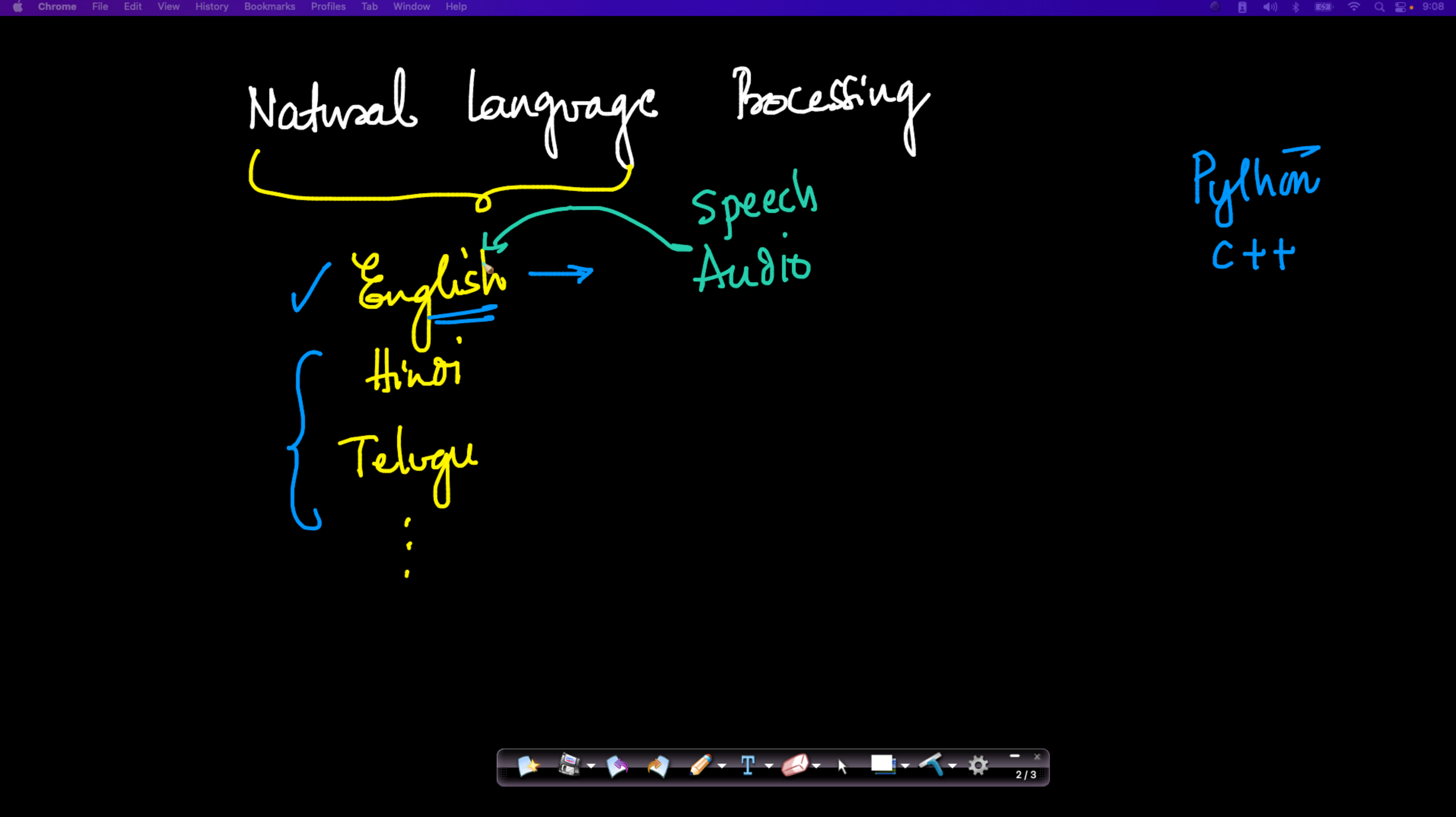
Agenda

- overview of NLP module (what) + applications → why?
- Basic terms & preprocessing

interactive

simpler

- Task: Tweet sentiment classification (Medical)
- Encode text as a vector → smart FE
- build a simple model


Natural language Processing
Speech
Audio
English
Hindi
Telugu
Python
C++
2 / 3

NLP

Tasks:

- → Chatbots
- → Translation
- → Speech to text
- → Closed Caption (Youtube) → Q&A ✓
- → AZ 126 → Number plate recognition
  - Image → YOLOV5 → OCR ✓
- → Text correction/enhance
- → Text summarization
- → Sentiment analysis
- → Text Generation

NLP

— Rule-based / theoretic NLP —

Classical-NLP → basics / encoding / Pre-processing...

↓

Bolo, TFIDF (few decades)

(DL) Word2Vec → autoencoders (~ 2012-13)

Probabilistic techniques → Naive Bayes..

→ LDA, CRF

↑

15-20 yrs

✓ RNN → sequence → words; chars; time-series; audio music ....

LSTM, GRU (~ 2014 - 18)

↓

variations

Transformers:

2018 – Date

↳ Attention

→ BERT, GPT ...

