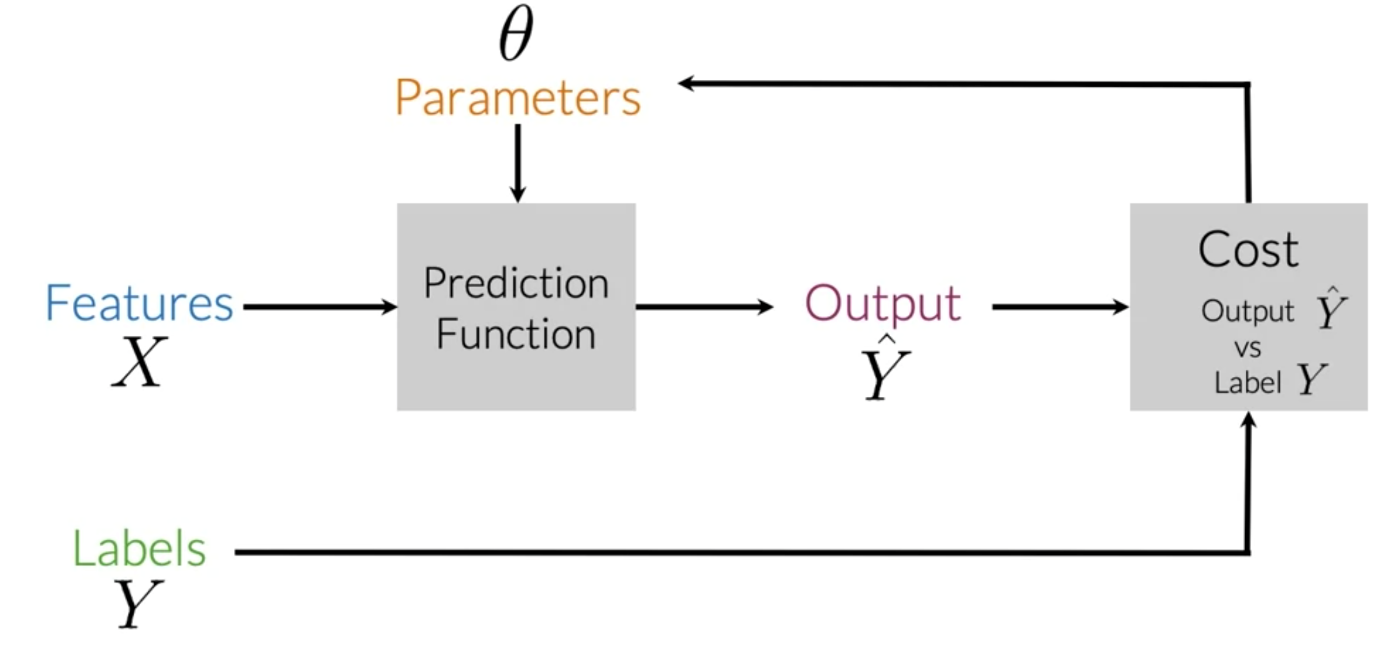
**Week 1 -**

**Supervised ML & Sentiment Analysis**

In supervised machine learning, you usually have an input X*X*, which goes into your prediction function to get your *Y*^. You can then compare your prediction with the true value Y. This gives you your cost which you use to update the parameters θ. The following image summarizes the process.



To perform sentiment analysis on a tweet, you first have to represent the text (i.e. "I am happy because I am learning NLP ") as features, you then train your logistic regression classifier, and then you can use it to classify the text.

