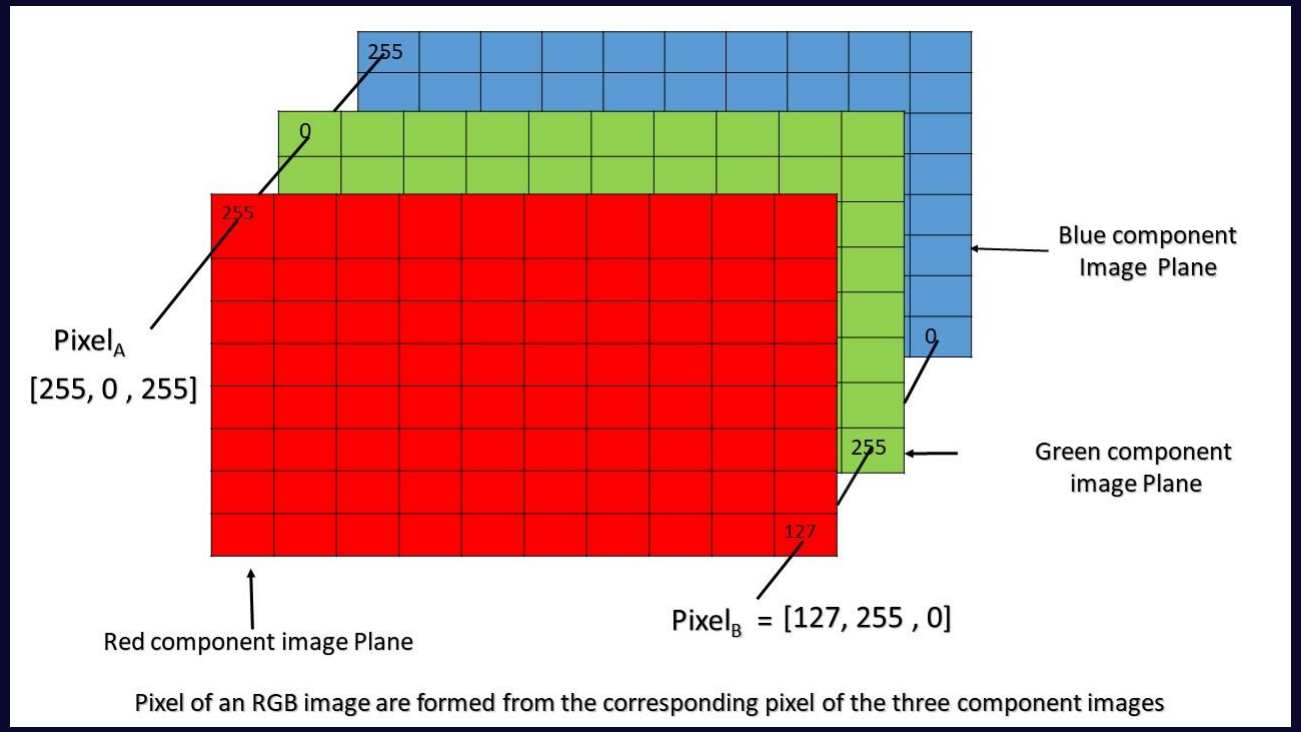
DATA ANALYSIS ON IMAGE AND TEXT DATA

MODULE-12

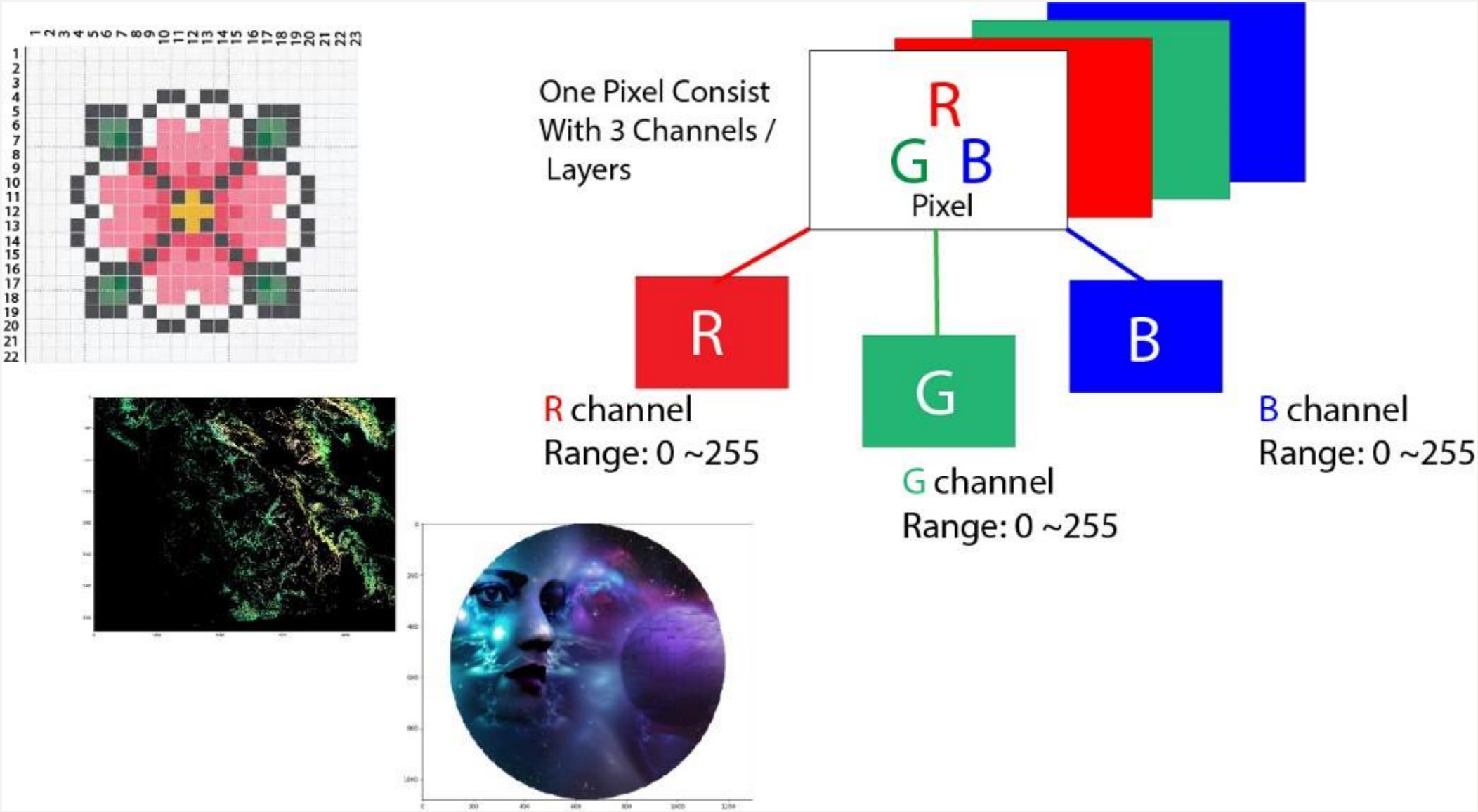
Real-World data will not always be of the numerical form, we need to know how to handle a lot of other forms of data like image and text. In this week we are going to see all essential data analysis techniques on image and text data.