1.1 Understanding the Dataset

1. Dataset source: kaggle covid tweets link
2. Total 11,663 tweets with positive and negative labels

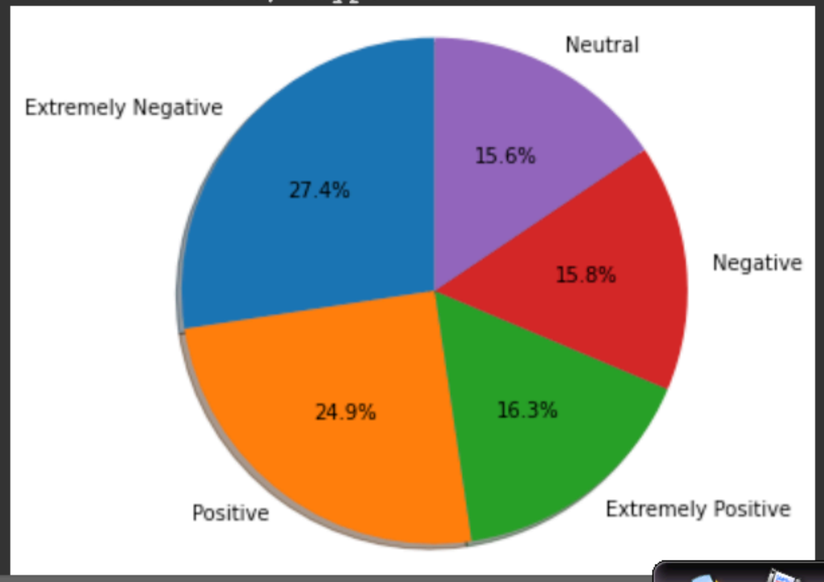
```
[ ] import os
    import re
    import random
    import string      # for string operations
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    # SetUp NLTK
    !pip install --user -U nltk
    import nltk
    nltk.download('punkt')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.7)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from nltk) (4.64.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.7/dist-packages (from nltk) (2022.6.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from nltk) (1.2.0)
```

```
  plt.pie(sizes, labels=labels, autopct='%1.1f%%',
          shadow=True, startangle=90)
  # Equal aspect ratio ensures that pie is drawn as a circle.
  plt.axis('equal')
  # Display the chart
  plt.show()

pie_chart(dataframe)
```

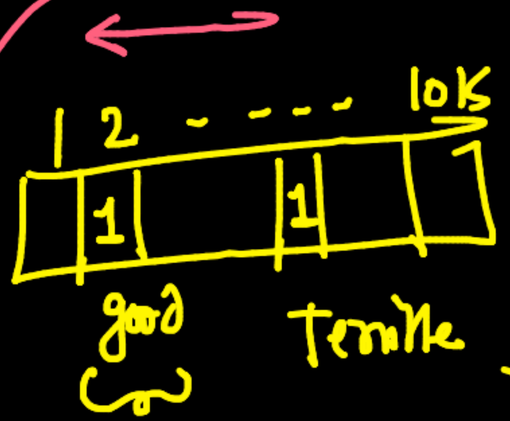
```
Labels in the dataset:  ['Extremely Negative' 'Positive' 'Extremely Positive' 'Negative' 'Neutral']
Negative              1041
Positive               947
Neutral                619
Extremely Positive     599
Extremely Negative     592
Name: Sentiment, dtype: int64
```



tweet/text ⟶ 1/0

① tweet → Numerical vector

↳ BoW (Naive Bayes) ✓



② $(x_i, y_i)_{i=1}^{n}$

$x_i \in \mathbb{R}^d$ $y_i \in \{1, 0\}$

logistic Regr

✓ [List of +ve words

✓ great

-ve words

✓ terrible

Rule based

- Meaning of the text could be interpreted by analyzing the words.

> **This is where tokenization is essential to proceed with NLP (text data), which is used to break down the text into tokens (words).**

What is tokenization?

- Word tokenization: split large text samples into words.

```
# Split on spaces
tweet = all_positive_tweets[12]
word = tweet.split() # space tokenizer
print('Using Split :',word)
```

```
', 'due', 'to', 'coronavirus.', 'https://t.co/5mYfz3RAD0', 'retail', '#ecommerce', '#study', '#coronavirus', 'https
```

regex regex

Note: split() does not consider punctuation as a separate token

What is a regular expression and what makes it so important?

- A regular expression is a notation to represent standards in strings.
- Series of characters that define an abstract search pattern.
- It serves to search and extract information in texts.

Regex is used in Google analytics in URL matching in supporting search and replaced in most popular editors like Sublime, Notepad++, Google Docs, and Microsoft Word.

```
Example :  Regular expression for an email address :
        ^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$
```

The above regular expression is used for checking if a given set of characters is an email address or not.

▸ How to write regular expression?

