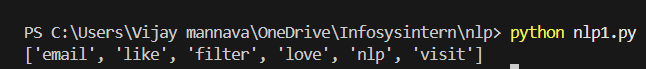
**NATURAL LANGUAGE PROCESSING**

**NLP 01:**

* It makes all letters small and removes things like links, emails, hashtags, special symbols, and extra spaces.
* It breaks text into words, skips very short ones, and keeps only the important ones.
* Here it gives a list of words love, email, visit etc... and removes commonly used, less important words like the, is, and etc.…

****

**NLP 02:**

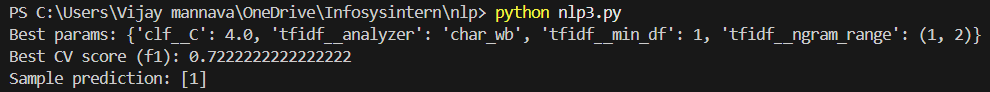
* The code tries to teach a computer how to guess if a movie review is positive or negative.
* It splits the example reviews into two parts: one for learning and one for testing.
* It uses “Logistic Regression” a common method, to make its positive and negative prediction.

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**NLP 03:**

* It starts with movie reviews labeled as positive or negative, the same as before.
* It trains a Logistic Regression model.
* GridSearchCV tries all combinations of these settings, using cross-validation to test performance on splits of the data, to find which combination works best.



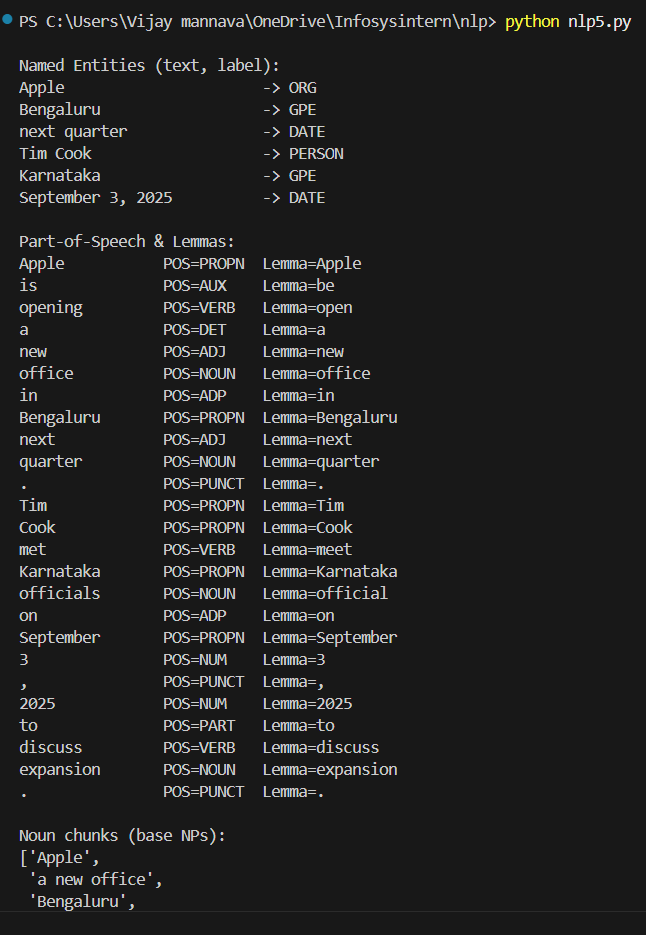
**NLP 04:**

* This code is about discovering hidden topics in a set of documenta using a method called LDA.
* The code has a few short documents, some about pets and other about finance.
* It converts the documents into a matrix of word counts, ignoring common English stop words.

**A screenshot of a computer

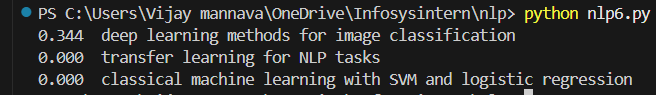
AI-generated content may be incorrect.**

**NLP O5:**

* This code demonstrates how to use spacy.
* It loads a pre-trained English Language model from spacy for fast text analysis.
* It prints each entity with its type label.
* It also enables each word with its part of speech and shows the lemma.
* Finally, it extracts noun chunks, which are basic nouns.

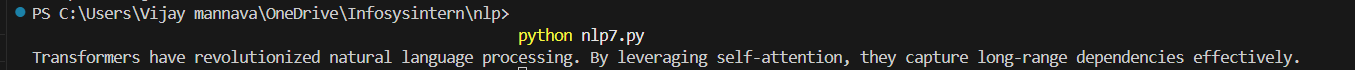
**NLP 06:**

* The code shows how to use spacy for text analysis.
* It loads a pre-trained English model for processing and undersyanding sentences.
* It finds and prints entities.

