will represent the presence of the name and 0 will represent the absence. Then we can perform correlation analysis to investigate the relationship between two variables.

1. Summarization-

a*. What would be a crude method of summarizing this text? Substantiating with the open access third party text mining modules/components available.*

Answer:

--We can use LexRank algorithm for summarizing the text documents. This unsupervised approach uses a method for extracting important sentences in the given corpus. LexRank is an algorithm designed for resolving extraction summarization problem and it is similar to PageRank. LexRank is basically a graph based approach where every sentence in the corpus is represented as one node and similarity relationship between the sentences in corpus is represented as edges. The likeness between the sentences is estimated by frequency of word occurrence in a sentence. The similarity strength measurement is basically performed using TF-IDF (Term Frequency-Inverse Document Frequency) formulation. TF is higher when the number of word occurrences is higher, while IDF measures the information providing capacity of word. Distance between the two sentences is measured by idf-modified-cosine formula.

idf- modified cosine (x,y) = , where x and y are two sentences

The approach brings more similar sentences more close to each other. Similarity matrix is formed which can be used as similarity graph between sentences. The LexRank algorithm will then measure the importance of sentence in the graph by considering its relative importance to its neighboring sentences. To extract the most important sentences, a thresholding approach is applied. A threshold value will filter out the relationships between the sentences whose weights are less than the threshold. From the resulting subset of similarity graph, the node with highest number of degree is selected as a summary sentence of the corpus.

--We can implement LexRank using Python based library ‘sumy’

*b. What would be best approach to test the performance of this summarization module?*

Answer:

-ROUGE metric is a standard way of evaluating automatically generated summaries of text documents. It compares machine generated summaries with human generated reference summaries. It is a recall based evaluation method, where recall can be computed with respect to unigram, bigram, trigram or four-gram matching. The formula for this metric can be given as follows:

-We can use ‘pyrouge’ python wrapper for the ROUGE summarization evaluation