* The computer needs to understanfd the text. The field that helps for this is called **Natural Language Processing.**
* The main goal of this is to understand and derive then meaning of the text from human language.
* ELIZA, a chatterbox, a NLP progress that focusses on understanding grammer.
* It is difficult for computers to understand text as it might contains ambiguity.
* Amazon Mechanical Turk:
  + A service that breaks tasks down into small components and distribute them in online.
  + People can sign up and solve those tasks for a fee
  + But these tasks are time consuming and need more human power to solve.
  + Each worker labels the tweets with numerical values and gives the count for each label.
  + After that we take the averag of all the workers scores for each tweet.
* **HOW to convert text to independent variables**:
  + Understanding the text completely is difficult.
  + So we use **BAG OF WORDS** that counts the numbers of times each word appears. These counts are used as independent variables.
  + Pre- processing can improve the performance of bag of words.
    - One part of pre-processing is to clean up Irregularities.
    - Remove all the characters other than letters and numbers.
    - Removing unhelpful terms (Stop Words)
    - Stemming:The algorithm to specify a common term for all the words that has similar spelling in the beginning
      * For example: argue, argued, argues ⇒ argu
      * Two methods for stemming are:
        + Build a database of words and their stems

Pro: handles exceptions

Con: won’t handle new words

* + - * + Write a rule absed algorithm

Removes ed, ing, ly

Pro: handles new owrds

Con: many exceptions

* TO handle dataset with text:
  + Tweets = read.csv(“tweets.csv”, stringAsFactors = FALSE)
  + TO preprocess our data set :
    - Install.packages(“tm”)
    - library(“tm”)
    - install.packages(“SnowballC”)
    - library(SnowballC)
    - We need to convert our data into corpus, a collection of documents to preprocess
      * corpus = Corpus(VectorSource(tweets$Tweet))
    - corpus = tm\_map(corpus, tolower)
    - corpus = tm\_map(corpus, removePunctuation)
    - corpus = tm\_map(corpus,removeWords, c(“apple”, stopwords(“english”)))
    - The above command removes all the english stopwords and apple words in the documents.
    - corpus = tm\_map(corpus, stemDocument)
  + To get the frequency of the words(BAG OF WORDS)
    - tm model provides a method called Document term matrix that has documents as rows and words in documents as columns. Frequency count are the values in the matrix.
    - Frequencies = DocumentTermMatrix(corpus)
    - To look the values of the matrix
      * inspect(frequencies[1000:1005, 505:515])
        + 1000:1005 ⇒ row numbers
        + 505:515 ⇒ column numbers
    - To get the specified frequecy terms:
      * findFreqTerms(frequencies, lowfreq = 20)
      * It dispalys the words that has frequency> 20
    - sparse = removeSparseTerms(frequencies, 0.98)
      * 0.995 means the terms that appear in 2 or more percent of the tweets)
    - To convert sparse matrix as dataframe
      * TweetsSparse = as.data.frame(as.matrix(sparse))
    - TO amke sure that the variable names are appropriate
      * colnames(tweetsSparse) = make.names(colnames(tweetsSparse))
    - To add a dependent variable
      * tweetsSparse$Negative = tweets$Negative
  + To split data set:
    - library(caTools)
    - set.seed(123)
    - Split = sample.split(tweetsSparse$Negative, SplitRatio = 0.7)
    - trainSparse =subset(tweetsSparse, split == TRUE)
    - testSparse =subset(tweetsSparse, split == FALSE)
  + To build the logistic model
    - library(rpart)
    - library(rpart.plot)
    - tweetCART = rpart(Negative ~. , data = trainSparse, method = “class”)
    - prp(tweetCART)
    - predictCART = predict(tweetCART, newdata = testSparse, type = “class”)
    - table(testSparse$Negative, predictCART) ⇒ confusion matrix
    - table(testSparse$Negative) ⇒ compare probabilities in both tables(this and above)
  + To build random Forest model
    - library(randomForest)
    - set.seed(123)
    - tweetRF = randomForest(Negative ~., dat = trainSparse)
    - PredictRF = predict(tweetRF, newdata = testSparse)
    - table(testSparse$Negative, predictRF) ⇒ confusion matrix