1. **Load data**

Load the given labeled dataset such that, each sentence from all documents is an element in the list.

Load the given unlabeled dataset such that, each sentence from all documents is an element in the list.

# View the first 5 lines of labeled\_corpus.

['### abstract ###\n',

'MISC\talthough the internet as level topology has been extensively studied over the past few years little is known about the details of the as taxonomy\n',

'MISC\tan as node can represent a wide variety of organizations e g large isp or small private business university with vastly different network characteristics external connectivity patterns network growth tendencies and other properties that we can hardly neglect while working on veracious internet representations in simulation environments\n',

'AIMX\tin this paper we introduce a radically new approach based on machine learning techniques to map all the ases in the internet into a natural as taxonomy\n',

'OWNX\twe successfully classify NUMBER NUMBER percent of ases with expected accuracy of NUMBER NUMBER percent \n']

1. **Pre-process the text (on both labeled and unlabeled dataset)**

You can perform any of the followings:

* Remove the '### abstract ###' and '### introduction ###' titles
* Remove stopwords
* Remove punctuation

After this step, you should be working on a clean dataset.

1. **Vectorize the clean labeled dataset**

You may remove the labels for this step. But note that you need the labels for future steps.

1. **Create the Logistic Regression classifier (sample code available in blackboard)**
2. **Evaluate the Logistic Regression classifier**

Sample code:

log\_reg\_report = classification\_report(Testing set,

Predicted testing set,

labels = ['AIMX', 'OWNX', 'CONT', 'BASE', 'NUMBER', 'MISC']

)

print(log\_reg\_report)

1. Try other classifiers: support vector machine and decision tree.

svm = SVC()

​

# Create the decision tree.

tree = DecisionTreeClassifier()

​

# Fit the svm to the training data.

svm.fit(train\_data, train\_labels)

​

# Fit the decision tree to the training data.

tree.fit(train\_data, train\_labels)