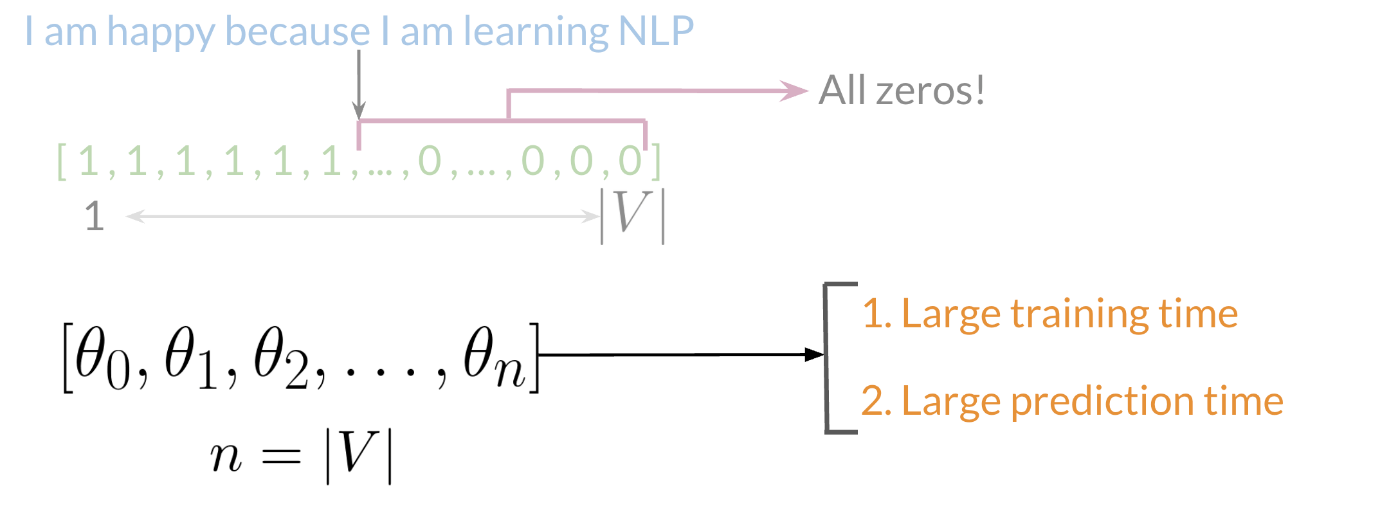
A diagram of a train class

Description automatically generated

Note that in this case, you either classify 1, for a positive sentiment, or 0, for a negative sentiment.

**Vocabulary & Feature Extraction**

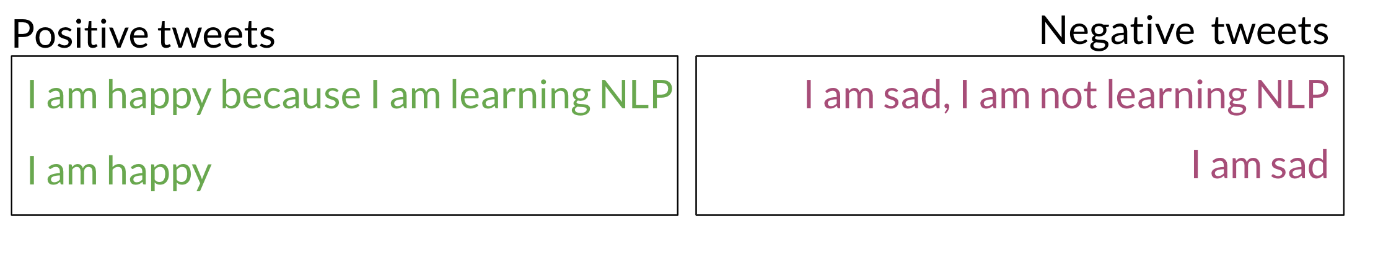
Given a tweet, or some text, you can represent it as a vector of dimension *V*, where *V* corresponds to your vocabulary size. If you had the tweet "I am happy because I am learning NLP", then you would put a 1 in the corresponding index for any word in the tweet, and a 0 otherwise.



