### **1. Few Shot Classification Using Pre-Trained Language Models**

#### **Models: T5 few shot, BERT few shot, Prototypical Network**

* **Description**: Create a few labeled examples for each y value - 5-100 depending on time constraints, no need for creation of unique sentences, just some minimal amount of labeling required.

### **2. Zero-Shot Classification Using Pre-trained Language Models**

#### **Model: facebook/bart-large-mnli (or similar)**

### **Description:** This model can perform zero-shot classification by using natural language inference (NLI) to determine if the input sentence X matches or "entails" anything in Y. I'm hoping this will be accurate, I have no way to tell except for actually implementing the code.

### **3. Sentence Embedding Similarity Using Sentence Transformers**

#### **Model: https://huggingface.co/sentence-transformers/all-mpnet-base-v2**

* **Description**: In this method, both the input sentence X and the sentences in Y are converted into dense vector embeddings using a sentence transformer model. The classification is done by comparing the cosine similarity between the embedding of X and each sentence in Y.
* **How it works**:
  + Create is hidden and unique sentence descriptions for each of the y values whichever y value has the highest probability of being connected to X is chosen.
  + Basically the sentence with the highest similarity score is selected.
* **Evaluation**:
  + Cosine similarity will be used to determine how well this model performs in matching X to Y. Dependent on how many unique, accurate and generalized sentences we are able to write for each y value.

#### **Potential Problems:**

* May struggle when sentences are very similar but have different meanings. Having Unique sentences is a must.

### **4. TF-IDF Vectorization with Cosine Similarity**

#### **Model: TF-IDF with Cosine Similarity**

* **Description**: A more traditional information retrieval technique, where we use Term Frequency-Inverse Document Frequency (TF-IDF) to represent sentences as vectors. We then compute cosine similarity between the vectors of Xand Y to classify X.
* **How it works**:
  + Almost exact same process as previous model, maybe lower accuracy in comparison due to being a simpler model
* **Evaluation**:
  + The classification accuracy will be measured by how well it matches X with the correct sentence.

#### **Potential problems:**

* May not handle long or complex sentences effectively.
* Will be better if our hidden sentences are simple as well.

Other methods include -

* **bert-base-uncase** with fine-tuned for sentence matching - Synthetic data examples would have to be made similar to how we created it for user stories, but doesn’t meet the objective as we would have to create a decent sized amount of data for this to be effective along with creating hidden sentences.
* **Paraphrase-mpnet-base-v2 -** Similar thought process (doesn't seem effective), detect paraphrases between the hidden sentences.