Jacob Kinser, Austin Buller:

* Need specifics on evaluating the models
* Be aware of how lemming and stemming can affect software documentation data
* Look into different vectorization models

Dhanush, Dan:

* POS weaknesses and strengths: word tokenize, character tokenize, subword tokenize
* Random Forest, Gradient Boosting

Will, Kelly, Zach

* N-Gram, Count Vectorizing, TF-IDF
* Feature importance
* Supervised, Unsupervised ML => clustering

How do we evaluate a Vectorization model?

* After getting the word vectors, we apply a machine learning algorithm on top to evaluate (Random Forest)
* Above approach doesn’t work every time because normally data doesn’t have labels ready.
* POS tag. How do we evaluate if a tag is correct? Is dance a noun or verb?

**Goals for next week:**

* When present, always cite articles that contain the information you presenting, please search more on your own besides the sample articles below.
* Focus on Software Documentation data: if I use this strategy, how would it affect a software document?
* Explain more clearly the weaknesses and strengths of each vectorization strategy?
  + word tokenize, character tokenize, subword tokenize? Which models in Python that use these strategies? Which would work better for software documentation (use your own reasoning for debate)
  + N-Gram, Count Vectorizing, more?
* Look into specific Vectorization models:
  + **Smoothing, Smooth Inverse Frequency (SIF)** and **TF-IDF**: What are they and how are they applied to word embedding models
    - Sample with TF-IDF: <https://intellica-ai.medium.com/comparison-of-different-word-embeddings-on-text-similarity-a-use-case-in-nlp-e83e08469c1c>
    - Sample with SIF: <https://towardsdatascience.com/fse-2b1ffa791cf9>
  + What are the word embedding (POS tag) models in NLTK: *word2vec CBOW, Skipgram, Fastex*t, etc
  + Other than NLTK: openNLP**, StandfordNLP, spaCy,** Glove, TextBlob. NLTK and these libraries use Neural Network Strategy, how are they different from SIF and TF-IDF embeddings?
  + Try to reason which model would work best for software documentation data and why. Some
* Find new datasets with or without labels. If without, apply new strategies to evaluate the word embeddings (Python clustering methods)

<https://www.kaggle.com>

<https://www.kaggle.com/fazilbtopal/popular-unsupervised-clustering-algorithms>

* Research on how to evaluate vectorization models other than using ML? Cosine Similarity between words, etc.
  + Try to search evaluation methods for **unsupervised learning.** Many found approaches can be quite manual like you group articles together and evaluate yourself if these grouped articles have the same topic
  + Look into articles like this one: <https://aclanthology.org/D15-1036.pdf>
* **Future goal**: Divide groups to different tasks:
  + Data cleaning
  + Model 1: NLTK
  + Model 2: Stanford
  + Model 3: spacy
  + Evaluate + Conclude