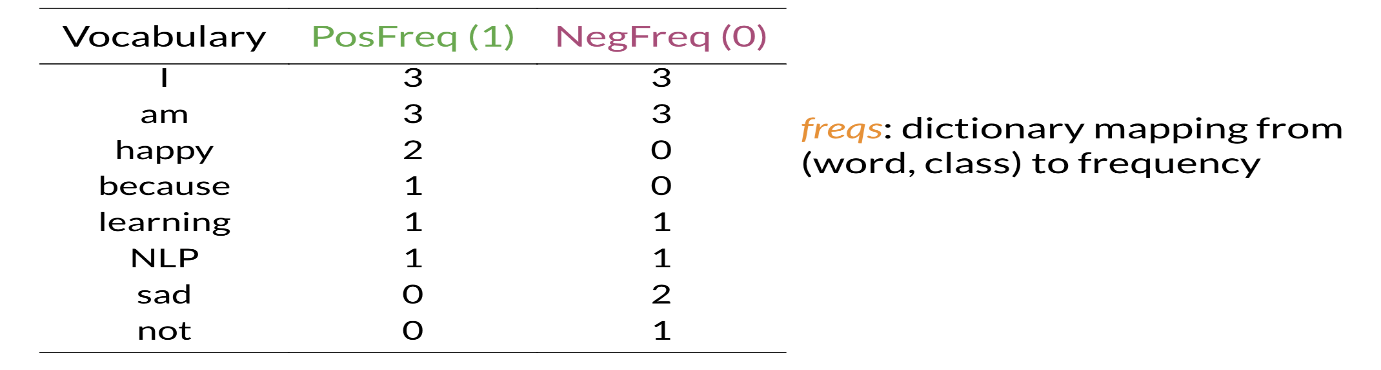
As you can see, as *V* gets larger, the vector becomes more sparse. Furthermore, we end up having many more features and end up training *θ* *V* parameters. This could result in larger training time, and large prediction time.

**Feature Extraction with Frequencies**

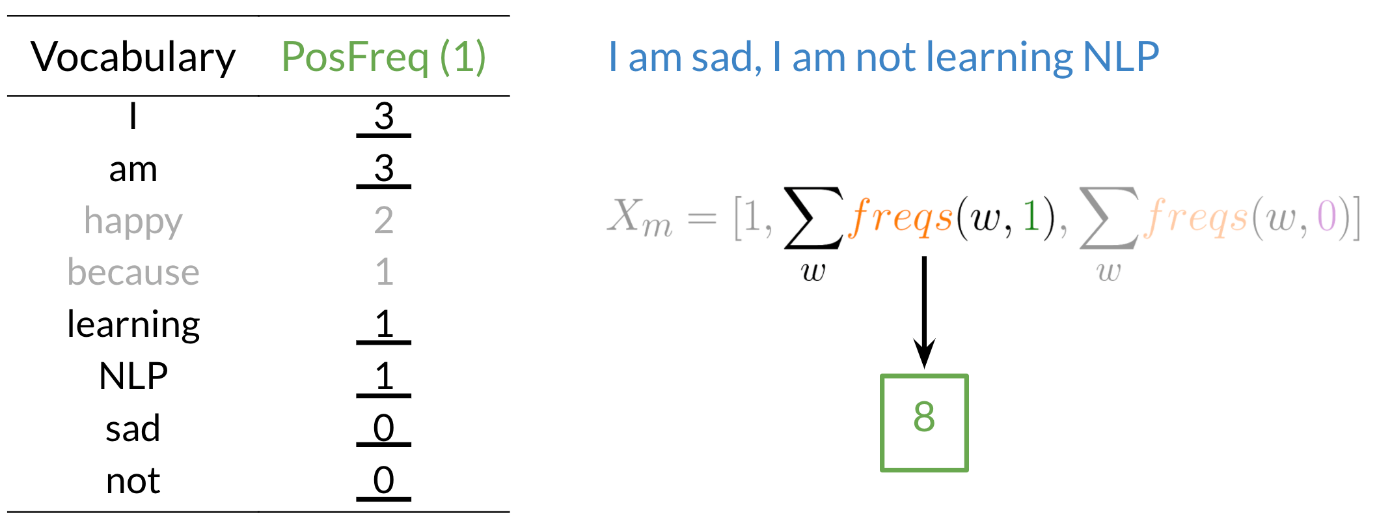
Given a corpus with positive and negative tweets as follows:



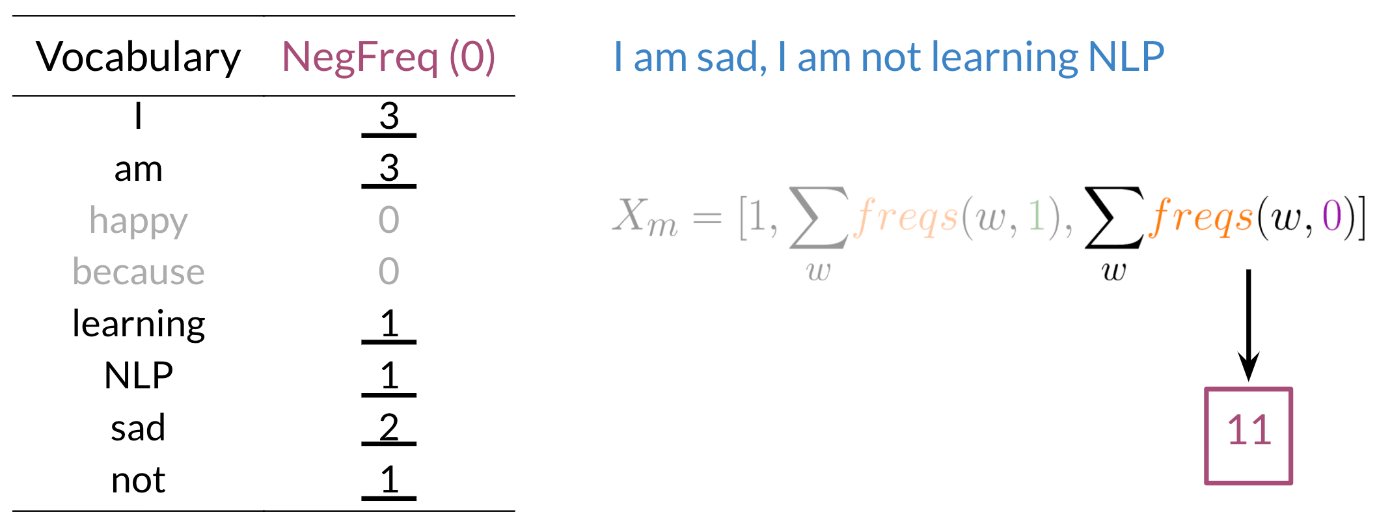
You have to encode each tweet as a vector. Previously, this vector was of dimension V*V*. Now, as you will see in the upcoming videos, you will represent it with a vector of dimension 33. To do so, you have to create a dictionary to map the word, and the class it appeared in (positive or negative) to the number of times that word appeared in its corresponding class.



In the past two videos, we call this dictionary `freqs`. In the table above, you can see how words like happy and sad tend to take clear sides, while other words like "I, am" tend to be more neutral. Given this dictionary and the tweet, "I am sad, I am not learning NLP", you can create a vector corresponding to the feature as follows:



To encode the negative feature, you can do the same thing.



Hence you end up getting the following feature vector [1,8,11] [1,8,11]. 11 corresponds to the bias, 88 the positive feature, and 1111 the negative feature.

**Preprocessing**

When preprocessing, you have to perform the following:

1. Eliminate handles and URLs
2. Tokenize the string into words.
3. Remove stop words like "and, is, a, on, etc."
4. Stemming- or convert every word to its stem. Like dancer, dancing, danced, becomes 'danc'. You can use porter stemmer to take care of this.
5. Convert all your words to lower case.

For example the following tweet "@YMourri and @AndrewYNg are tuning a GREAT AI model at https://deeplearning.ai!!!" after preprocessing becomes