Read More: Link1, Link2, regex-quickstart, How to write regular expression?

```
[ ]  # Using 're' library to work with regular expression.

    tokens = re.findall("[\w']+", tweet)
    print('Using regexes :',tokens)
```

```
Using regexes : ['Consumers', 'have', 'increased', 'their', 'online', 'shopping', 'due', 'to', 'coronavirus', 'https
```

```
Using NLTK : ['Consumers', 'have', 'increased', 'their', 'online', 'shopping', 'due', 'to', 'coronavirus', '.', 'htt
```

Note: `word_tokenize()` is considering punctuation as a token. Hence we need to remove the punctuations from the initial list.

How to tokenize into a Sentence?

- A sentence usually ends with a full stop `.`, so we can use `.` as a separator to break the string:

```
[ ] # Splits at '.'
    splits = tweet.split('. ')
    print('Using Splits :',splits)
    # Using Regular Expressions (RegEx)
    sentence_splits = re.compile('[.!?] ').split(tweet)
    print('Using regexes :',sentence_splits)
```

```
Using Splits : ['Consumers have increased their online shopping due to coronavirus', 'https://t.co/5mYfz3RAD0 #retai
Using regexes : ['Consumers have increased their online shopping due to coronavirus', 'https://t.co/5mYfz3RAD0 #reta
```

Note: a drawback of using Python's `split()` method is that we can use only one separator at a time.

- Using RegEx, we have an edge over t... ltaneously

```
[ ] tweet
```

```
'Consumers have increased their online shopping due to coronavirus. https://t.co/5mYfz3RAD0 #retail #ecommerce #stud
y #coronavirus https://t.co/Dz3H6zrWUT'
```

```
[ ] # Removing hyperlinks and hashtags

    print('\033[92m' + tweet)
    print('\033[94m')

    # remove old style retweet text "RT"
    tweet2 = re.sub(r'^RT[\s]+', '', tweet)

    # remove hyperlinks
    tweet2 = re.sub(r'https?://[^\s\n\r]+', '', tweet2)

    # remove hashtags
    # only removing the hash # sign from the word
    tweet2 = re.sub(r'#', '', tweet2)
    print(tweet2)
```

```
Consumers have increased their online shopping due to coronavirus. https://t.co/5mYfz3RAD0 #retail #ecommerce #study

Consumers have increased their online shopping due to coronavirus.  retail ecommerce study coronavirus
```

A lot of zeros! That's a sparse representation.

n

~ 2000 tweets

Drawbacks of sparse representation

1. A lot of features are equal to 0.
2. Logistic regression model would have to learn `n + 1` parameters, where `n = size of the vocabulary`
3. For large vocabulary sizes, this would be problematic.

10K words

10,000

d >> n

What problems will it cause?

▼ **Answers**

1. Excessive amount of time to train a model and more time to make predictions.

1.5 How can we reduce the dimension?

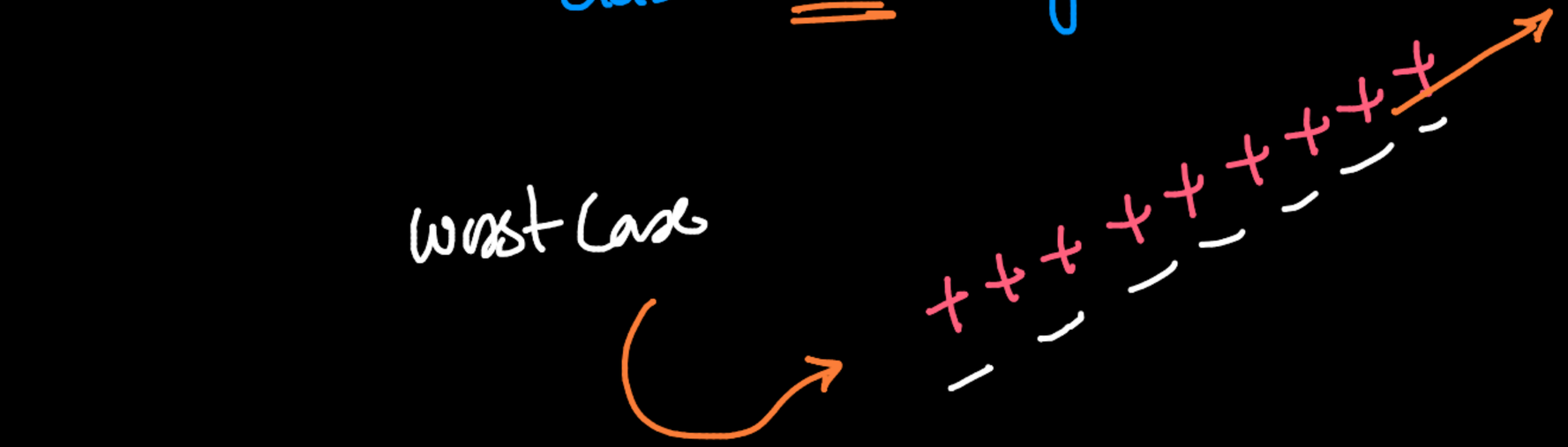
- Previously, the word vector was of dimension `V`. Now, we will represent it with a vector of dimension `3`
- To do so, create a dictionary to map) to the number of times that word

