**Learning Styles Classification**Using natural language processing

**Diagram, venn diagram

Description automatically generated**

**Name: Rami Ghanem**

**Supervision: Mona Qasem**

**Introduction**

Nowadays, many schools use the learning styles of their student to improve academic outcomes. There are various learning styles, with the three most common being visual, auditory, and kinesthetic. The traditional method for identifying learning styles involves administering a questionnaire. However, due to the large number of questions, many students may select answers randomly, which can lead to inaccurate results.

In This project, I will solve this problem by developing a natural language processing model that predicts learning styles using sentences written by the students. After training this model the school will ask the student to write an essay about a specific topic. and this text will be Prepared and input it to the model, and the output will be the learning styles of the student.

**Background and Literature:**

One study explored this topic in 2015 In this study, the authors developed a classification system for learning styles based on Kolb's Learning Styles Inventory (LSI). They used natural language processing techniques to analyze the responses of students to the LSI questionnaire and found that the classification system was able to accurately predict the learning styles of the students.

another study in 2016 also used natural language processing to classify learning styles based on the Felder-Solomon Index of Learning Styles (ILS). They found that the classification system was able to accurately predict the learning styles of the students.

**Design:**

The aim of this project is to accurately predict the learning styles of a student based on analysis of their written sentences using deep learning and natural language processing.

The dataset consists of 15,451 observations with 2 variables **Feature:** a column containing English sentences and **Target:** a column indicating the learning style of the corresponding English sentence (either visual, auditory, or kinesthetic). in order to get the final model, I followed these steps:

* **Data Preprocessing:**
  + removing duplicated values :
    - number of samples Before :

|  |  |
| --- | --- |
| **Type** | **Sentence** |
| Visual | 5827 |
| Kinesthetic | 4819 |
| Auditory | 4804 |

* + - number of samples After:

|  |  |
| --- | --- |
| **Type** | **Sentence** |
| Visual | 5527 |
| Kinesthetic | 4572 |
| Auditory | 4496 |

* + convert the target column into numeric form using One Hot Encoding technique
    - Here is an example of one hot encoding for the three classes "visual," "auditory," and "kinesthetic":

|  |  |  |  |
| --- | --- | --- | --- |
| **Original Data** | **Visual** | **Auditory** | **Kinesthetic** |
| Visual | 1 | 0 | 0 |
| Auditory | 0 | 1 | 0 |
| Kinesthetic | 0 | 0 | 1 |
| Auditory | 0 | 1 | 0 |
| Visual | 1 | 0 | 0 |

In this example, the original data contains three categorical values: "visual," "auditory," and "kinesthetic." One hot encoding creates three new columns, one for each possible value. The values in the new columns are binary, with a 1 indicating the presence of that value in the original data and a 0 indicating its absence.

For example, the first row of the original data contains the value "visual," so the value in the "visual" column is 1 and the values in the other two columns are 0. The second row contains the value "auditory," so the value in the "auditory" column is 1 and the values in the other two columns are 0, and so on.

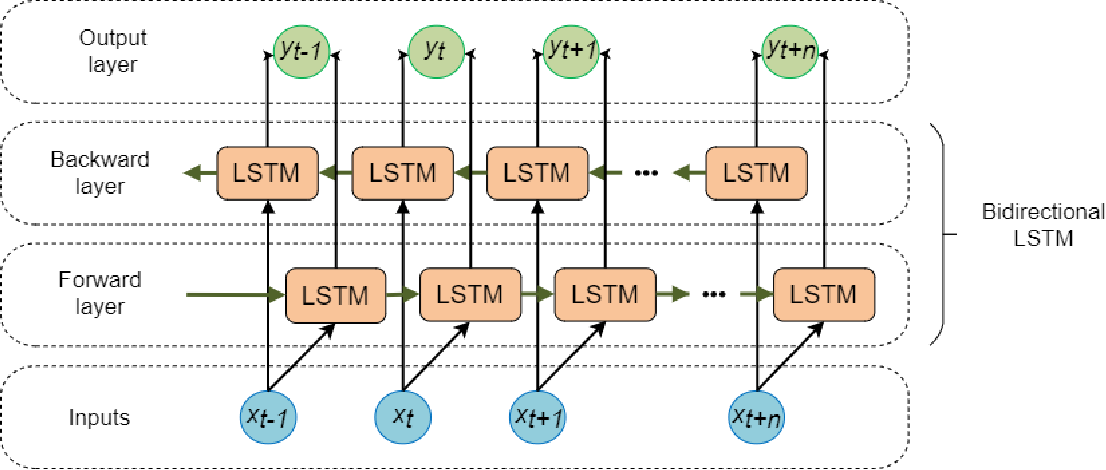
* + Convert the entire text into lower case
  + removing stop words using spacy library
  + removing digits
  + Removes all punctuation

