1. Problem Statement

These days, we may have ended up with more than 1000 languages around the world. There is no limitation to the number of languages people can learn. However, they do have a limitation of time. It is highly impossible for a person to understand all of the languages around the world. Cardinal Giuseppe Mezzofanti (1774-1849) was recorded as the most impressive language learner. Giuseppe Mezzofanti is fluent in talking in at least 30 languages and learning another 42 languages (Okrent, 2013).

When we see a product review, we can see that sometimes the review will be based on multiple languages. Using natural language processing (NLP), we can categorize the review based on positive or negative context.

In these cases, an agent of NLP to solve language barrier problems to understand comment/review context. The agent will learn from a dataset with 81 languages words, including chinses characters. This agent will determine if the words are positive or negative regardless of the language. However, in these cases, the agent will only see it by words, not sentences. This will help to create a faster learning process to spend less time learning sentences.

2. Selection of AI technique

On solving this problem, we will implement NLP or Natural Language Processing. Later on, we will combine classification techniques including Decision Tree, Support Vector Machine, or Naïve Bayes to do the classification. In solving this problem, the AI technique will learn 90 % of the datasets and will use 10% of the datasets to test the accuracy.

The datasets are found on Kaggle; these datasets are separated by multiple countries. To compile the datasets, there is a file in the "text_data" folder called "Compile Negative and Positive Words.bat." This file can be used to compile the positive and negative words into text regardless of the country.

All of the classifications AI techniques will be tried at least five times in order to find the best technique for this specific problem. There will be ranked based on the accuracy train and accuracy test. After testing five times for each technique, average accuracy will be used to determine the AI technique for solving this problem

		DT Accuracy	SVM Accuracy	Naïve Bayes Accuracy
#Test01	Train	62.07 %	62.07 %	62.14 %
	Test	62.04 %	62.09 %	61.47 %
#Test02	Train	62.08 %	62.07 %	62.02 %
	Test	61.97 %	62.05 %	62.53 %
#Test03	Train	62.05 %	62.08 %	62.14 %
	Test	62.2 %	62.01 %	61.44 %
#Test04	Train	62.05 %	62.17 %	62.09 %
	Test	62.29 %	61.16 %	61.88 %
#Test05	Train	62.06 %	62.09 %	62 %
	Test	62.18 %	61.86 %	62.68 %
Total	Train	310.31 %	310.48 %	310.39 %
	Test	310.68 %	309.17 %	310 %
Average	Train	62.06%	62.10%	62.08%
	Test	62.14%	61.83%	62.00%
Total Train + Test		620.99%	619.65%	620.39%

As we can see from the test result, Decision Tree (DT), Support Vector Machine (SVM), Naïve Bayes algorithm results are close to each other. In these cases, we can conclude that everything is acceptable to be used. Every algorithm has less than a 1% difference, and all of the accuracies for train and test are around 61 to 62 %.

Based on the five tests, we can see that; Naïve Bayes holds the highest record on testing accuracy, shown on Test05. On the other hand, based on Test04, SVM hold the highest record for training accuracy. However, the highest total after combining Train and Test accuracy DT hold the highest percentage. Therefore, I will use the Decision Tree classification algorithm on solving this problem

3. Explanation of AI technique

Natural Language Processing or NLP refers to the ability that is given to a computer to understand a text or spoken words in the same way as a human did. NLP can be used to understand the full meaning of human intent and sentiment from the given text or speech. We have used NLP to help our daily activities. Some of the famous implementations of NLP are Chatbots, speech-to-text dictation, digital assistants (Alexa, Siri, etc.) (IBM Cloud Education, 2020).

A Decision Tree is a Supervised Machine Learning with a continuous split based on the data parameter. Decision Trees consist of Nodes, Edges, and Leaf nodes. Nodes are the test value for a specific attribute. Edges are the connection to the next node or leaf. At the same time, Leaf nodes are the end of the nodes which will be used to predict the outcome. Decision Trees can be used in Classification Trees or Regression Trees. Decision Tree Classification has nodes and leaf nodes output of Yes/No or Boolean types (Chakure, 2019).

In order to solve this problem, NLP is used to process text from the user. After receiving the text, the text that the user input will be considered as an input for the Decision Tree. Continuing to that, the input will be used in the Decision Tree Classification technique to receive True or False output. This output will be used to consider the positive or negative words.

4. Solution Result

As we can see from the result, the Decision Tree Classification technique holds the best accuracy for both tests and training. This can be concluded with a total of nearly 621 % from training and testing accuracy after doing five tests. However, the other algorithm can also be used in order to solve this problem. Moreover, SVM holds the highest percentage of training accuracy.

Based on the five tests conducted, all the algorithms have an accuracy of around 61-62%. This may occur due to multiple languages that have the same words on the datasets. Moreover, the datasets are not big enough for 81 languages. To get a higher percentage, we can add more datasets into the learning files.

However, this will also increase the loading time of the program and the chance of having multiple words that have different meanings based on different languages. The other option that we can do is teach the agent in 1 language and translate the input to the specific language. This may have better accuracy; however, it will face the same problem as one word may have a different meaning in multiple languages. Moreover, this technique will also be determined by translation accuracy.

5. References

Chakure, A. (2019, July 6). *Decision tree classification*. Medium. https://medium.com/swlh/decision-tree-classification-de64fc4d5aac

IBM Cloud Education. (2020, July 2). *What is natural language processing?* IBM. https://www.ibm.com/cloud/learn/natural-language-processing

Okrent, A. (2013, February 27). *How many languages is it possible to know?* Mental Floss. https://www.mentalfloss.com/article/49138/how-many-languages-it-possible-know

Tatman, R. (2017). *Sentiment lexicons for 81 languages*. Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/rtatman/sentiment-lexicons-for-81-languages

6. Screenshot

a. Decision Tree Classification

#Test01

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.07481415718128
         Test Accuracy : 62.045319022063204
```

#Test02

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.0821021108564
         Test Accuracy : 61.97376267143709
```

#Test03

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.056925543615094
         Test Accuracy : 62.200357781753134
```

#Test04

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.046987424967206
         Test Accuracy : 62.28980322003578
```

#Test05

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.05957570858786
         Test Accuracy : 62.17650566487776
```

b. Naïve Bayes

#Test01

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.13775557528456
         Test Accuracy : 61.47286821705427
```

#Test02

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.02048577523951
         Test Accuracy : 62.5283243887895
```

#Test03

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.142393363986905
         Test Accuracy : 61.43709004174121
```

#Test04

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.09270277074748
         Test Accuracy : 61.88431723315444
```

#Test05

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.0039222441597
         Test Accuracy : 62.67740011926058
```

c. Support Vector Machine

#Test01

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.0734890746949
         Test Accuracy : 62.05128205128205
```

#Test02

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.078126863397245
         Test Accuracy : 62.00954084675014
```

#Test03

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.17220771993056
         Test Accuracy : 61.16279069767442
```

#Test04

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.094690394477055
         Test Accuracy : 61.86046511627907
```

#Test05

```
In [23]: print("Train Accuracy : {}".format(accuracy_train))
         print("Test Accuracy : {}".format(accuracy_test))

         Train Accuracy : 62.069513827235745
         Test Accuracy : 62.08706022659511
```