**Vocabulary:**

I love NLP +ve Tweet

NLP is used widely in everyday life +ve Tweet

I want to develop some cool application +ve Tweet

this movie is not good -ve Tweet

I felt sleepy watching this movie -ve Tweet

Tweets: a list of tweets

[tweet\_1, tweet\_2, ….,tweet\_m] 🡪

[I love NLP, NLP is used widely in everyday life, I want to develop some cool application, this movie is not good, I felt sleepy watching this movie]

V = [ I, love, NLP, is used, widely, in everyday, life, want, to, develop, some, cool, application, this, movie, is, not, good, felt, sleepy, watching ]

**Feature Extraction:**

For every tweet, a vector of 1s and 0s depending on the presence of its words in the vocabulary.

For exp: for the tweet “I love NLP”, it’s corresponding feature is

[1 1 1 0 0 0 0….] 🡪 It’s shape is equal to size of the vocabulary.

It’s a sparse representation of 1s and many 0s.

There are problems with this representation.

There will be n+1 parameters, n being the size of the vocabulary. So, if n is very large, training time will be much larger. Secondly, there is no logical connection among the words in the vocabulary.

So, let’s circumvent and look for other ways:

**Positive and negative frequencies for Sentiment Analysis:**

For each word appearing in the vocabulary, check its frequencies occurring in all the positive tweets. Similarly for the negative tweets.

Vocabulary PosFreq NegFreq

I 2 1

love 1 0

NLP 2 0

is 1 1

used 1 0

widely 1 0

in 1 0

everyday 1 0

life 1 0

want 1 0

to 1 0

develop 1 0

some 1 0

cool 1 0

application 1 0

this 0 1

movie 0 1

not 0 1

good 0 1

felt 0 1

sleepy 0 1

watching 0 1

PosCount & NegCount frequencies can be implemented using dictionary in Python.

**Feature Extraction using frequencies**

For each word appearing in +ve tweet, look for its appearance in PosFreq. If it does not appear, make it zero in PosFreq. If it appears, leave its frequency as it is in PosFreq. is Next, sum all the frequencies in PosFreq. for words appearing in the given tweet. Similarly for the negative tweet, same steps are applied.

For the mth input feature:

Here 1 is for the bias.

For example, for the positive tweet, “NLP is used widely in everyday life”, corresponding feature looks like:

= [ 1, 8, 1]

So, instead of shape m of the vector for the input feature, we have the better representation of input tweets as vectors of shape 3!

**Preprocessing**

**Stopwords and punctuation** -

Stopwords: are, in, at, has, for, and…

Punctuation: . ; : ! ‘ “

Handles: @

URLs

For the tweet: “@Manish, you are working hard in NLP at www.manisharya.com!!!”, eliminate stopwords, punctuation, handles and URLs.

~~@Manish~~, you ~~are~~ working hard ~~in~~ NLP ~~at~~ [~~www.manisharya.com~~](http://www.manisharya.com)

It looks like this: you working hard NLP

**Stemming & lowercasing –**

Stem of work – working, worked… so ing, ed are removed in stemming. Also, caps don’t make much of sense, so the letters are lowercased.

So, final tweet looks like: you work hard nlp

**Combining preprocessing and feature extraction (using a vector of shape 3, described above)**

After preprocessing and extracting features (using the above methods) for all tweets: feature input matrix looks like

So by this time, we have our input matrix **X** and label ready to be fed into the model like: logistic regression for SENTIMENT ANALYSIS.