Pixel of an RGB image are formed from the corresponding pixel of the three component images

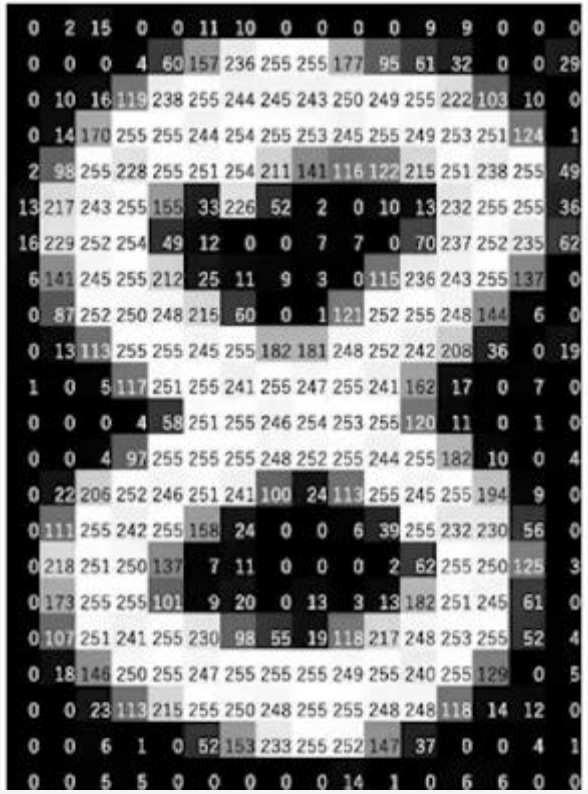
IMAGE DATA ANALYSIS

Image Data Analysis



INTRODUCTION - PIXEL

- Computer store images as a combination of tiny squares (pixels)
- The more and smaller tiles we use, the smoother the images will be. (resolution of the images)
- The word **pixel** means a **picture element.**
- Every photograph, in digital form, is made up of pixels. They are the smallest unit of information that makes up a picture. Usually round or square, they are typically arranged in a 2-dimensional grid.
- A pixel is stored as a 8-bit number, the value ranges from 0-255.
  - 255 - full intensity - White
  - 0 - colors are muted - black



RGB IMAGE

- A simple way to describe each pixel is using a combination of three colors, namely Red, Green, Blue. This is what we call an RGB image.
- The combination of these three colors tends to the highest value among them. Since each value can have 256 different intensity or brightness value, it makes 16.8 million total shades.
- refer this notebook to understand the properties of the image

GREY SCALE IMAGES

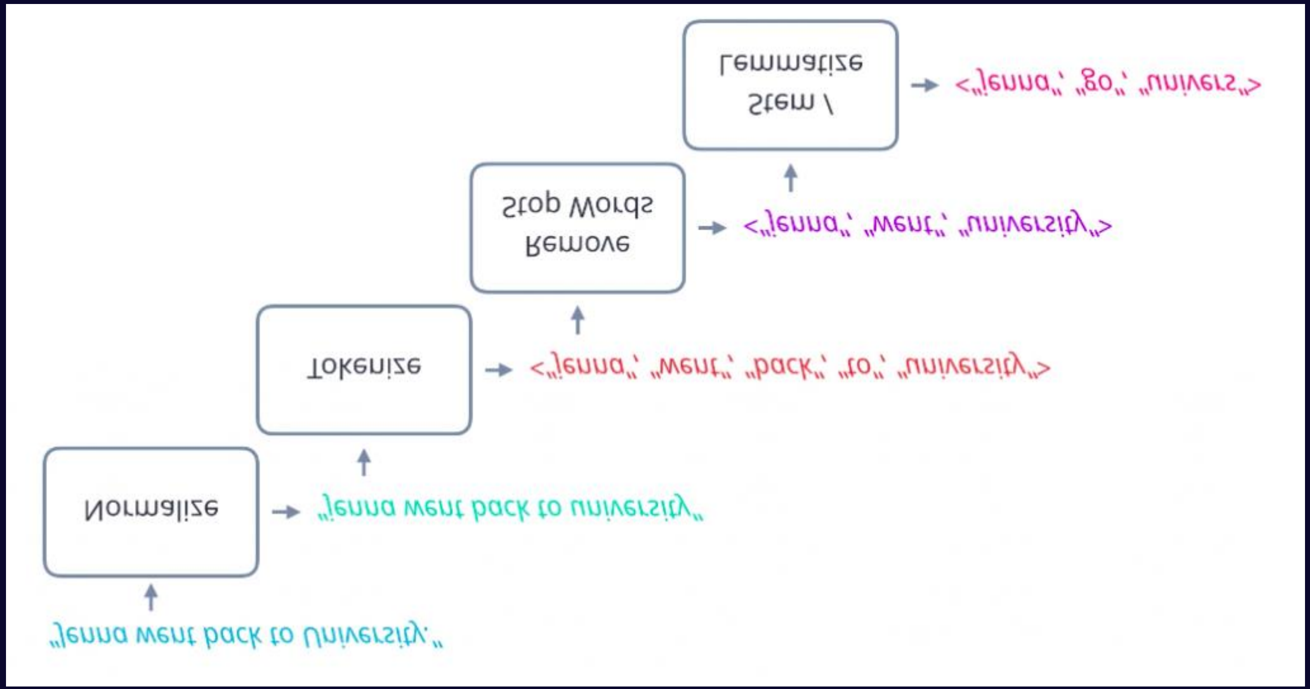
- Black and white images are stored in 2-Dimensional arrays. There're two types of black and white images:
  - Binary: Pixel is either black or white:0 or 255
  - Greyscale: Ranges of shades of grey:0 ~ 255
- Greyscaling
  - is a process by which an image is converted from a full color to shades of grey.
  - In image processing tools, for example: in OpenCV, many functions use greyscale images before processing, and this is done because it simplifies the image, acting almost as noise reduction and increasing processing time as there's less information in the images.