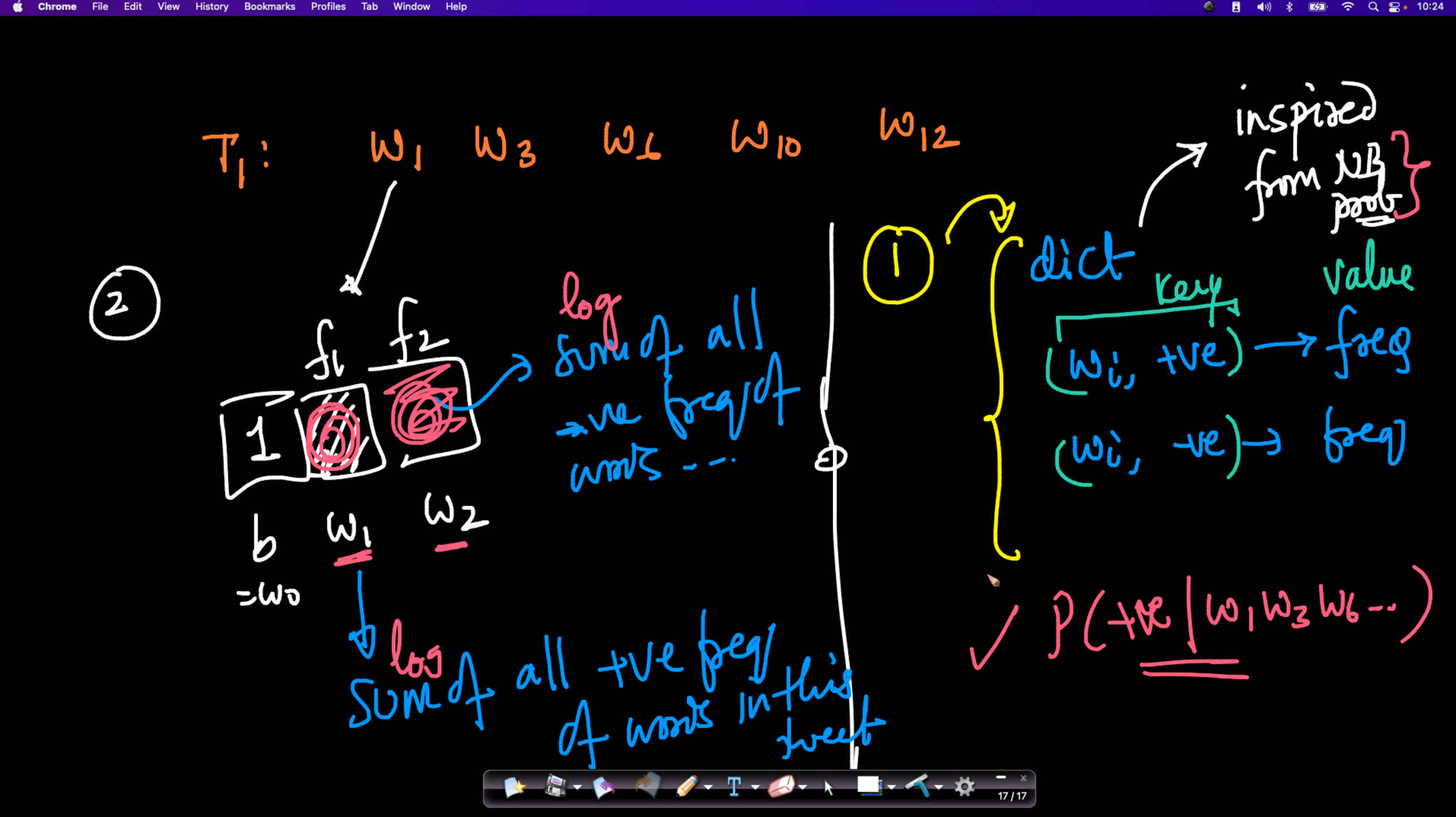
Q

BoW + PCA → Can do it → may
→ maynot

↓

class label agnostic

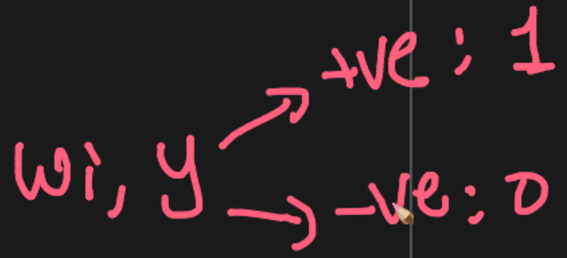



Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:24
T_1: w_1 w_3 w_4 w_{10} w_{12}
2
f_1 f_2
1
b
w_1
w_2
=w0
log
Sum of all
+ve freq of
words ---
log
Sum of all +ve freq
of words in this
tweet
1
dict
inspired
from NB
pools
key
value
$(w_i, +ve) \rightarrow$ freq
$(w_i, -ve) \rightarrow$ freq
$P(+ve \mid w_1, w_3, w_6 \ldots)$
17 / 17

```
        freqs: a dictionary mapping each (word, sentiment) pair (=key) to its frequency (=value)
    """
    # Convert np array to list since zip needs an iterable.
    # The squeeze is necessary or the list ends up with one element.
    # Also note that this is just a NOP if ys is already a list.
    yslist = np.squeeze(ys).tolist()

    # Start with an empty dictionary and populate it by looping over all tweets
    # and over all processed words in each tweet.
    freqs = {}
    for y, tweet in zip(yslist, tweets):
        for word in process_tweet(tweet):
            pair = (word, y)
            if pair in freqs:
                freqs[pair] += 1
            else:
                freqs[pair] = 1

    return freqs
```




```
training_tweets = positive_train + negative_train
# make a numpy array representing labels of the tweets
labels = np.append(np.ones((len(positive_train))), np.zeros((len(negative_train))))
```

```
# create frequency dictionary
```

```
[ ]            pos = freqs[(word, 1)]

        # retrieve number of negative counts
        if (word, 0) in freqs:
            neg = freqs[(word, 0)]

        # append the word counts to the table
        data.append([word, pos, neg])

    data
```

pos neg

```
[['void', 1, 0],
 ['commun', 15, 10],
 ['spirit', 1, 2],
 ['stop', 14, 32],
 ['merri', 0, 0],
 ['nice', 6, 6],
 ['good', 49, 28],
 ['bad', 1, 8],
 ['sad', 0, 5],
 ['mad', 2, 8],
 ['best', 7, 3],
 ['pretti', 6, 3],
 [':)', 1, 1],
 [':(', 3, 0],
 ['song', 2, 0],
 ['idea', 6, 8],
```

```
[ ]             pos = freqs[(word, 1)]

        # retrieve number of negative counts
        if (word, 0) in freqs:
            neg = freqs[(word, 0)]

        # append the word counts to the table
        data.append([word, pos, neg])

    data
```

```
[['void', 1, 0],
 ['commun', 15, 10],
 ['spirit', 1, 2],
 ['stop', 14, 32],
 ['merri', 0, 0],
 ['nice', 6, 6],
 ['good', 49, 28],
 ['bad', 1, 8],
 ['sad', 0, 5],
 ['mad', 2, 8],
 ['best', 7, 3],
 ['pretti', 6, 3],
 [':)', 1, 1],
 [':(', 3, 0],
 ['song', 2, 0],
 ['idea', 6, 8],
```

```
[ ]            pos = freqs[(word, 1)]

        # retrieve number of negative counts
        if (word, 0) in freqs:
            neg = freqs[(word, 0)]

        # append the word counts to the table
        data.append([word, pos, neg])

    data
```

```
[['void', 1, 0],
 ['commun', 15, 10],
 ['spirit', 1, 2],
 ['stop', 14, 32],
 ['merri', 0, 0],
 ['nice', 6, 6],
 ['good', 49, 28],
 ['bad', 1, 8],
 ['sad', 0, 5],
 ['mad', 2, 8],
 ['best', 7, 3],
 ['pretti', 6, 3],
 [':)', 1, 1],
 [':(', 3, 0],
 ['song', 2, 0],
 ['idea', 6, 8],
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

NLP curriculum - Google Docs | Intro_to__NLP.ipynb - Colabor | Demo – InferKit | NLTK :: Natural Language Tooll | spaCy · Industrial-strength Nat | printing - Python output compl | Regex Cheat Sheet

colab.research.google.com/drive/183HUnHXFQxUJrrpy6mn-HwRCvokGVj2X#scrollTo=_ltH48Nrhq9U

+ Code + Text Connect