**Example:**

**Original Sentence:** The quick brown fox JUMPS OVER the lazy dog. 123

**After text preprocessing:** quick brown fox jumps lazy dog

* **Exploratory Data Analysis:** I employed term frequency analysis to identify the most commonly occurring words in each class sentence.
* **Model Selection**: I build two models Doc2Vec and Word Embedding in both models, I used these as part of the architecture:
  + **bidirectional long short-term memory (LSTM) layer :** it is a type of recurrent neural network (RNN) layer that processes input data in both forward and backward directions. An LSTM layer processes input data sequentially, using a memory cell and three gates (input, forget, and output) to control the flow of information into and out of the cell. A bidirectional LSTM layer processes the input data in both forward and backward directions, using two separate LSTM layers for each direction. The outputs from the two LSTM layers are then concatenated, allowing the model to capture contextual information from both past and future time steps.



* + **GlobalMaxPool1D Layer :** it is a special case of max pooling where the window size is the entire spatial dimension of the input, such that the output of the pooling layer is a tensor with a single element. For example, suppose we have an input tensor with shape (batch\_size, time\_steps, features), and we apply GlobalMaxPool1D to this tensor. The output tensor will have the shape (batch\_size, features), where each element in the output is the maximum value of the corresponding feature over all time steps.
  + **Dropout Layer :** It is a regularization technique that randomly sets a fraction of the input units to zero during training, which helps to prevent overfitting and improve the generalization ability of the model.
  + **Dense Layer :** it is a fully connected layer in a neural network, which means that all the neurons in the layer are connected to all the neurons in the preceding and following layers.
  + **Early Stopping Callbacks:** it is a technique used to prevent overfitting in deep learning models. It involves monitoring the performance of the model on a validation set during training and stopping the training process when the performance stops improving.
  + **Categorical cross entropy Loos Function:** it is a loss function used for classification tasks with multiple classes, and it is a measure of the difference between the true distribution of the classes and the predicted distribution of the classes. The output label is assigned a one-hot category encoding value in form of 0s and 1.
  + **Adam optimizer:** it is an alternative optimization algorithm that provides more efficient neural network weights by running repeated cycles of “adaptive moment estimation.” Adam extends on stochastic gradient descent to solve non-convex problems faster while using fewer resources than many other optimization programs.
  + **Batch size:** is the number of samples processed by the model in a single forward/backward pass. The batch size is a hyperparameter that can have a significant impact on the performance and efficiency of the model.

The difference between these models is how we deal with text data:

* **Word Embedding model:** Word embedding is a method of representing words in a continuous, numerical space such that the relationships between words are preserved.

it is a pre-trained model which learned from a large dataset of text using unsupervised learning techniques. The resulting embeddings are usually organized into a vocabulary of words, where each word is represented by a fixed-length vector. The vectors are chosen such that semantically similar words are close to each other in the vector space, while dissimilar words are farther apart.

