Collaborate ,Learn, Build, Grow

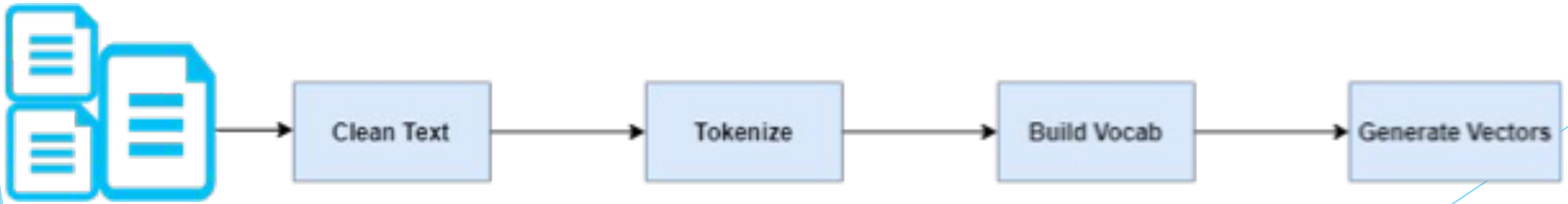
AI-Chatbot Challenge
Data processing and cleaning

Goals

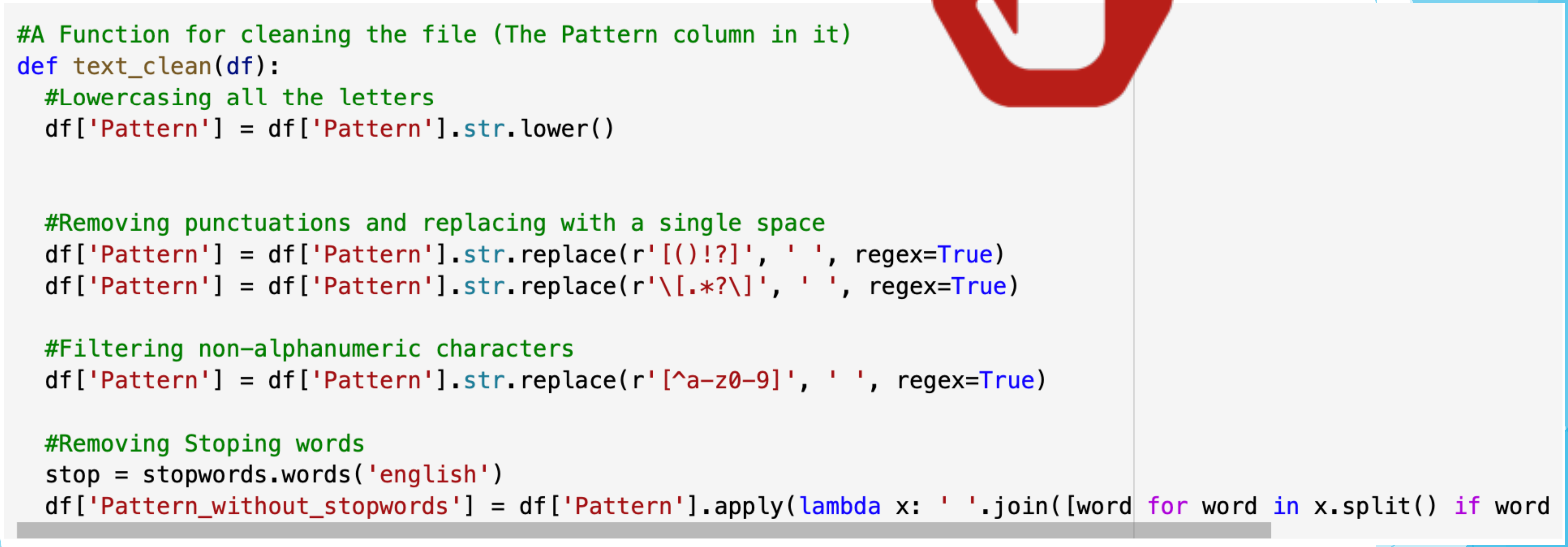
1. Cleaning words in sentence patterns, tokenization, and stemming.
2. Extracting features from the dataset.
3. Encoding the words in sentence patterns into numerical values (array/matrix).

NLP Pipeline

1. Collect and label the data.
2. Clean the dataset and remove stop words.
5. Tokenization.
6. Lemmatization and stemming.
8. Build a vocabulary.
9. Encoding words to numbers. Bag of words.