```
[ ]  data
```

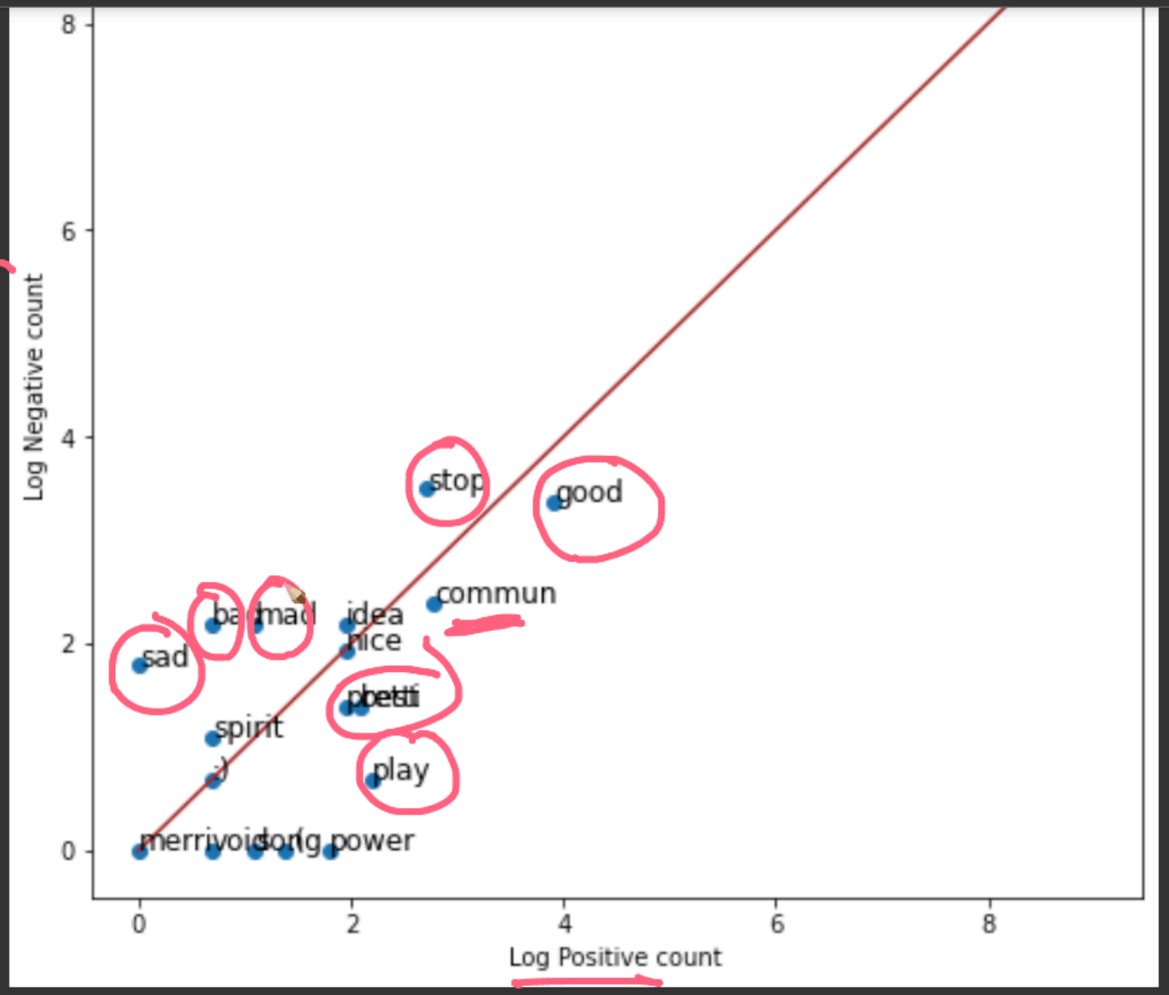
```
[['void', 1, 0],
 ['commun', 15, 10],
 ['spirit', 1, 2],
 ['stop', 14, 32],
 ['merri', 0, 0],
 ['nice', 6, 6],
 ['good', 49, 28],
 ['bad', 1, 8],
 ['sad', 0, 5],
 ['mad', 2, 8],
 ['best', 7, 3],
 ['pretti', 6, 3],
 [':)', 1, 1],
 [':(', 3, 0],
 ['song', 2, 0],
 ['idea', 6, 8],
 ['power', 5, 0],
 ['play', 8, 1]]
```

```
[ ]  fig, ax = plt.subplots(figsize = (8, 8))