MASKING

- Image masking is an image processing technique that is used to remove the background from which photographs those have fuzzy edges, transparent or hair portions.

- Now, we'll create a mask that is in shape of a circular disc. First, we'll measure the distance from the center of the image to every border pixel values. And we take a convenient radius value, and then using logical operator, we'll create a circular disc. It's quite simple, let's see the code.

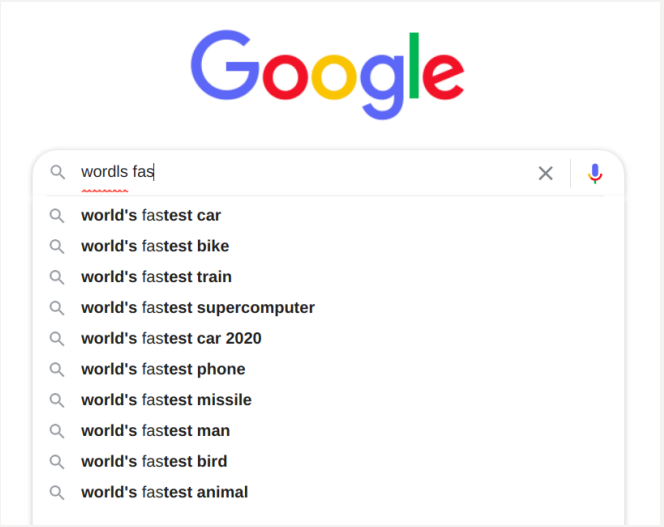
- Refer to colab notebook for basic image preprocsessing
- Explore MNIST notebook
- Learn how to load data from Google drive