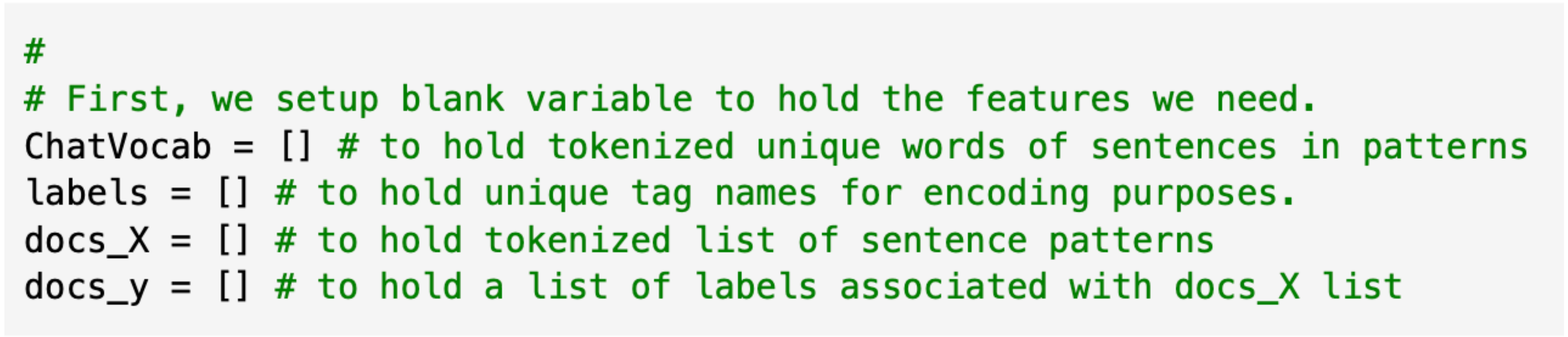


Cleaning the data

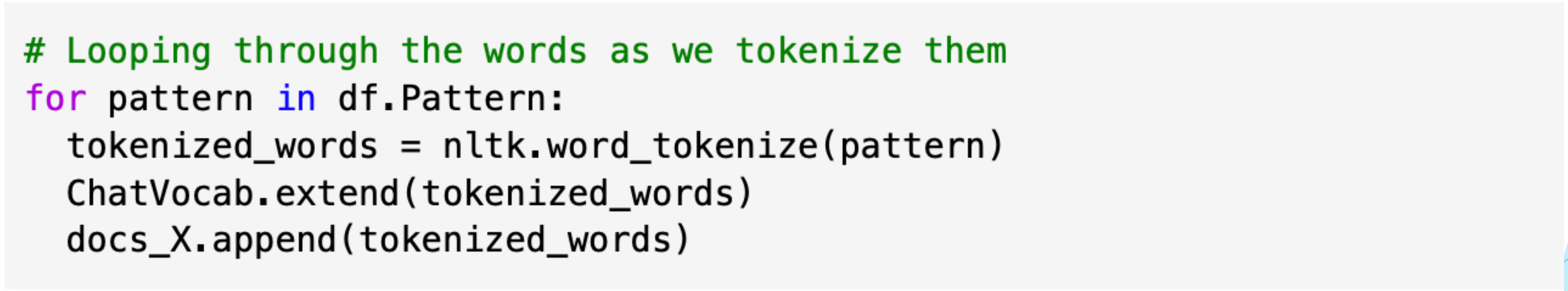


```python
#A Function for cleaning the file (The Pattern column in it)
def text_clean(df):
  #Lowercasing all the letters
  df['Pattern'] = df['Pattern'].str.lower()


  #Removing punctuations and replacing with a single space
  df['Pattern'] = df['Pattern'].str.replace(r'[()!?]', ' ', regex=True)
  df['Pattern'] = df['Pattern'].str.replace(r'\[.*?\]', ' ', regex=True)

  #Filtering non-alphanumeric characters
  df['Pattern'] = df['Pattern'].str.replace(r'[^a-z0-9]', ' ', regex=True)

  #Removing Stoping words
  stop = stopwords.words('english')
  df['Pattern_without_stopwords'] = df['Pattern'].apply(lambda x: ' '.join([word for word in x.split() if word
```

Tokenization



```
#
# First, we setup blank variable to hold the features we need.
ChatVocab = [] # to hold tokenized unique words of sentences in patterns
labels = [] # to hold unique tag names for encoding purposes.
docs_X = [] # to hold tokenized list of sentence patterns
docs_y = [] # to hold a list of labels associated with docs_X list
```



```
# Looping through the words as we tokenize them
for pattern in df.Pattern:
  tokenized_words = nltk.word_tokenize(pattern)
  ChatVocab.extend(tokenized_words)
  docs_X.append(tokenized_words)
```

