Report When I only choose 1000 most frequent, I observed homogeneisity of the occurrence (lots of '1'), so I expanded to 5000 and introducted Laplace smoothing to counteract the zero counts for certain words to make to multiplication invalid. Also, rule out some function words which we can't get too much information from to improve the performance.

In [1]:

```
# import packages
import pandas as pd
import numpy as np
from collections import Counter
```

In [2]:

```
# read data
target = pd.read_csv('trg.csv').to_numpy()
test = pd.read_csv('tst.csv').to_numpy()
```

In [3]:

```
# Collect all words in abstracts
abstracts = target[:,2]
vocabulary = []
for abstract in abstracts:
    for word in abstract.split():
        vocabulary.append(word);
```

In [4]:

```
# Eliminate words that make little sense to classification (i.e. numbers, fun
vocabulary = list(filter(lambda a: a.isalpha()!=0, vocabulary))
function_words = ["a","about","above","after","again","against","ain","all","
for function_word in function_words:
    if word in vocabulary:
        vocabulary = list(filter(lambda a: a!= function_word, vocabulary))
```

In [5]:

```
# Find the most frequent words
most_frequent = []
for counter in Counter(vocabulary).most_common(5000):
    most_frequent.append(counter[0])
```

In [6]:

```
# We add each of the most frequent words as an separate attribute, with 1 or
training = np.concatenate((target,np.zeros((len(target),len(most_frequent)),d
for rowindex in range(0,len(training)-1):
    for columnindex in range(0,len(most_frequent)-1):
        if training[rowindex,2].find(most_frequent[columnindex]):
            training[rowindex,3+columnindex] = 1
```

In [7]:

```
# Naive Bayes Algorithm
```

In [8]:

```
## Priors
priors= {'A':0,'B':0,'E':0,'V':0}
for prior in Counter(training[:,1]).most_common(4):
    priors[prior[0]] = prior[1]/len(training)
```

In [9]:

```
## Conditionals with Laplace smoothing
donominatorcounts= {'A':0,'B':0,'E':0,'V':0}
for prior in Counter(training[:,1]).most_common(4):
    donominatorcounts[prior[0]] = prior[1] + len(most_frequent)

# nominatorcounts is the sum of numbers in the columns of the correponding cl

# Then, we can apply the probabilities to test set and choose the class of Hm
```