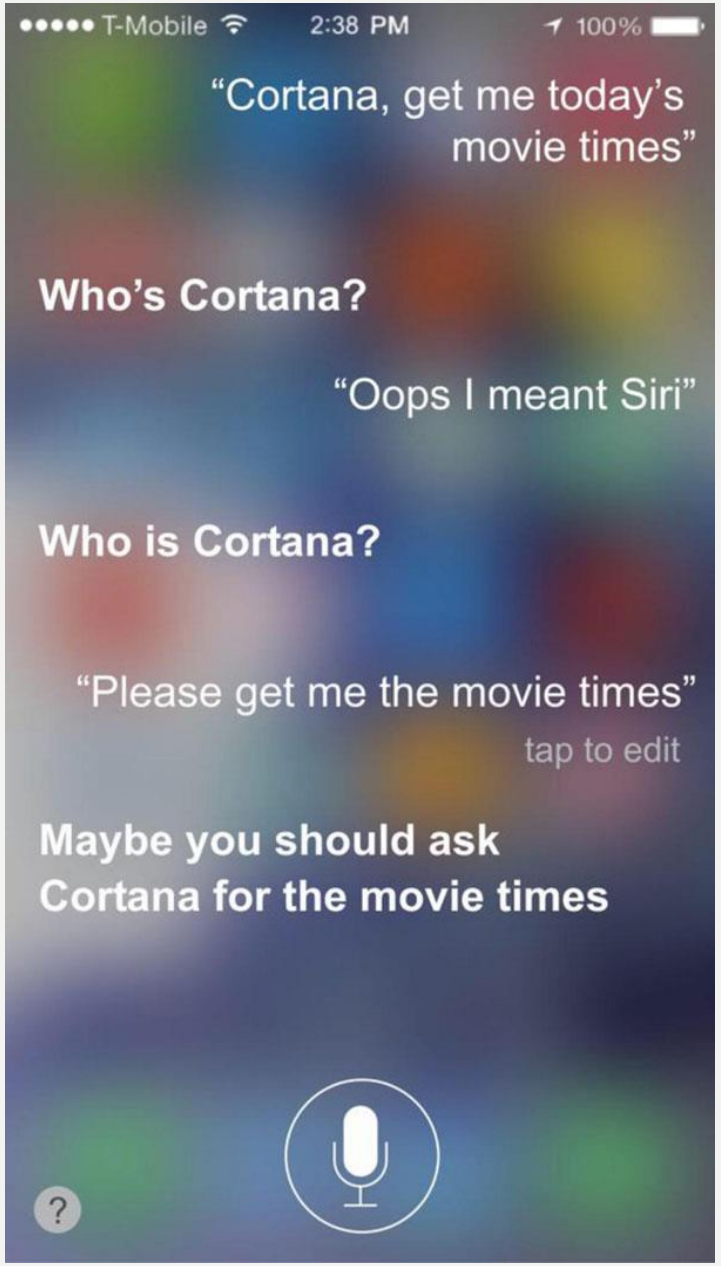
TEXT DATA ANALYSIS

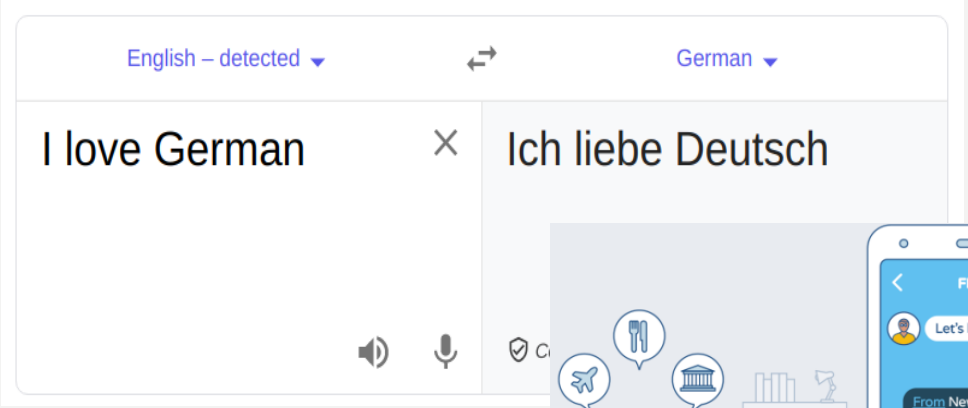
NATURAL LANGUAGE PROCESSING

- NLP
  - is among the hottest topic in the field of data science.
  - Companies are putting tons of money into research in this field.
  - Everyone is trying to understand NLP and its applications to make a career around it.
  - Every business out there wants to integrate it into their business somehow.
- Are you using NLP these days?


