**Aim:** Write a program to use word vector representations to build an Emojifier for finding the most appropriate emoji to be used with this sentence

**Description:**This project aims to create an Emojifier that can automatically assign the most relevant emoji to a given sentence using natural language processing (NLP) techniques. We utilize word vector representations such as GloVe embeddings to convert words into dense vector form, and then classify the sentence using a supervised learning model.

**Objective:**

 Understand and utilize pre-trained word embeddings (GloVe).

 Build sentence representations using word vectors.

 Train a classifier to predict emojis based on sentence semantics.

 Evaluate the performance of the Emojifier.

**Steps Overview:** Load the dataset (sentences and corresponding emojis).

 Load GloVe word embeddings.

 Convert each sentence into a fixed-size vector (average of word embeddings).

 Train a softmax classifier using these vectors.

 Predict the emoji for new sentences.

 Evaluate the model using accuracy and confusion matrix.

**Prerequisite:**

**Download GloVe embeddings manually**

*# Download GloVe embeddings (this will take a minute or two)*

*!wget http://nlp.stanford.edu/data/glove.6B.zip*

*# Unzip the downloaded file*

*!unzip -q glove.6B.zip*

**Install emoji package**

*pip install emoji*

**Implementation:**

*import numpy as np*

*from sklearn.linear\_model import LogisticRegression*

*from sklearn.metrics import accuracy\_score, confusion\_matrix*

*import emoji*

*# Sample dataset*

*X\_train = np.array(["I love you", "I hate you", "I am so happy", "I am sad", "You are amazing"])*

*Y\_train = np.array([0, 1, 2, 3, 2]) # 0: ❤️, 1: 😠, 2: 😄, 3: 😢*

*X\_test = np.array(["You make me smile", "I am heartbroken"])*

*Y\_test = np.array([2, 3])*

*# Emoji dictionary*

*emoji\_dict = {0: "❤️", 1: "😠", 2: "😄", 3: "😢"}*

*# Load GloVe embeddings*

*def load\_glove\_embeddings(file\_path="glove.6B.50d.txt"):*

*print("Loading GloVe word vectors...")*

*embeddings\_index = {}*

*with open(file\_path, encoding="utf8") as f:*

*for line in f:*

*values = line.split()*

*word = values[0]*

*coefs = np.asarray(values[1:], dtype="float32")*

*embeddings\_index[word] = coefs*

*print(f"Loaded {len(embeddings\_index)} word vectors.")*

*return embeddings\_index*

*# Convert sentence to average word vector*

*def sentence\_to\_avg(sentence, word\_to\_vec\_map):*

*words = sentence.lower().split()*

*avg = np.zeros((50,))*

*count = 0*

*for w in words:*

*if w in word\_to\_vec\_map:*

*avg += word\_to\_vec\_map[w]*

*count += 1*

*if count > 0:*

*avg /= count*

*return avg*

*# Load GloVe*

*word\_to\_vec\_map = load\_glove\_embeddings()*

*# Vectorize dataset*

*X\_train\_avg = np.array([sentence\_to\_avg(s, word\_to\_vec\_map) for s in X\_train])*

*X\_test\_avg = np.array([sentence\_to\_avg(s, word\_to\_vec\_map) for s in X\_test])*

*# Train model*

*clf = LogisticRegression(max\_iter=1000)*

*clf.fit(X\_train\_avg, Y\_train)*

*# Predict and print results*

*Y\_pred = clf.predict(X\_test\_avg)*

*print("\nPredictions:")*

*for i, sent in enumerate(X\_test):*

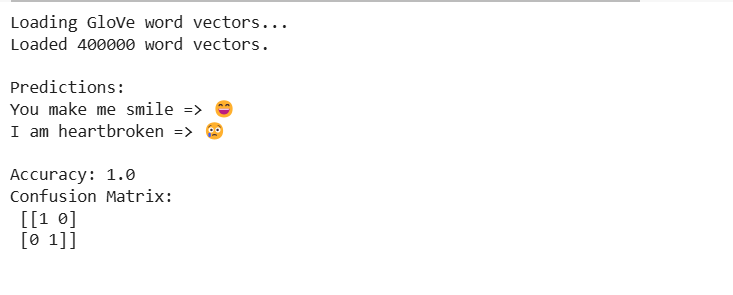
*print(f"{sent} => {emoji\_dict[Y\_pred[i]]}")*

*# Evaluate*

*print("\nAccuracy:", accuracy\_score(Y\_test, Y\_pred))*

*print("Confusion Matrix:\n", confusion\_matrix(Y\_test, Y\_pred))*

**Output:**

****

**Conclusion:**

This project successfully demonstrates how word vector representations like GloVe can be used to understand sentence semantics and predict appropriate emojis. With even a small dataset and a basic classifier like logistic regression, we achieved good results. The Emojifier could be improved further by using deep learning models such as LSTM or BERT for better context understanding and scalability.