     # convert positive raw counts to logarithmic scale. we add 1 to avoid log(0)
     x = np.log([x[1] + 1 for x in data])
```

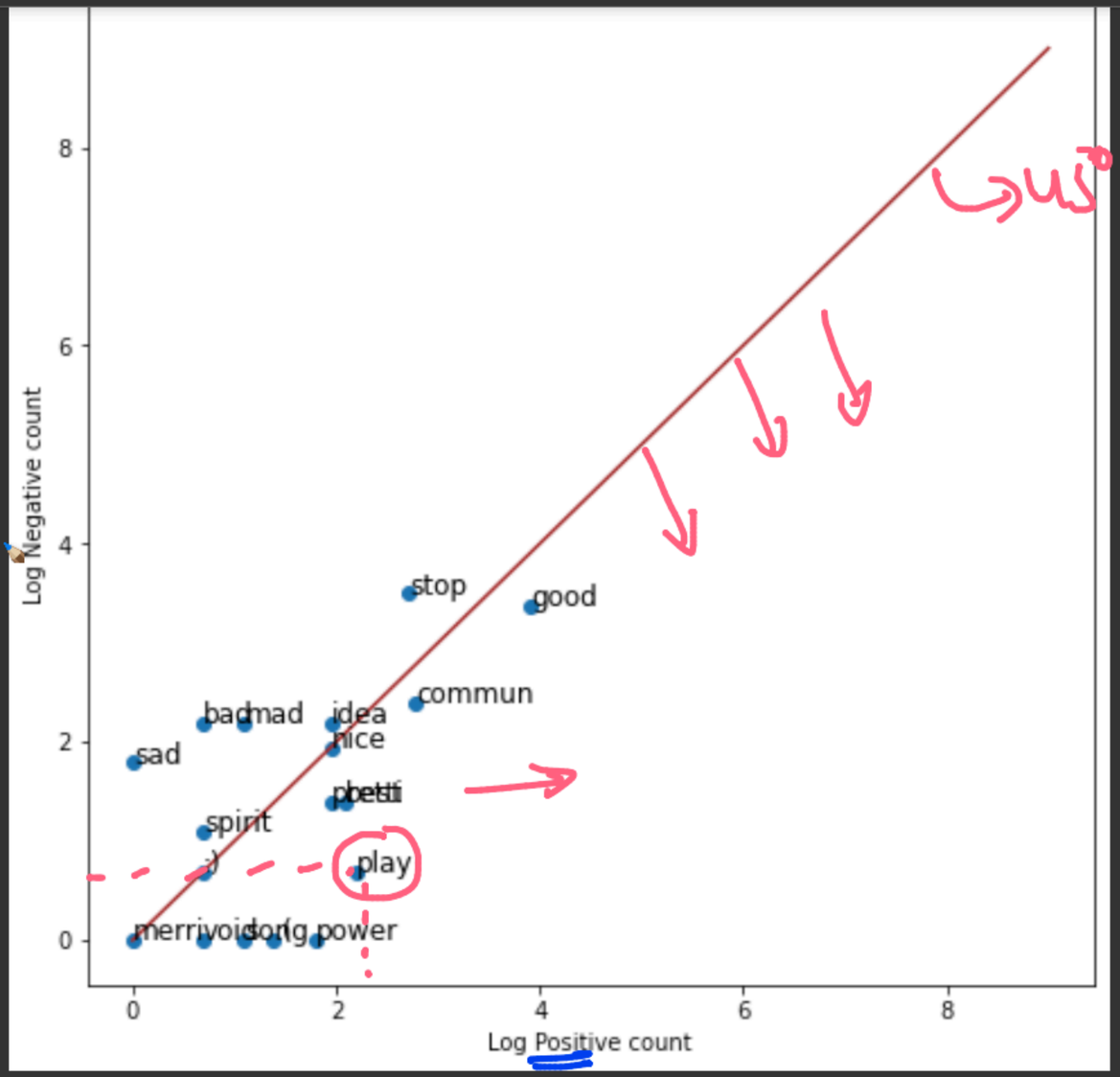
22 / 22

[ ]



Extracting features function.

[ ]



Extracting features from Frequency Dictionary.

Feature extraction

freqs: dictionary mapping from (word, class) to frequency