**Steps:**

1. **Preprocess the text data:** This typically involves tokenizing the text into individual words or tokens
2. **use texts to sequences function:** this is a function in TensorFlow that is used to convert a list of texts into a list of sequences of integers
3. **Use pad sequence’s function:** this is a function in TensorFlow that is used to pad sequences of data to a fixed length. The pad sequence’s function allows you to do this by adding padding elements (usually zeros) to the shorter sequences so that they all have the same length

**Example:**

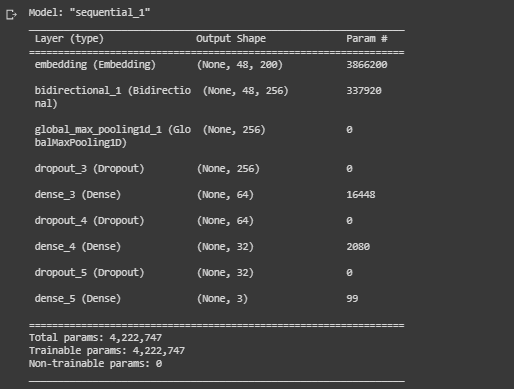
sequences = [[1, 2, 3], [4, 5], [6, 7, 8, 9]] pad\_sequences(sequences, maxlen=4)

The output of this code will be a 2D array of padded sequences, with each sequence having a length of 4:

[[1 2 3 0] [4 5 0 0] [6 7 8 9]]

1. using the numerical text data as an input for the model

**model architecture :**

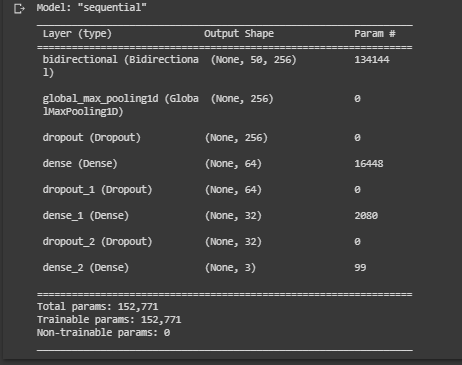
****

* **Doc2Vec model**: unlike word embeddings, which map individual words to vectors, Doc2Vec maps entire documents to vectors. The main advantage of using Doc2Vec is that it allows you to capture the meaning and context of a whole document, rather than just individual words. This can be useful for tasks such as document classification, information retrieval, and semantic similarity, where the context and meaning of a document as a whole are important.

**Steps:**

1. **Preprocess the text data:** This typically involves tokenizing the text into individual words or tokens
2. **Create a list of TaggedDocument objects:** TaggedDocument is a class in gensim that represents a document as a list of words with an associated label or tag. to create a list of TaggedDocument objects we need to iterate over the preprocessed text data and create a TaggedDocument for each document.
3. **Train the Doc2Vec model:** I use the Doc2Vec class in gensim to train a Doc2Vec model on the list of TaggedDocument objects.
4. **Extract the Doc2Vec embeddings:** use the docvecs attribute to get the trained Doc2Vec embeddings for each document

**model architecture :**



* **Evaluation:** To evaluate the performance of the models, I applied the testing data to the models and compared the predictions with the actual targets using the following metrics:
  + **Accuracy:** is defined as the percentage of correct predictions for the test data.
  + **Precision:** is defined as the fraction of relevant examples (true positives) among all the examples which were predicted to belong in a certain class.
  + **Recall:** is defined as the fraction of examples that were predicted to belong to a class with respect to all the examples that truly belong in the class.
  + **F-score:** is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model’s precision and recall.

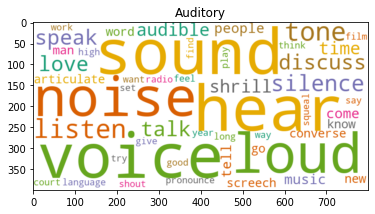
**Results**

Exploration Data Analysis :

I looked at the most frequent words in each class sentences and this is the results :

