II Naive Bayes (assumption: words in sentence are INDEPENDECE)

① $P(X|Y) = P(Y|X)\frac{P(X)}{P(Y)}$ (Bayes Rule)

② Overview Steps

Raw data

y = 1 : pos tweet
= 0 : neg ↑

train_x	train_y
Tweets	y
T_1	1
T_2	0
⋮	1
T_m	1

⇒ + remove stopwords punctuations
• stemming

Tweets	y
[list of words] (1)	1
↑ ↑ ↑ (2)	0
⋮	1
↑ ↑ ↑ (N)	1

Freqs (count)

	Vocab	Pos	Neg
	word 1	9	80
	word 2	100	0
	⋮		
	$word_N$		
Stem	V words	N_{pos}	N_{neg}

unique words in vocab

Probability (Laplacian Smoothing (+1 in numerator) to avoid probability of a word = 0)

same as table

- Probability of Pos and Neg words

$$P(word_{Pos}) = \frac{freq_{Pos} + 1}{N_{pos} + V}$$

$$P(word_{Neg}) = \frac{freq_{Neg} + 1}{N_{neg} + V}$$

where freq dictionary

```
freqs = {("happi", 1): 99,   # pos class
         ("happi", 0): 8,    # neg class
         ⋮
        }
```

- likelyhood (of the same word w.r.t different classes)

$$\frac{P(word_{Pos})}{P(word_{Neg})} \begin{cases} > 1 : Pos \\ = 1 : neutral \\ < 1 : Neg \end{cases}$$

Ex. $\frac{P('happi'_{Pos})}{P('happi'_{Neg})} = \frac{0.02}{0.001} = 20 > 1 \Rightarrow$ Positive class.

- log likelyhood $\log\left(\frac{P(word_{Pos})}{P(word_{Neg})}\right) = \log P(word_{Pos}) - \log P(word_{Neg})$ $\begin{cases} \geq 0 : Positive \\ < 0 : Negative \end{cases}$

- Prior (of documents)

D : number of documents (in this case tweets)
D_{Pos} : ↑ ↑ ↑ classed in database as Pos class (ex. Pos tweets)
D_{Neg} : ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ Neg class (ex. Neg tweets)

⇒ Probability $P(D_{pos}) = D_{pos}/D$; $P(D_{Neg}) = \frac{D_{Neg}}{D}$

Prior probability = $\frac{P(D_{pos})}{P(D_{Neg})} = \frac{D_{Pos}}{D_{Neg}} \Rightarrow \log Prior = \log D_{pos} - \log D_{Neg}$

Python

```
def train(freqs, train_x, train_y):
    ⋮
    return loglikelyhood: dict (ex. {'word1': 1.7, 'word2': -0.3, ..})
           logprior: float (ex. 0.0)

def predict(tweet, logprior, loglikelyhood):
    words = process_tweet(tweet)
    p = logprior + Σ_i^N loglikelyhood[word]
    p ≥ 0 : positive
    p < 0 : Negative
```