Lemmatization and stemming

```
lmtzr = WordNetLemmatizer()
df['lemmatize'] = df['tokenized_sents'].apply(
                          lambda lst:[lmtzr.lemmatize(word) for word in lst])
df.head(20)
```

```
#creating a list of root words using our earlier imported stemmer from nltk
ChatVocab = [stemmer.stem(word.lower()) for word in ChatVocab if word != "?"]

# I have only removed "?" which is most likely to occure in chats
```

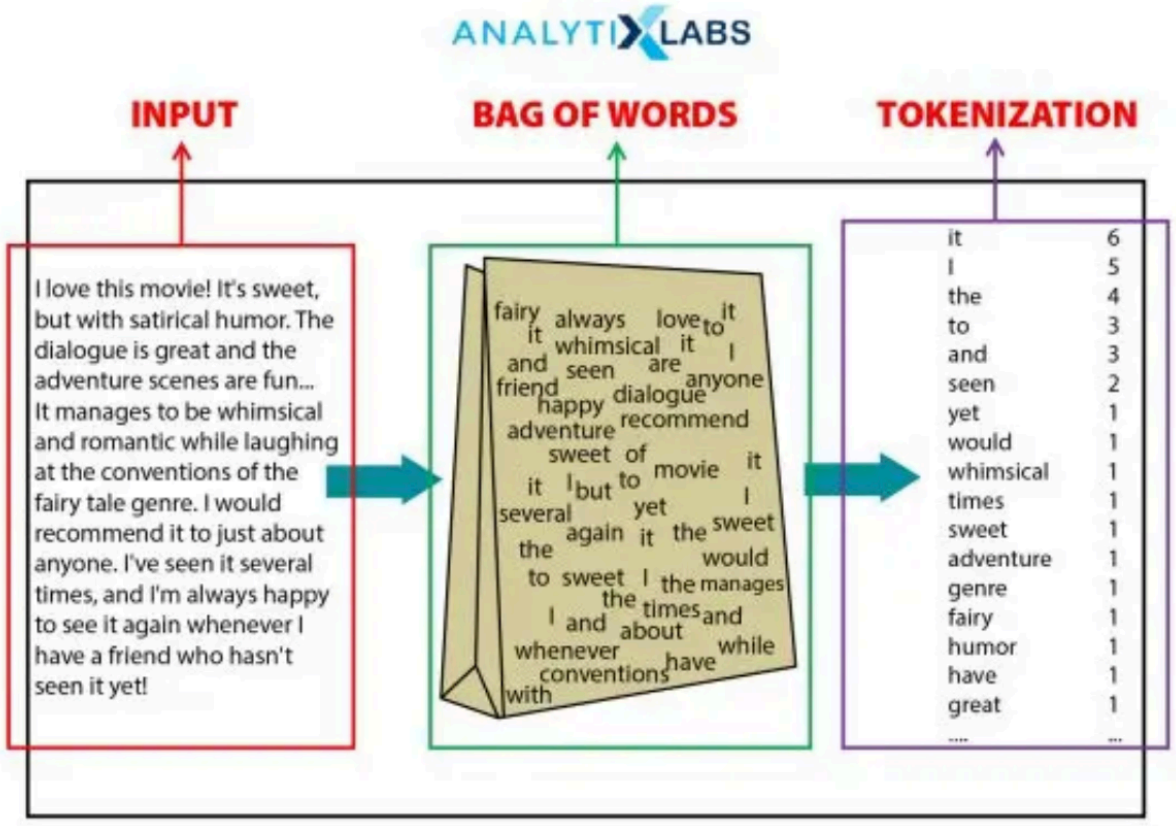
```
ChatVocabulary = sorted(list(set(ChatVocab)))
ChatVocabulary[:10]
```

```
["''", ',', '.', 'a', 'about', 'account', 'afternoon', 'am', 'am….i', 'and']
```

```
len(ChatVocabulary)
```

```
109
```

Bag of words


ANALYTIXLABS
INPUT
BAG OF WORDS
TOKENIZATION
fairy always love it to it whimsical it I and seen are friend dialogue anyone happy recommend adventure sweet of it movie it I but to yet I several again it the sweet the would to sweet I the manages the times and I and about whenever while conventions have with

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
....	...

```
output_data
```

```
array([[0, 0, 1, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 1, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
training_data
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Thank you for your attention!