$$X_m = [1, \sum_w freqs(w, 1), \sum_w freqs(w, 0)]$$

- X_m: Features of tweet m
- 1: Bias
- $\sum_w freqs(w, 1)$: Sum Pos. Frequencies
- $\sum_w freqs(w, 0)$: Sum Neg. Frequencies

f_1 f_2

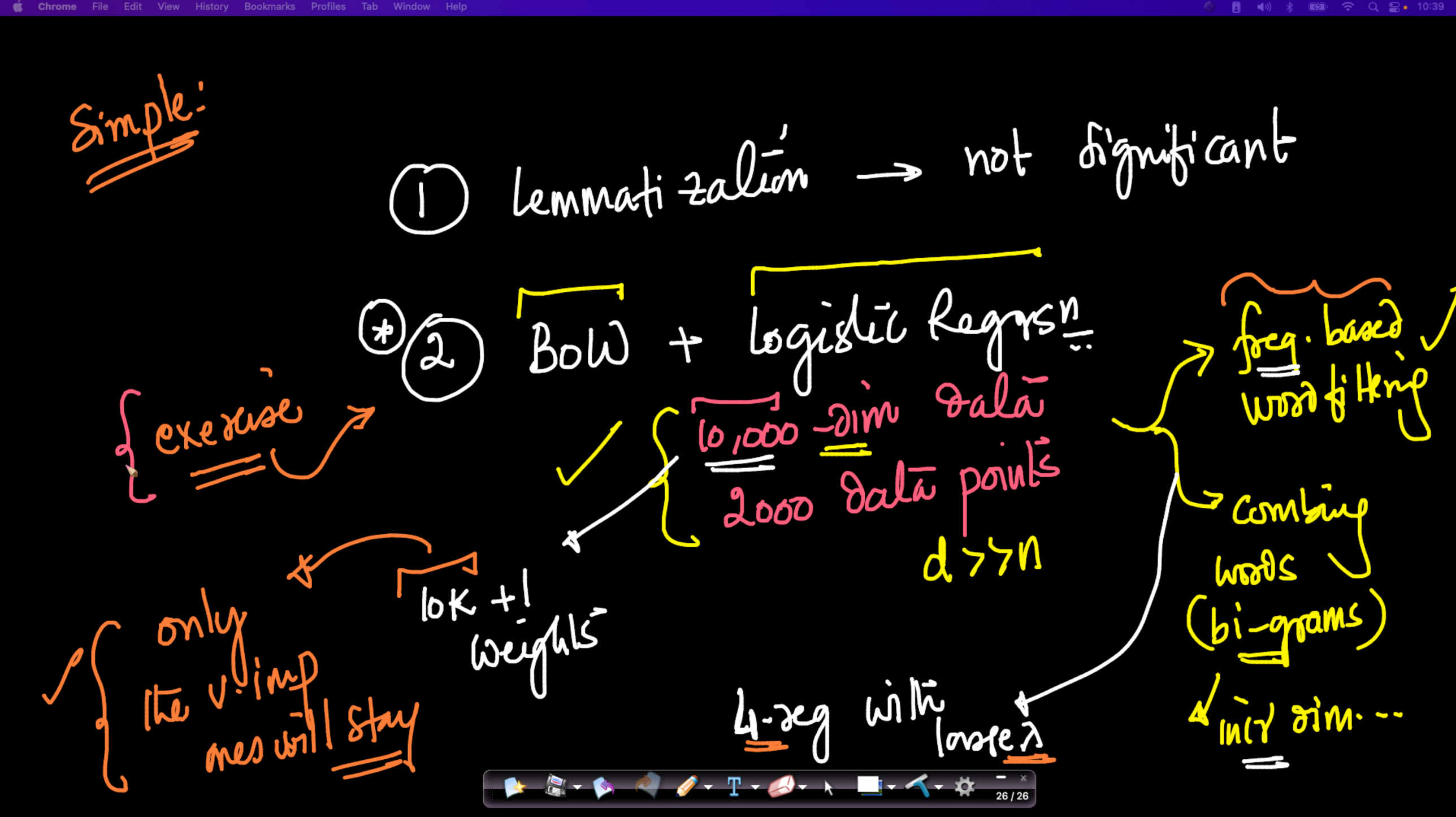
$x_i \in \mathbb{R}^3 \to \mathbb{R}^m$

$y_i \in \{0, 1\}$

FE

$P(w, +ve)$

