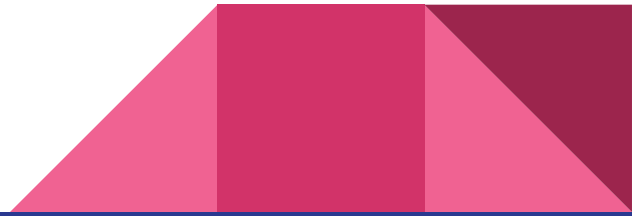
Sentiment Analysis with Naive Bayes Classifier

CS 481 - Artificial Intelligence Language Understanding
Illinois Institute of Technology

The Team

- Mohammad Firas Sada: Co-term CS/AI
- Aleksander Popovic: Co-term CS/AI

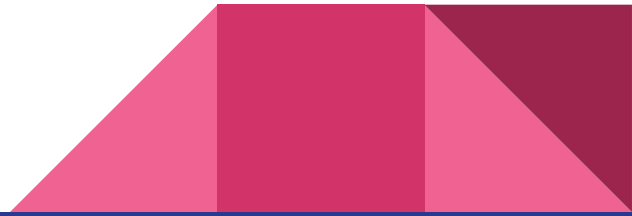


Assignment

- Implementing a naive bayes classifier from scratch to perform sentiment analysis on a dataset of our choice from Kaggle.
- Using the classifier in two training conditions:
  - With all the necessary pre-processing steps
  - Skipping one step of our choice
- Evaluating the efficiency and accuracy of both models and offering a comparison between the two models.

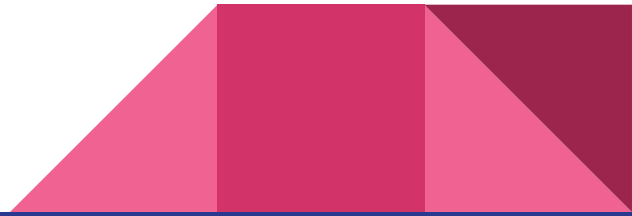
The Dataset

- The dataset is called "Amazon Reviews for Sentiment Analysis" and it contains customer reviews from Amazon, along with star ratings as labels, which can be used for training a machine learning model to perform sentiment analysis.
- The data is provided as a zip file which contains two bz2 files. One file contains the training dataset and the other contains the testing dataset.
- Each dataset is a text file containing binary strings representing the Amazon reviews.



The Dataset (continued)

- The dataset contains a total of 4,000,000 datapoints, which are individual Amazon reviews.
- The split between the training and testing datasets is 90/10, meaning that 90% of the data is used for training and 10% is used for testing.
- Initially, we tried merging the entire dataset and then splitting it into 80/20 to meet the requirements, but this approach proved to be resource-intensive due to the large number of datapoints (4,000,000) in the dataset.



The Dataset (continued)

- Therefore, we decided to use a smaller subset of the data (100,000 reviews) for training and testing our model, which allowed us to achieve reasonable accuracy without overwhelming our resources.
- Datapoints are labelled as positive/negative with with ___label__X notation:

```
dataset[0]
```

```
'__label__2 Stuning even for the non-gamer: This sound track was beautiful! It paints the senery in your mind so wel
l I would recomend it even to people who hate vid. game music! I have played the game Chrono Cross but out of all of
the games I have ever played it has the best music! It backs away from crude keyboarding and takes a fresher step wi
th grate guitars and soulful orchestras. It would impress anyone who cares to listen! ^_^\n'
```

Training / Test Sets

- The dataset is huge (~500MB).
- At first our approach is using the Kaggle API to download and extract the dataset.

```
###
### Connecting to Kaggle, downloading and extracting the dataset
###

try:
    if os.path.exists('input/test.ft.txt.bz2') and os.path.exists('input/train.ft.txt.bz2'):
        print("Dataset already downloaded!")
    else:
        os.environ['KAGGLE_USERNAME'] = "XXXXXXXXXXXXX"
        os.environ['KAGGLE_KEY'] = "XXXXXXXXXXXXXXXXXXXXXXXXX"

        !kaggle datasets download -d bittlingmayer/amazonreviews
        shutil.move('amazonreviews.zip', 'input/amazonreviews.zip')
        with zipfile.ZipFile('input/amazonreviews.zip', 'r') as zip_ref:
            zip_ref.extractall('input/')
    os.remove('input/amazonreviews.zip')
except:
    print('Error downloading and extracting the dataset.\nPlease make sure you have all the requirements in requireme
```

Training / Test Sets

- However, that approach is very slow, because it involves downloading the entire dataset (4,000,000 entries) and extracting compressed data.
- In our data preparation, we download the Kaggle dataset, and we extract a subset of 100,000 entries and process them into a csv file.
- Our submission uses the csv data.

```
try:
    print('\nLoading the dataset...\nThis may take a few minutes...')
    with open('input/dataset.csv', 'r') as file:
        train_file_lines = file.readlines()
        train_file_lines = [line.strip()[1:] for line in train_file_lines]
except:
    print('Error loading the dataset. Please make sure input/dataset.csv exists!')
```

