**CA08**

Due 3/31 11:59 pm

This CA is based on k-means in the previous module over a much reduced version of the Java Development Tools Bugs dataset named JDT\_Bugs\_sm.csv provided below.

The dataset consists of 6 categories of bug reports under the column 'Component'. Your task is to cluster reports into 6 clusters using the k-means clustering algorithm with k = 6.

1. Data preparation: Clean, normalize, and lemmatize reports. Remove tokens that are not English words (including punctuations, numbers) and remove spaCy stopwords. Hints and suggestions: (1) Use regular expressions to clean data (as we have always done using function "clean" defined earlier). (2) Use textacy to normalize data. (3) Use spaCy for lemmatization using the following snippet to reduce processing time:

nlp = spacy.load("en\_core\_web\_sm", disable=["tok2vec", "tagger", "parser", "attribute\_ruler","ner"])

for i, row in tqdm(df.iterrows(), total=len(df)):  
    doc = nlp(str(row["clean text"]))   
    df.at[i, "lemma"] = " ".join(token.lemma\_ for token in doc if wanted\_word(str(token)))

df above is the dataframe of the dataset, df["clean text"] is the clean and normalized text, and wanted\_word(str(token)) returns True if token is a legitimate English word (you'd need to implement this function yourself; e.g., you may use NLTK WordNet library from nltk.corpus import wordnet and use wordnet.synsets(str(token)) to determine if token is a legitimate English word, which returns True if it is).

1. Feature engineering (text vectorization): Represent each report as a vector of tf-idf values. Hints and suggestion: use sklearn TfidfVectorizer to compute tf-idf values for df["lemma"].
2. K-means implimentation: Implement k-means using cosine similarity on vector representations. Hints and suggestions: (1) Select 6 reports, one from each category, as initial centroids. (2) Add a column of cluster labels in df to have 6 labels of 0, 1, ..., 6, and take advantage of the dataframe data structure. (3) You may use the following line to return the needed index, where c[j]is a sub-dataframe of df with df[df["cluster label"] == j]:         idx = df.index.get\_loc(c[j].iloc[row].name)
3. Run k-means on the tf-idf vectors of reports for 10 epochs (this running may take upto 15 minutes and so during debugging you should run your code for 1 epoch). Is the resulted clustering reasonably consistent with the  component labels? Justify your finding.

Additional task for 20 bonus points: Represent each report as a vector of BM25 values in Task 2 and repeat Tasks 3 and 4. You may borrow code at <https://gist.github.com/koreyou/f3a8a0470d32aa56b32f198f49a9f2b8>, which provides a sklearn TfidfVectorizer implementation of BM25.