Google
wordls fas
world's fastest car
world's fastest bike
world's fastest train
world's fastest supercomputer
world's fastest car 2020
world's fastest phone
world's fastest missile
world's fastest man
world's fastest bird
world's fastest animal


Alexa
Siri
Google Now
Cortana

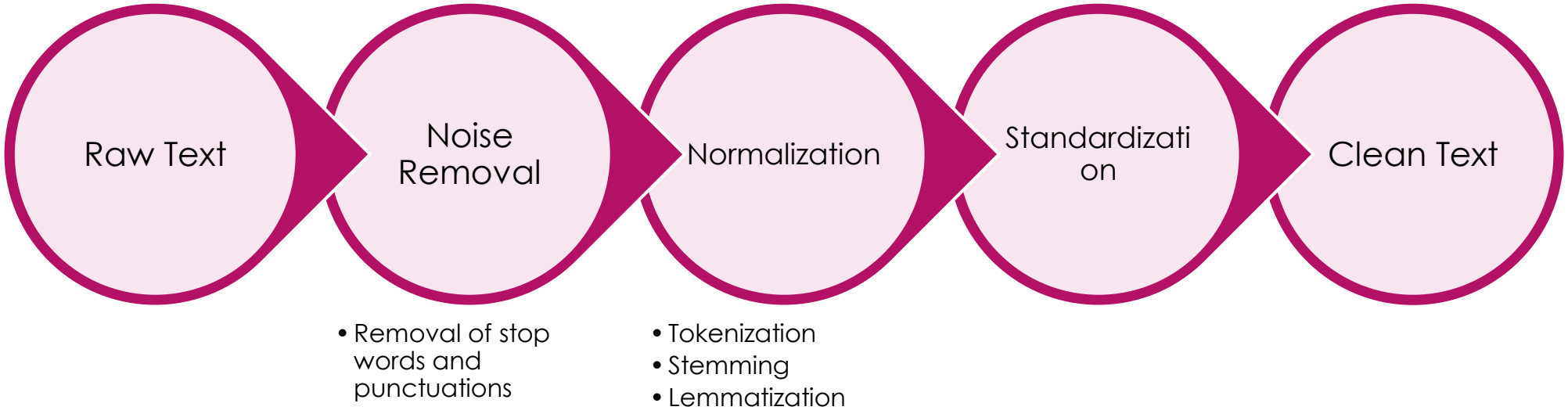

T-Mobile
2:38 PM
100%
"Cortana, get me today's movie times"
Who's Cortana?
"Oops I meant Siri"
Who is Cortana?
"Please get me the movie times"
tap to edit
Maybe you should ask Cortana for the movie times


English – detected
German
I love German
Ich liebe Deutsch


FLIGHT BOT
Let's look for tickets!
Ok
From New York
To Seatle
Departing on Sept 15
Returning on Sept 19
for 2 Adults
Is this info correct?
Yes
I have found 17 results
See all results
Type your message here...

Getting text to workable format

Approximate order of steps for preprocessing text data



Noise Removal

- **Noise** : Any piece of text which is not relevant to the context of the data.
- Generally, the noisy entities are
  - Stop words,
  - punctuation marks.
- Stop words Removal
  - It is a process of removing common language articles, Pronouns and propositions such as “and”, “the” or “to” in English.
  - These words provide little or no value to NLP.
  - Stop words are removed /filtered from text for better efficiency