****

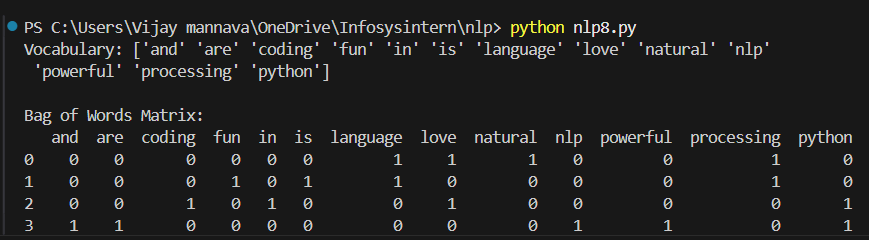
**NLP 07:**

* This code splits the input text into sentences using punctuation marks like periods and question marks.
* It preprocesses the text by extracting words, filtering out common stop words and short words.
* The method picks the most important sentences using word frequency.

****

**NLP 08:**

* This code demonstrates the Bag of Words model for representing text data as numbers.
* It takes a small collection of sentences and extracts all unique words to create a vocabulary.
* The output is matrix where rows are sentences, and columns correspond to a word from vocabulary.



**NLP 09:**

* This code shows how to use TF-IDF to convert text documents into numerical vectors that reflect word importance.

A screenshot of a computer screen

AI-generated content may be incorrect.

**NLP 10:**

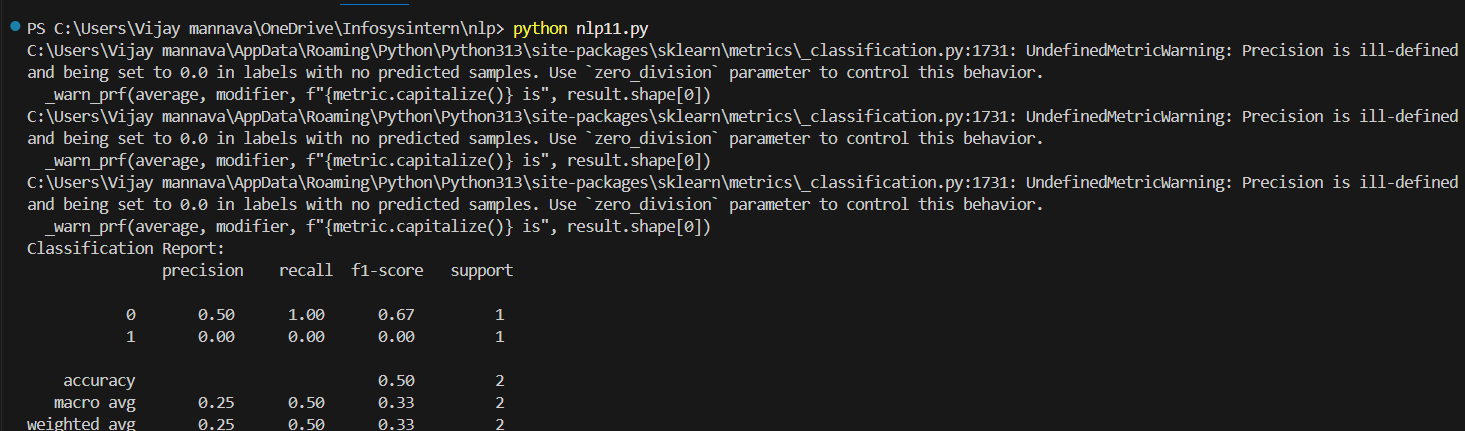
* This code uses genism to create word embeddings from a small list of sentences that are already split into words.
* This model helps to understand word meaning and relationships in numeric form for use in NLP tasks.

A screen shot of a computer

AI-generated content may be incorrect.

**NLP 11:**

* The code uses a tiny dataset of short movie review sentences labelled as positive or negative sentiments.
* It splits the data into training and test sets to evaluate the model’s performance.



**NLP 12:**

* This code converts a list of text documents into numerical vectors using TF-DIF vectorization.
* The result is a matrix.

A computer screen shot of a number

AI-generated content may be incorrect.

**NLP 13:**

* The code takes a small list of documents and splits each into lowercase words.
* It is an NLP technique for understanding large text datasets.