Training / Test Sets

```
print('Training data length:' ,len(train_file_lines))
print('Testing data length:' ,len(test_file_lines))
print('The ratio is: ', len(train_file_lines)//len(test_file_lines),':1', sep = '')
```

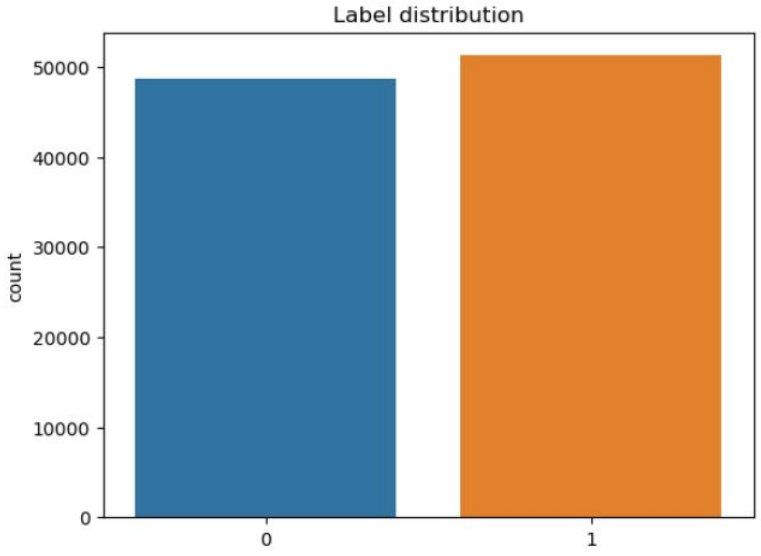
```
Training data length: 3600000
Testing data length: 400000
The ratio is: 9:1
```

```
###
### The assignment asks for a 80/20 split, therefore, we combine the two sets and split them later
### The dataset is huge, and 4,000,000 is a very huge length
### We chnage our approach and take a subset of 100,000

#dataset = train_file_lines + test_file_lines
dataset = train_file_lines[:100000]
len(dataset)
```

```
100000
```

Training / Test Sets



Training / Test Sets

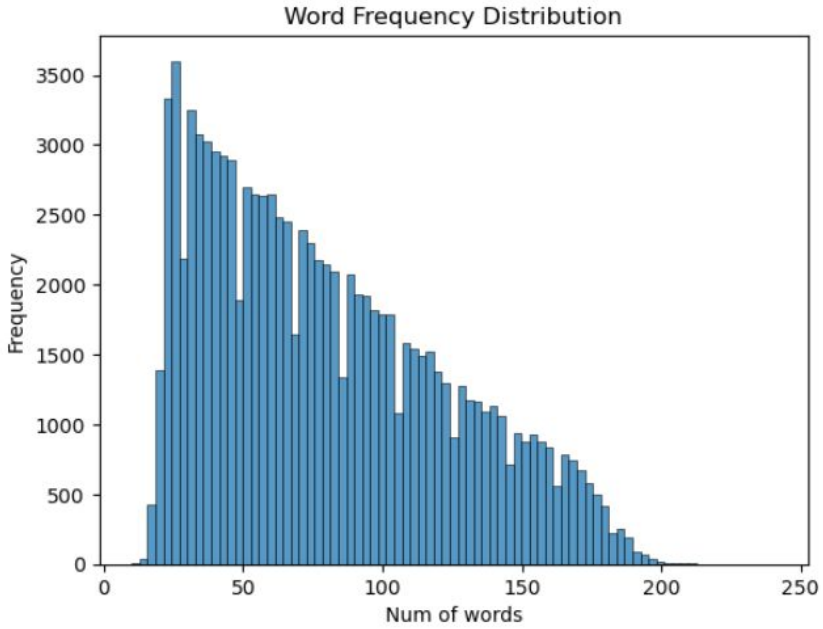
```
###
### Data preparation: extract taining data
###

dataset = [x.split(' ', 1)[1][:-1] for x in dataset]
```

```
print(dataset[0])
```

```
Stuning even for the non-gamer: This sound track was beautiful! It paints the senery in your mind so well I would re
comend it even to people who hate vid. game music! I have played the game Chrono Cross but out of all of the games I
have ever played it has the best music! It backs away from crude keyboarding and takes a fresher step with grate gui
tars and soulful orchestras. It would impress anyone who cares to listen! ^_^
```

Training / Test Sets



Training / Test Sets

```
###
### Data preparation: cleaning out URLs
###

for i in range(len(dataset)):
    if 'www.' in dataset[i] or 'http:' in dataset[i] or 'https:' in dataset[i] or '.com' in dataset[i]:
        dataset[i] = re.sub(r"([^ ]+(?<=\.[a-z]{3}))", "<url>", dataset[i])
```

```
list(filter(lambda x: '<url>' in x, dataset))[0]
```

```
"Bad Deal!!: I ordered this DVD and received a substitute I never received the DVD I ordered from Importcds (the Ven
dor). I contacted them and did not recieve any feedback. I can't rate a DVD I have never seen. I didn't bother to se
nd it back because it would have cost me more that I orginally paid for it. In the future I will watch for the name
of the person and/or persons I am buying from. I thought they were a good company. I understand a simple mistake bu
t, to not get a response at all is not good businees sense. I spend hundreds of dollars a month on <url> building my
DVD collection. I guess I will be more careful in the future."
```

* A user on Kaggle pointed out that the dataset has a substantial amount of urls, so to increase accuracy, we introduce a url token <url>

Training / Test Sets

- We split the dataset which is a set of strings into a sentence list and a label list (0/1 for negative/positive).
- Data preprocessing:
  a. Tokenizing
  b. Removing stop words (the optional step)
  c. Stemming

```
nltk.download('stopwords')
nltk.download('punkt')
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
for i in range(len(dataset)):
    words = nltk.tokenize.wordpunct_tokenize(dataset[i].lower())
    if IGNORE == 'NO':
        words = [word for word in words if word not in stop_words]
    words = [stemmer.stem(word) for word in words]
    dataset[i] = words
```

Our Approach (Pseudocode)

- Initialize the classifier with an empty vocabulary, empty dictionaries for class probabilities, word counts, and class-word counts.
- Fit the classifier by taking in the training dataset and labels. Split the dataset into a train and test set. Calculate class probabilities, word counts, and class-word counts for each class in the train set.
- Predict the class labels for the given dataset by calculating scores for each class based on the probabilities of each class, word counts for each word, and class-word counts.
- Predict the class label for a single sentence by calculating scores for each class based on the probabilities of each class, word counts for each word, and class-word counts for the given sentence.
- Print the predicted class label and the probabilities of being positive or negative.

Our Approach

```
split_idx = int(0.8 * len(dataset))
test_data, test_labels = dataset[split_idx:], dataset_labels[split_idx:]
test_predictions = clf.predict(test_data)

tp, tn, fp, fn = 0, 0, 0, 0
for true_label, pred_label in zip(test_labels, test_predictions):
    if true_label == 1 and pred_label == 1:
        tp += 1
    elif true_label == 0 and pred_label == 0:
        tn += 1
    elif true_label == 1 and pred_label == 0:
        fn += 1
    elif true_label == 0 and pred_label == 1:
        fp += 1

sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)
precision = tp / (tp + fp)
npv = tn / (tn + fn)
accuracy = (tp + tn) / (tp + tn + fp + fn)
f_score = 2 * precision * sensitivity / (precision + sensitivity)
```

Evaluation

Our testing results:

With all steps

Test results/metrics:

Metric	Value
True positives	9585
True negatives	6247
False positives	1685
False negatives	2483
Sensitivity (recall)	0.79
Specificity	0.79
Precision	0.85
Negative predictive value	0.72
Accuracy	0.79
F-score	0.82

Keeping stop words

Test results/metrics:

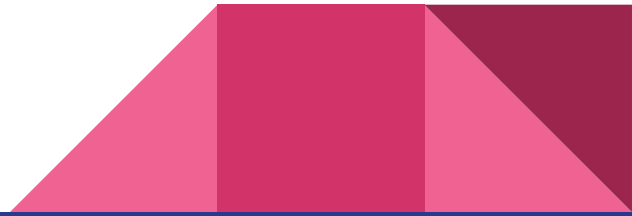
Metric	Value
True positives	9455
True negatives	6411
False positives	1521
False negatives	2613
Sensitivity (recall)	0.78
Specificity	0.81
Precision	0.86
Negative predictive value	0.71
Accuracy	0.79
F-score	0.82

Demo

Summary / Comments

- Naive Bayes classifier implemented for sentiment analysis on a dataset of 4,000,000 Amazon product reviews
- Bag-of-words model used with add-1 smoothing and binary representation
- Dataset split into 80% training and 20% testing sets
- Classifier correctly classified sentiment of reviews almost 80% of the time, regardless of stop words removal
- Bag-of-words model represents text data as a set of unique words with frequency recorded
- Add-1 smoothing prevents overfitting by adding small constant value to word probability
- Classifier trained on 80,000 reviews and tested on 20,000 reviews
- Skipping stop words removal degrades accuracy slightly due to large dataset with ample training data

Questions?



References

- Amazon Reviews for Sentiment Analysis https://www.kaggle.com/datasets/bittlingmayer/amazonreviews
- IIT CS 481 Lecture Slides