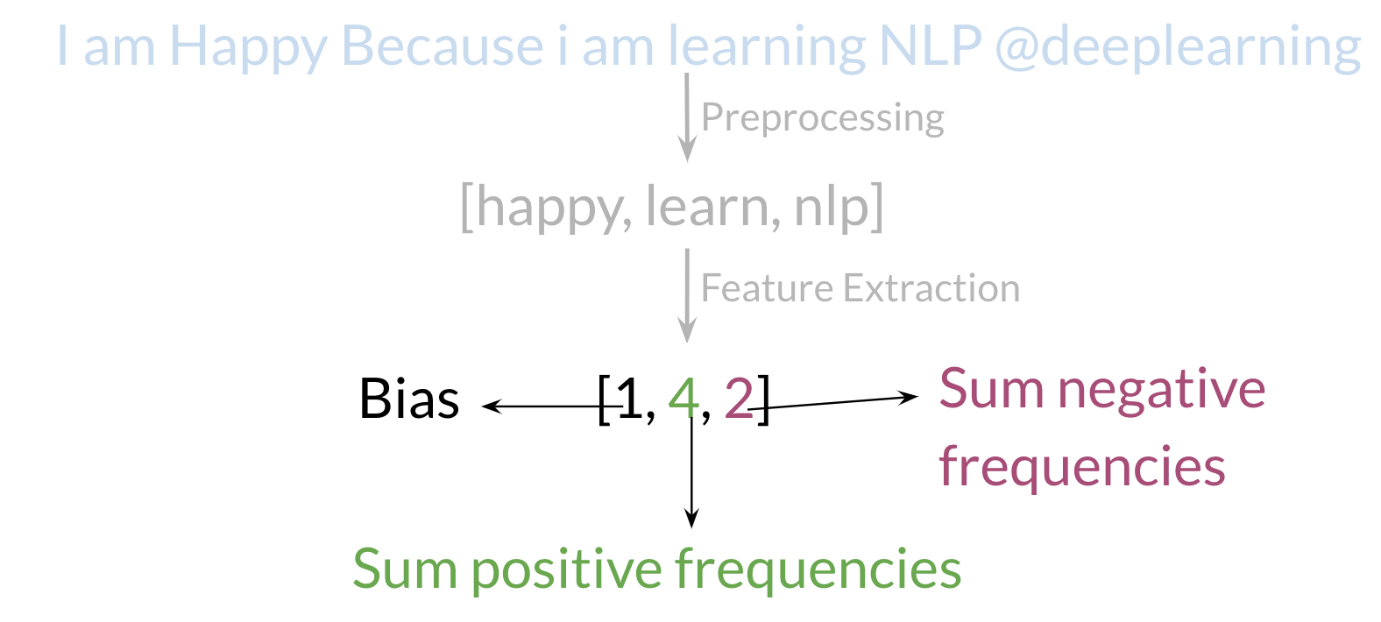
A screenshot of a computer

Description automatically generated

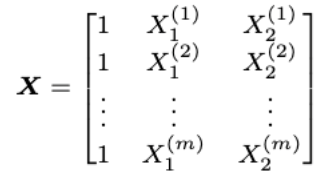
[tun,great,ai,model][*tun*,*great*,*ai*,*model*]. Hence you can see how we eliminated handles, tokenized it into words, removed stop words, performed stemming, and converted everything to lower case.

**Putting it all together**

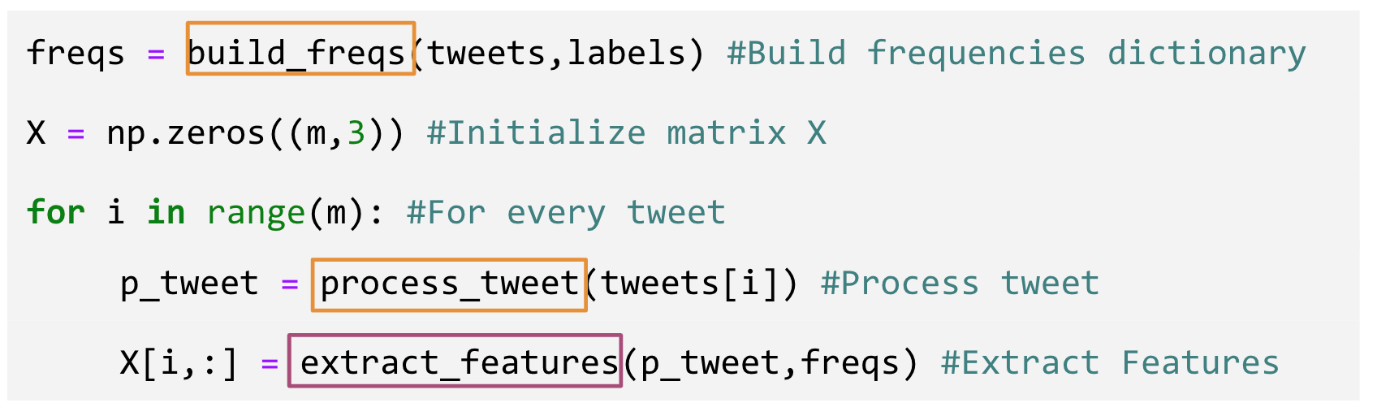
Overall, you start with a given text, you perform preprocessing, then you do feature extraction to convert text into numerical representation as follows:



Your X*X* becomes of dimension (m,3) (*m*,3) as follows.



When implementing it with code, it becomes as follows:



You can see in the last step you are storing the extracted features as rows in your X*X* matrix, and you have m*m* of these examples.