Overview:

The basic idea behind my approach was to use Word2Vec to train word embeddings for all words that occur in the ICD10 descriptions. Then using a weighting scheme such as a simple average, term frequency, etc. defined the embedding of an ICD10 code as a weighting scheme applied to the embeddings for all words in the given ICD10 code's description.

Data:

I figured I should somehow use the hierarchical structure of ICD10 codes to retrieve different types of text from PubMed and Wikipedia while reading through the code descriptions. For every 3 character ICD10 code, I query Wikipedia for a page summary and PubMed for the top 3 articles returned using the code's description as a query. I figured this was a good way to cover supplemental texts for a wider range of the codes while keeping runtime down. If I queried on 4 letter codes, runtime increased exponentially.

Training Embeddings:

I used the Word2vec SkipGram model to train word embeddings. It is the first thing that popped into my head for training word embeddings and I was familiar with the library. The data for the embeddings always includes all ICD10 long descriptions. I have included embeddings for only the descriptions as data and descriptions along with supplemental texts as data.

Weighting Schemes:

I included three different weighting schemes for computing an ICD10 code embedding: simple average, term frequency by family, and a tf-idf-like scheme. The functions are as follows:

- **Simple Average -** The average over all word embeddings in a code's description. Each word is weighted equally
- **Family Frequency -** Trying to incorporate the subcategories of codes better I grouped each family defined by the first three digits of an ICD10 code. This scheme weights the embeddings for each word in a description by the frequency at which it occurs inside it's family. Looking back, I think this would have worked better or at least made more sense to do over the first 4 letters of a code.
- **TF-IDF-Like -** Figuring that words like 'right' or 'left' would occur through the ICD10 descriptions while things like hemarthrosis would not, this seemed like a good scheme to try and highlight the main descriptor of a family of codes. For this, I use the TF within a family and consider a description to be a document for the purposes of IDF.

Results:

I think using the simple average and family frequency weighting schemes weight words like 'right' and 'left' or 'unspecified' as too high while the tf-idf scheme sort of drowns out those types of words. For many queries I tried, asking for the top 5 most similar, I got at least 2 or 3 words that were obviously relevant to the query, while sometimes I received a set of scientific words that were generally related to the query term. By no means do I think these are SOTA results or anything, but with using a small amount of data and time I think it was a decent outcome. I had fun thinking about weighting schemes, thanks for the project!