Stop words using NLTK

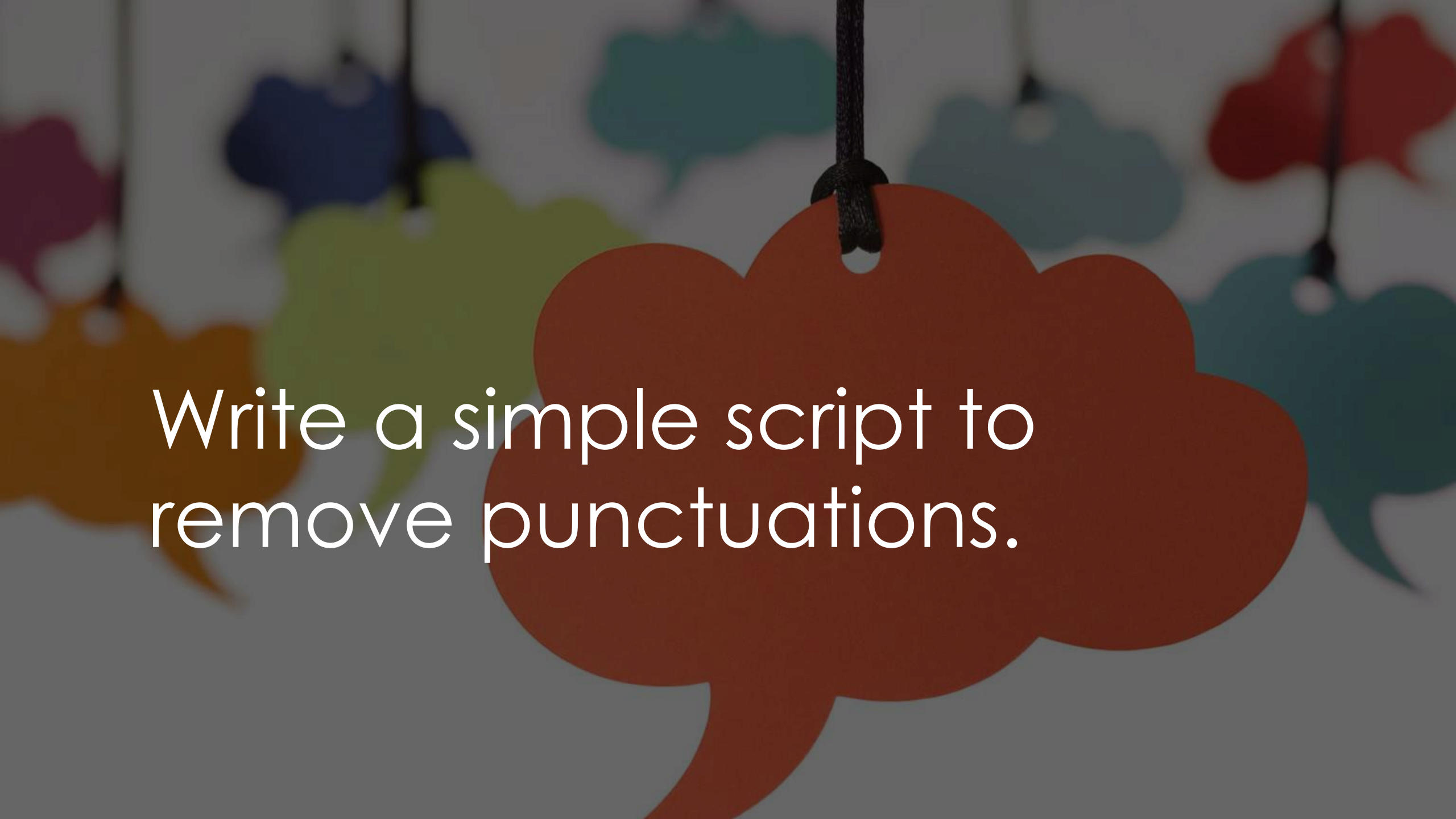
```
import nltk
from nltk.corpus import stopwords
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you
'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she
', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these',
'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'havi
ng', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', '
out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'whe
n', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'som
e', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 'r
e', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doe
sn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn'
, "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't
", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Removing stop words from a sentence using NLTK

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = """We are learning Natural Language Processing as part of
                  Fundamentals of Machine Learning in our second year B.Tech."""

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```

```
['We', 'are', 'learning', 'