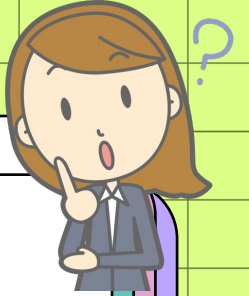


What Do Students Think About You? (Sentiment Analysis)

Nur Syuhada Azhar
Nursyaza Nisa Arfarizal
Syakirah Hanim Zulkernain
Thiveya Mahendran

Introduction

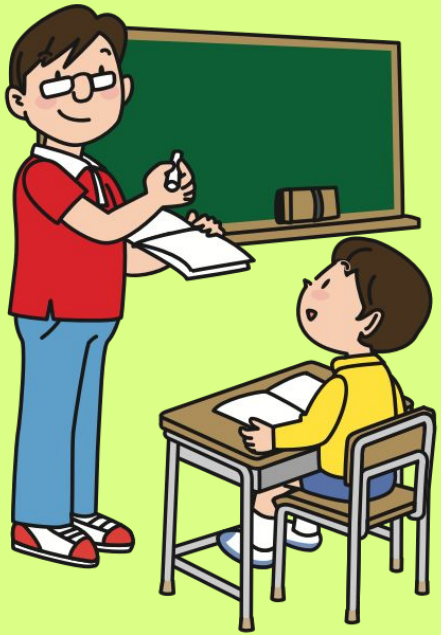
- **Dataset**:
  - Collect feedback from students of our course, **BITI**
- **Subjects**:

  BITI3413 Natural Language Processing

  BITI3533 Artificial Intelligence Project Management

  BITI3523 Artificial Intelligence in Robotics and Automation

  BLLW 3162 English for Professional Interaction
- **Target users:**
  - Lecturer
  - Student



- **A web-based application to allow lecturer to analyse feedback from students using sentiment analysis.**
  - Pie chart
  - Word cloud

Analysis of the developed system

- Data is the feedback collected from students.
- Sentiment analysis performed by labelling feedback into positive and negative feedbacks,
- Feature extraction by converting it to lowercase, stemming and remove stopwords.
- Data pre-processed by natural language processing techniques :
    1. **Term Frequency-Inverse Document Frequency (TF-IDF) :** identify the most important words and phrases that the collected feedback consists of and rare words have high scores, common words have low scores.
- TF-IDF resulted in higher accuracy.

Implementation

In [1]:
```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from math import log, sqrt
import pandas as pd
import numpy as np
import re
```

In [2]:
```
review = pd.read_csv("AIPM.csv", encoding = 'latin-1')
review.rename(columns = {'v1': 'labels', 'v2': 'review'}, inplace = True)
#print(tweet.head())
print(review['labels'].value_counts())
review['label'] = review['labels'].map({'negative': 0, 'positive': 1})
review.drop(['labels'], axis = 1, inplace = True)
print(review.head())
```

```
negative    65
positive    35
Name: labels, dtype: int64
                                review  label
0           Good on delivery method      1
1         I don't like this subject      0
2            Presentation non stop       0
3                  too many slides      0
4  The lecturer is okay in teaching      1
```

Implementation

In [3]:
```
totalReview = 65 + 35
trainIndex, testIndex = list(), list()
for i in range(review.shape[0]):
    if np.random.uniform(0, 1) < 0.75:
        trainIndex += [i]
    else:
        testIndex += [i]
trainData = review.loc[trainIndex]
testData = review.loc[testIndex]
```

In [4]:
```
trainData.reset_index(inplace = True)
trainData.drop(['index'], axis = 1, inplace = True)
print(trainData.head())
print(trainData['label'].value_counts())

testData.reset_index(inplace = True)
testData.drop(['index'], axis = 1, inplace = True)
print(testData.head())
print(testData['label'].value_counts())
```

```
                                   review  label
0                    Good on delivery method      1
1                      Presentation non stop      0
2                                too many slides      0
3            The lecturer is okay in teaching      1
4  The lecturer needs to have more intonation      0
0    48
1    26
Name: label, dtype: int64
                                          review  label
0                          I don't like this subject      0
1                honestly i dont really understand      0
2                   the lecturer is very reponsive      1
3          why do i need to study project management      0
4  Presentation! Presentation! Presentation! I do...      0
0    17
1     9
Name: label, dtype: int64
```

Implementation

In [6]:
```
# word cloud
positive_words = ' '.join(list(review[review['label'] == 1]['review']))
positive_wc = WordCloud(width = 512,height = 512).generate(positive_words)
plt.figure(figsize = (10, 8), facecolor = 'k')
plt.imshow(positive_wc)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

Implementation

```
In [5]: # word cloud
negative_words = ' '.join(list(review[review['label'] == 1]['review']))
negative_wc = WordCloud(width = 512,height = 512).generate(negative_words)
plt.figure(figsize = (10, 8), facecolor = 'k')
plt.imshow(negative_wc)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

Implementation

In [7]:
```
#Pre-process data

def process_message(message, lower_case = True, stem = True, stop_words = True, gram = 2):
    if lower_case:
        message = message.lower()
    words = word_tokenize(message)
    words = [w for w in words if len(w) > 2]
    if gram > 1:
        w = []
        for i in range(len(words) - gram + 1):
            w += [' '.join(words[i:i + gram])]
        return w
    if stop_words:
        sw = stopwords.words('english')
        words = [word for word in words if word not in sw]
    if stem:
        stemmer = PorterStemmer()
        words = [stemmer.stem(word) for word in words]
    return words
```

In [8]:
```
class ReviewClassifier(object):
    def __init__(self, trainData, method = 'tf-idf'):
        self.review, self.labels = trainData['review'], trainData['label']
        self.method = method

    def train(self):
        self.calc_TF_and_IDF()
        if self.method == 'tf-idf':
            self.calc_TF_IDF()    #TF-IDF

    def calc_prob(self):
        self.prob_positive = dict()
        self.prob_negative = dict()
        for word in self.tf_positive:
            self.prob_positive[word] = (self.tf_positive[word] + 1) / (self.positive_words + \
                                        len(list(self.tf_positive.keys())))
        for word in self.tf_negative:
            self.prob_negative[word] = (self.tf_negative[word] + 1) / (self.negative_words + \
                                        len(list(self.tf_negative.keys())))
        self.prob_positive_review, self.prob_negative_review = self.positive_review / self.total_review, self.negative_review /

    def calc_TF_and_IDF(self):
        noOfReview = self.review.shape[0]
        self.positive_review, self.negative_review = self.labels.value_counts()[1], self.labels.value_counts()[0]
        self.total_review = self.positive_review + self.negative_review
        self.positive_words = 0
        self.negative_words = 0
        self.tf_positive = dict()
        self.tf_negative = dict()
        self.idf_positive = dict()
        self.idf_negative = dict()
        for i in range(noOfReview):
            message_processed = process_message(self.review[i])
            count = list() #To keep track of whether the word has ocured in the message or not.
                           #For IDF
            for word in message_processed:
                if self.labels[i]:
                    self.tf_positive[word] = self.tf_positive.get(word, 0) + 1
                    self.positive_words += 1
                else:
                    self.tf_negative[word] = self.tf_negative.get(word, 0) + 1
                    self.negative_words += 1
                if word not in count:
                    count += [word]
```

Implementation

In [9]:
```
#Evaluation Metric

def metrics(labels, predictions):
    true_pos, true_neg, false_pos, false_neg = 0, 0, 0, 0
    for i in range(len(labels)):
        true_pos += int(labels[i] == 1 and predictions[i] == 1)
        true_neg += int(labels[i] == 0 and predictions[i] == 0)
        false_pos += int(labels[i] == 0 and predictions[i] == 1)
        false_neg += int(labels[i] == 1 and predictions[i] == 0)
    precision = true_pos / (true_pos + false_pos)
    recall = true_pos / (true_pos + false_neg)
    Fscore = 2 * precision * recall / (precision + recall)
    accuracy = (true_pos + true_neg) / (true_pos + true_neg + false_pos + false_neg)

    print("Precision: ", precision)
    print("Recall: ", recall)
    print("F1-score: ", Fscore)
    print("Accuracy: ", accuracy)
```

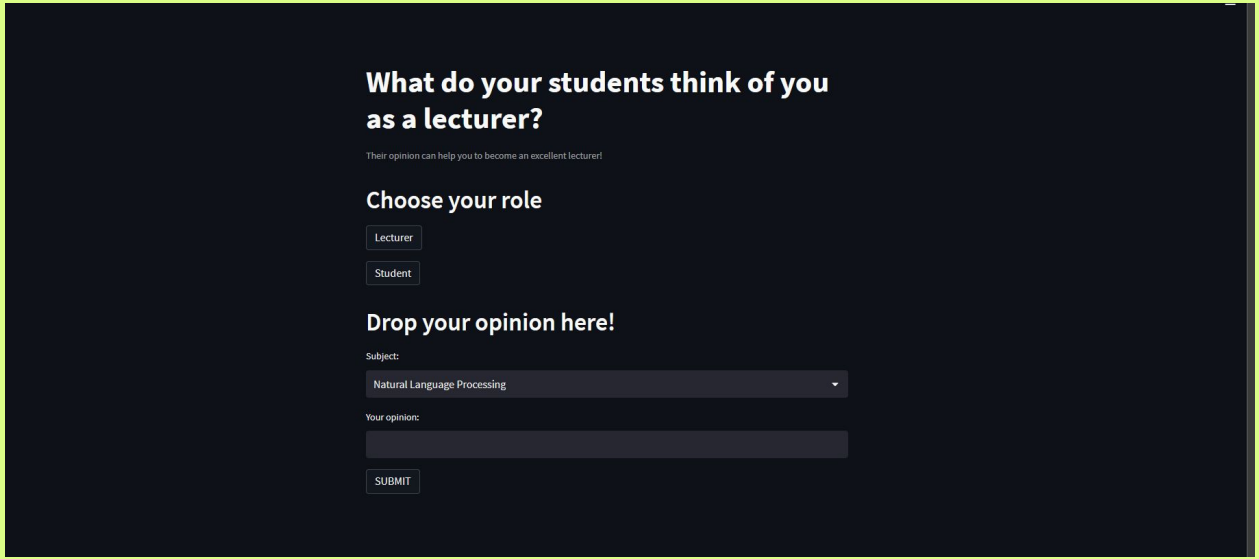
In [10]:
```
sc_tf_idf = ReviewClassifier(trainData, 'tf-idf')
sc_tf_idf.train()
preds_tf_idf = sc_tf_idf.predict(testData['review'])
metrics(testData['label'], preds_tf_idf)
```

```
Precision:  0.47368421052631576
Recall:  1.0
F1-score:  0.6428571428571429
Accuracy:  0.6153846153846154
```

Design

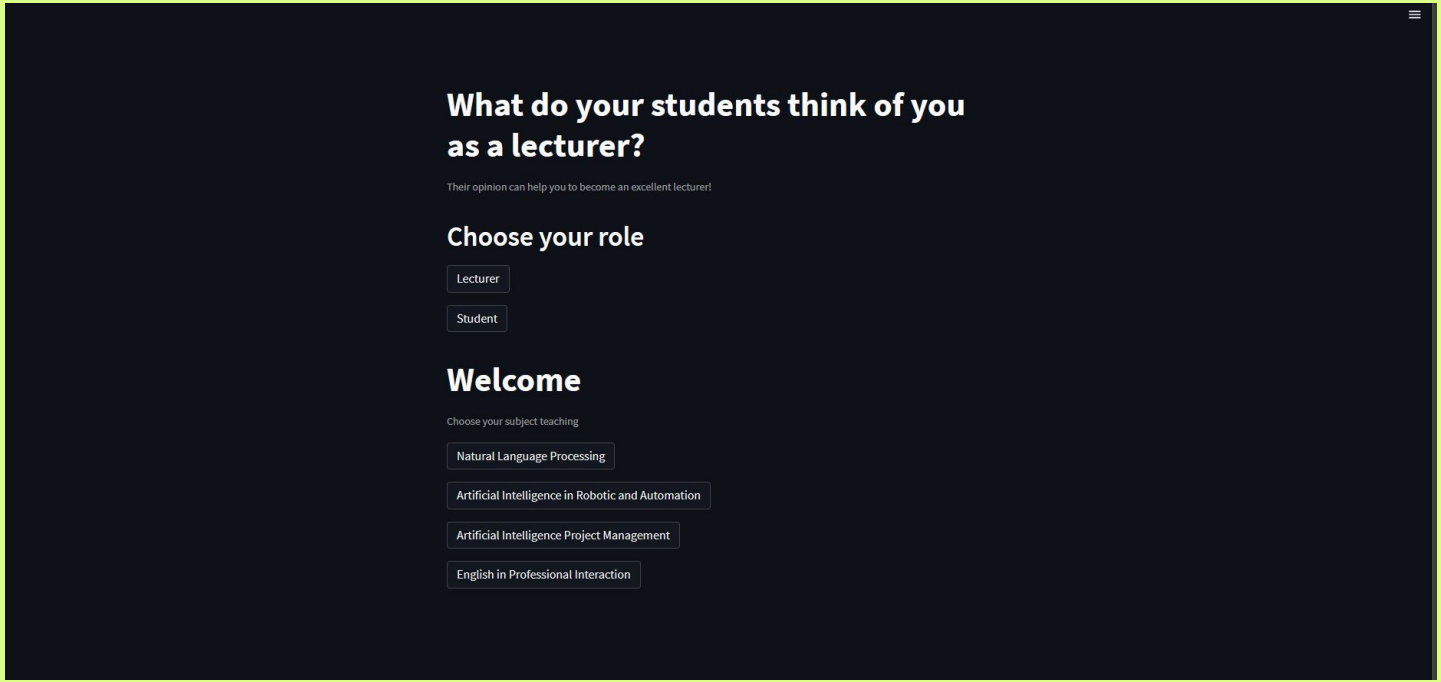
★ **Streamlit** was utilized for the design of the system by implementing Python codes.

★ User interface of the student main page



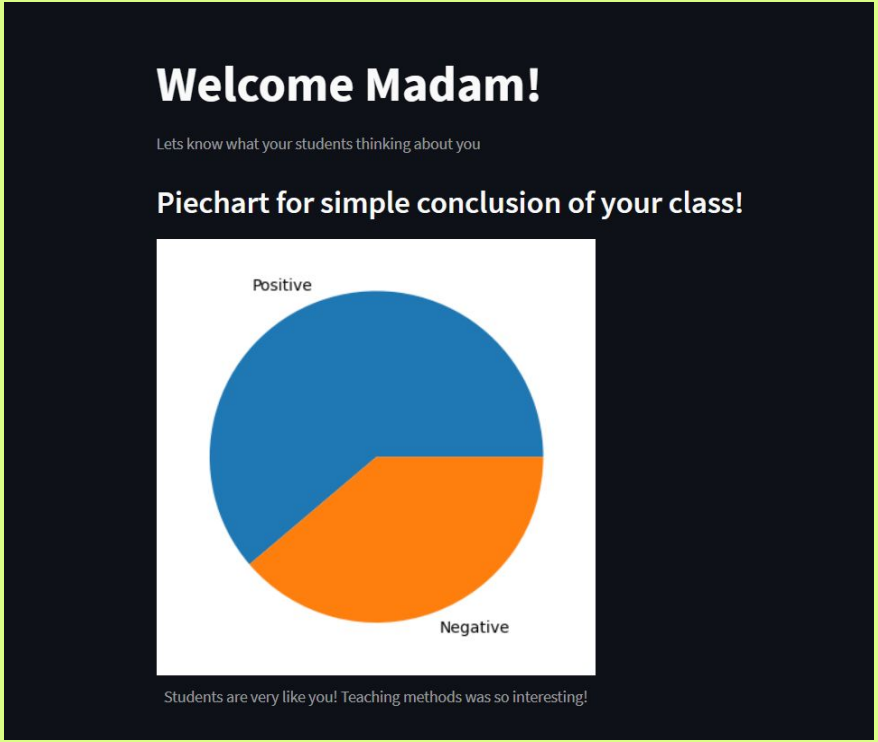
Design

★ User interface of the lecturer main page



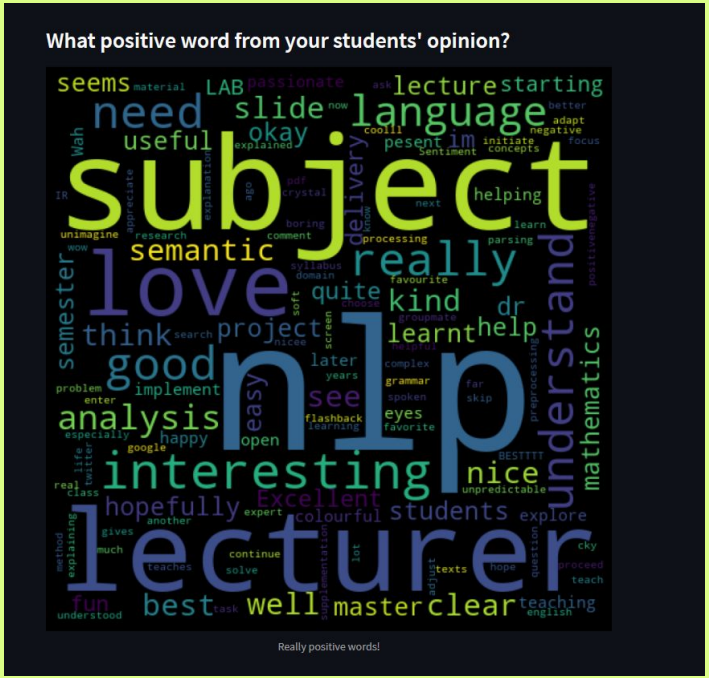
Design

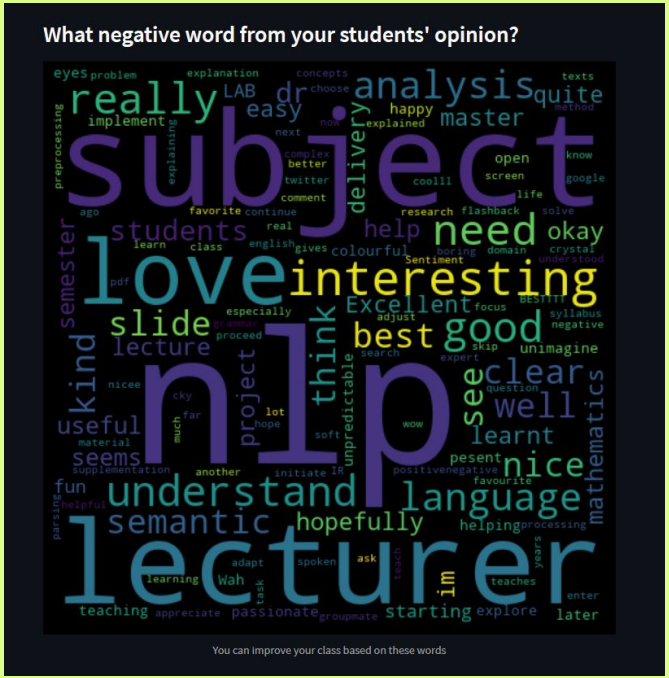
★ User interface of the lecturer sub-page containing pie chart which shows information on the positive and negative feedback percentage.



Design

★ User interface of the lecturer sub-page containing positive and negative word clouds.





Conclusion

Strengths:

- Sentiment Analysis
  - Analyze the feedbacks received from the student
- Data Visualization
  - Display pie chart using streamlit data visualization

Suggestions:

- Improve the design
  - Makes it more interesting and user friendly
- Include prediction module
  - Gives a valuable insight
- Include recommendation module
  - Gives suggestions

Conclusion

Reflection:

- Sentiment Analysis is very important in NLP
  - To monitor the performance
  - To categorize the opinions expressed
  - To build data visualization with this text data

Contribution:

- Potential to be promoted at school, college or university
  - To improve the performance of the lecturer
  - To provide a method of facilitating development

System Demo!