1. Auditory class:

[('sound', 490), ('hear', 469), ('voice', 397), ('noise', 345), ('loud', 337), ('listen', 325), ('tone', 318), ('silence', 314), ('discuss', 305), ('love', 263), ('speak', 244), ('talk', 217), ('like', 196), ('audible', 173), ('shrill', 168), ('time', 164), ('people', 145), ('tell', 128), ('come', 126), ('music', 125), ('articulate', 123), ('word', 120), ('screech', 110), ('know', 109), ('new', 107), ('converse', 99), ('man', 96), ('go', 95), ('pronounce', 89), ('say', 87), ('language', 87), ('shout', 85), ('think', 81), ('work', 80), ('want', 79), ('radio', 79), ('long', 77), ('find', 74), ('good', 74), ('year', 73), ('way', 72), ('give', 72), ('court', 71), ('high', 69), ('film', 68), ('try', 67), ('play', 66), ('set', 64), ('squeal', 64), ('feel', 62)]



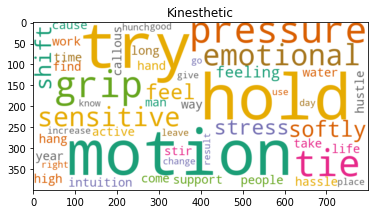
1. Visual class:

[('idea', 329), ('vision', 327), ('focus', 321), ('dream', 320), ('picture', 318), ('horizon', 316), ('notice', 311), ('imagine', 308), ('observe', 308), ('obvious', 305), ('illusion', 305), ('clarity', 302), ('illustrate', 302), ('view', 237), ('look', 216), ('watch', 208), ('scope', 206), ('time', 204), ('inspect', 178), ('like', 173), ('give', 153), ('analyze', 151), ('work', 151), ('conspicuous', 147), ('people', 146), ('appear', 130), ('new', 126), ('sight', 124), ('come', 118), ('take', 118), ('create', 118), ('year', 117), ('way', 116), ('good', 113), ('world', 110), ('great', 104), ('use', 104), ('day', 103), ('long', 102), ('line', 99), ('life', 97), ('eye', 95), ('point', 95), ('go', 87), ('envision', 85), ('man', 84), ('light', 80), ('study', 77), ('say', 76), ('include', 74)]



1. Kinesthetic class:

[('try', 489), ('hold', 349), ('motion', 327), ('pressure', 322), ('grip', 310), ('tie', 309), ('emotional', 305), ('sensitive', 302), ('softly', 281), ('shift', 250), ('stress', 199), ('feel', 189), ('feeling', 187), ('support', 168), ('active', 166), ('time', 166), ('intuition', 154), ('hang', 144), ('like', 136), ('hassle', 123), ('people', 121), ('water', 116), ('high', 105), ('work', 99), ('take', 96), ('year', 95), ('stir', 94), ('life', 90), ('hand', 88), ('way', 87), ('hustle', 87), ('man', 85), ('find', 85), ('come', 83), ('callous', 83), ('long', 81), ('cause', 80), ('change', 79), ('go', 76), ('good', 75), ('know', 75), ('increase', 74), ('leave', 70), ('use', 69), ('place', 68), ('result', 67), ('give', 66), ('right', 66), ('hunch', 66), ('day', 65)]



Model Evaluation :

To check the performance of the models I applied the testing data to the models and compare the prediction with the real target (confusion matrix) and the result as below

1. Document To Vector model :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** |
| **Auditory** | 0.75 | 0.76 | 0.76 |
| **Kinesthetic** | 0.73 | 0.77 | 0.75 |
| **Visual** | 0.79 | 0.75 | 0.77 |
| **Overall Accuracy** |  |  | 0.76 |

1. Word Embedding model :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** |
| **Auditory** | 0.95 | 0.94 | 0.95 |
| **Kinesthetic** | 0.95 | 0.98 | 0.97 |
| **Visual** | 0.96 | 0.95 | 0.96 |
| **Overall Accuracy** |  |  | 0.96 |

**Conclusion**

by applying the different techniques to convert text to numeric values I discovered that word embedding yielded better results than Document-To-Vector for this case. Specifically, the overall accuracy of the word embedding model was 96%, which demonstrates its superiority in this context.

**future work**

In order to fully leverage the potential of this model, we need to incorporate additional features and build additional models. This will allow us to create a comprehensive system that can be utilized by schools to enhance student learning.