```
[ ]  # select some words to appear in the report.
     keys = ['void', 'commun', 'spirit', 'stop', 'merri', 'nice', 'good', 'bad', 'sad', 'mad', 'best', 'pretti',
             ':)', ':(', 'song', 'id
```


Simple:
1 lemmatization → not significant
2 BoW + Logistic Regression
exercise
10,000-dim data
2000 data points
d >> n
10K+! weights
only the v! imp ones will stay
freq. based word filtering
combine words (bi-grams)
incr dim...
L1-reg with lower λ
26 / 26

f

Feature extraction

freqs: dictionary mapping from (word, class) to frequency

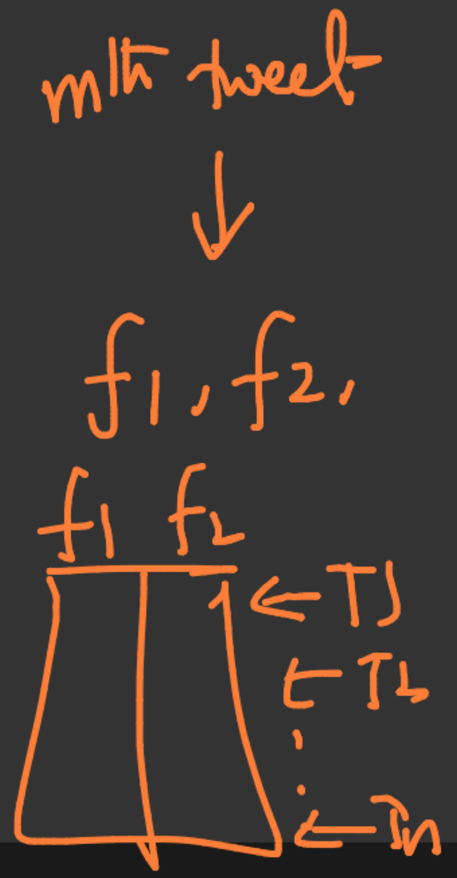
$$X_m = [1, \sum_{w \in tweet\ m} freqs(w, 1), \sum_{w \in tweet\ m} freqs(w, 0)]$$

Features of tweet m

Bias

Sum Pos. Frequencies

Sum Neg. Frequencies




```
[ ]  # select some words to appear in the report.
     keys = ['void', 'commun', 'spirit', 'stop', 'merri', 'nice', 'good', 'bad', 'sad', 'mad', 'best', 'pretti',
             ':)', ':(', 'song', 'idea', 'power', 'play']
```

Feature extraction

freqs: dictionary mapping from (word, class) to frequency

w_1 w_2

$$X_m = [1, \sum_w freqs(w, 1), \sum_w freqs(w, 0)]$$

Features of tweet m

Bias

Sum Pos. Frequencies

Sum Neg. Frequencies

```
[ ]  # select some words to appear in the report.
     keys = ['void', 'commun', 'spirit', 'stop', 'merri', 'nice', 'good', 'bad', 'sad', 'mad', 'best', 'pretti',
             ':)', ':(', 'song', 'idea', 'power', 'play']
```

Feature extraction

freqs: dictionary mapping from (word, class) to frequency

$$X_m = [1, \sum_w freqs(w, 1), \sum_w freqs(w, 0)]$$

Features of tweet m

Bias

Sum Pos. Frequencies

Sum Neg. Frequencies

```
# select some words to appear in the report.
keys = ['void', 'commun', 'spirit', 'stop', 'merri', 'nice', 'good', 'bad', 'sad', 'mad', 'best', 'pretti',
        ':)', ':(', 'song', 'idea', 'power', 'play']
```