UTILITY FUNCTIONS

This are helper functions used all over the code.

```
def read_entire_dataset()
```

> **DESCRIPTION**
>
> > Loads the whole dataset into a dataframe with two columns, **Sentence** and **Output** (Label).
> >
> > **OUTPUT**
>
> - **dataframe** : A Panda dataframe containing the data

```
def word_extraction(sentence)
```

> **DESCRIPTION**
>
> > Takes in a sentence and removes stop words.
> >
> > **PARAMETER**
>
> - **sentence** : A single sentence
>
> > **OUTPUT**
>
> - **cleaned_text** : A List of all the words in the sentence

```
def clean_sentences(sentences)
```

> **DESCRIPTION**
>
> > Takes in a list of sentences and applies word_extraction on them.
> >
> > **PARAMETER**
>
> - **sentence** : A list of sentences.

> **OUTPUT**
>
> - **extracted_sentences** : A List of all the extracted sentences.

```
def build_label_dict()
```

> **DESCRIPTION**
>
> > Extracts all the labels from the dataset and gives each unique label a unique id (number) starting from 0.
> >
> > **OUTPUT**
>
> - **label_to_id** : A dictionary that maps each unique label to a unique id.

```
def create_embedding_matrix(embedding_dict, vocab_dict)
```

> **DESCRIPTION**
>
> > Creates the embedding matrix using the words in the vocabulary. The first row of the matrix contains zeros (used for padding). The unknown words in vocabulary (i.e words in the vocabulary but not found in the glove embeddings) use the "#UNK#" vector as their vectors in the embedding matrix.
> >
> > **PARAMETER**
>
> - **embedding_dict** : A dictionary of each word and it embedding vector.
> - **vocab_dict** : A dictionary of each word in the vocabulary and it unique id.
>
> > **OUTPUT**
>
> - **embedding_matrix** : A numpy array representing the matrix.

```
def make_bow_vector(sentence, embedding_matrix)
```

> **DESCRIPTION**
>
> > Creates the Bag of Word vector for a sentence.

> PARAMETER
>
> - **sentence** : The sentence representation i.e list of all the ids of the words in a sentence.
> - **embedding_matrix** : A numpy array representing the embedding matrix.
>
> OUTPUT
>
> - **sentence_as_bow** : A vector of the sentence embeddings generated by applying the bag of words formula.

DATA PREPROCESSING CLASSES

```
class PerformDataSplit()
```

> DESCRIPTION
>
> > Splits dataset into training and validation sets and saves the result as a csv file.

PerformDataSplit Class Public Methods

```
def load_splited_datasets(sentence, embedding_matrix)
```

> DESCRIPTION
>
> > Gets the partitioned datasets from the file system and load them into a dataframe.
>
> PARAMETER
>
> - **train_path** : Path to the training dataset in the file system.
> - **val_path** : Path to the validation dataset in the file system.
> - **test_path** : Path to the testing dataset in the filesystem.
>
> OUTPUT
>
> - **dataframe_tuples** : A tuple of the dataframes for the testing, training and validation datasets.

```
class Preprocess(dataset_sentence, dataset_labels, batch_size)
```

> **DESCRIPTION**
>
> > This class performs all the data preprocessing, it takes in the raw dataset and performs the following operations on it:
> >
> > - Cleans the sentences (removes stop words).
> > - Convert the words in the sentences to ids.
> > - Convert the labels to ids.
> > - Create the embedding matrix.
> > - Break dataset into batches based on the sentence lengths.
> > - Pad the sentences.
>
> **PARAMETER**

- **dataset_sentences** : The sentence column of the dataframe
- **dataset_labels** : The Label column of the dataframe
- **batch_size** : The number of sentences per batch

Preprocess Class Public Methods

```
def get_vocab_size()
```

> **DESCRIPTION**
>
> > Gets the size of the vocabulary.
>
> **OUTPUT**

- **vocab_size** : The number of unique words in the corpus.

```
def sentences_as_id()
```

> **DESCRIPTION**
>
> > Convert the words in each sentence in the dataset to ids for example the sentence ["I", "am", "here"] is replaced with [1, 2, 3].
>
> **OUTPUT**

- **sentence_repr** : A list containing all the sentences in the dataset as ids.

```
def labels_as_id()
```

> DESCRIPTION
>
> > Convert the labels for each sentence in the dataset to ids for example the sentence "DESC:manner" is replaced with 1.
>
> OUTPUT

- **labels_repr** : A list containing all the labels for each sentence in the dataset as ids.

```
def pad_sentence(batched_dataset)
```

> DESCRIPTION
>
> > Make each batch's sentences the same length by taking the length of the batch's longest sentence and padding all other sentences shorter than it with zero. This is done until the batch's sentences are of the same length.
>
> PARAMETER

- **batched_dataset** : A list of sentences in each batch from longest to shortest.

> OUTPUT

- **(padded_sentences, labels_, sentence_lengths)** : For each batch, it returns a tuple with the padded sentences and their labels, as well as a list with the original length of each sentence before padding was applied.

```
def create_batched_dataset()
```

> DESCRIPTION
>
> > Performs the batching process on the dataset
>
> OUTPUT

- **batch_datasets** : Returns a dictionary holding a mapping between each batch number and the batched items for that batch.

MODELS CLASSES

```
class BILSTMModel(vocab_size, embed_dim, hidden, n_label, n_layers, embedding_matri
```

DESCRIPTION

Contains the architecture for the Feed forward Neural Net Model which uses BILSTM.

PARAMETER

- **vocab_size** : The number of unique words in the corpus.
- **embed_dim** : The size of each embedding vector.
- **hidden** : The number of features in the hidden state h for LSTM.
- **n_label** : The number of categories (unique label) a sentence can belong to.
- **n_layers** : Number of recurrent layers. Two is used for BILSTM.
- **embedding_matrix** : A numpy array representing the embedding matrix.
- **random_init** : Set to True to generate random embeddings or False to use pretrained embeddings.
- **freeze** Set to False to finetune the embedding weights or True to Freeze weights.

```
class CNNModel(vocab_size, embed_dim, hidden = None , n_label, n_layers = None, emb
```

DESCRIPTION

Contains the architecture for the Convolutional Neural Network classifier.

PARAMETER

- **vocab_size** : The number of unique words in the corpus.
- **embed_dim** : The size of each embedding vector.
- **n_label** : The number of categories (unique label) a sentence can belong to.
- **embedding_matrix** : A numpy array representing the embedding matrix.

- **random_init** : Set to True to generate random embeddings or False to use pretrained embeddings.
- **freeze** Set to False to finetune the embedding weights or True to Freeze weights.

```
class FeedforwardNeuralNetModel(input_dim, hidden_dim, output_dim)
```

DESCRIPTION

Contains the architecture for the Feed forward Neural Net Model which uses Bag of Words.

PARAMETER

- **input_dim** : The size of each embedding vector.
- **hidden_dim** : The number of features in the hidden layers.
- **output_dim** : The number of categories (unique label) a sentence can belong to.

TRAINING, TESTING AND EVALUATION HELPER FUNCTIONS

```
def model_accuracy(predict, y)
```

DESCRIPTION

Computes the accuracy of the model.

PARAMETER

- **predict** : The list of predicted labels.
- **y** : The list of original labels.

OUTPUT

- **accuracy** : Returns the accuracy as float.

```
def train(model, train_dataset, optimizer, criterion)
```

> **DESCRIPTION**
>
> > Used by BILSTM and CNN models for training.
> >
> > **PARAMETER**
>
> - **model** : The instance of the model (BILSTMModel or CNNModel).
> - **train_dataset** : The batched training dataset.
> - **optimizer** : The optimizer to be used by the model for training.
> - **criterion** : The loss function to be used by the model for training.
>
> **OUTPUT**
>
> - **avg_total_loss, avg_total_acc** : Returns a tuple of the average loss and accuracy after training is done.

```
def evaluate(model, val_dataset, criterion)
```

> **DESCRIPTION**
>
> > Used by BILSTM and CNN models for validating.
> >
> > **PARAMETER**
>
> - **model** : The instance of the model (BILSTMModel or CNNModel).
> - **val_dataset** : The batched validation dataset.
> - **criterion** : The loss function to be used by the model for validation.
>
> **OUTPUT**
>
> - **avg_total_loss, avg_total_acc** : Returns a tuple of the average loss and accuracy after validation is done.

```
def train_bow(model, train_dataset, optimizer, criterion, embedding_matrix, dataset
```

> **DESCRIPTION**

> Used by Bag of Words models (FeedforwardNeuralNetModel) for training.

> PARAMETER

- **model** : The instance of the BOW model.
- **train_dataset** : The batched training dataset.
- **optimizer** : The optimizer to be used by the model for training.
- **criterion** : The loss function to be used by the model for training.
- **embedding_matrix** : A numpy array representing the embedding matrix using the pretrained embeddings or randomly initialised embeddings.
- **dataset_size** : The size of the training dataset.

> OUTPUT

- **avg_total_loss, avg_total_acc** : Returns a tuple of the average loss and accuracy after training is done.

```
def evaluate_bow(model, val_dataset, criterion, embedding_matrix, dataset_size)
```

> DESCRIPTION

> Used by Bag of Words models (FeedforwardNeuralNetModel) for validating.

> PARAMETER

- **model** : The instance of the BOW model.
- **val_dataset** : he batched validation dataset.
- **criterion** : The loss function to be used by the model for validation.
- **embedding_matrix** : A numpy array representing the embedding matrix using the pretrained embeddings or randomly initialised embeddings.
- **dataset_size** : The size of the validation dataset.

> OUTPUT

- **avg_total_loss, avg_total_acc** : Returns a tuple of the average loss and accuracy after validation is done.

```
def test_model(model, embedding_matrix, bow_model=False)
```

> **DESCRIPTION**
>
> > Tests the model using the test dataset loaded from the filesystem.
> >
> > **PARAMETER**
>
> - **model** : The instance of the BOW model (BILSTMModel, CNNModel or FeedforwardNeuralNetModel)
> - **embedding_matrix** : A numpy array representing the embedding matrix using the pretrained embeddings or randomly initialised embeddings.
> - **bow_model** : Set to True if the model to be tested is the BOW model otherwise False.
>
> > **OUTPUT**
>
> - **accuracy** : Returns the accuracy of the test as float.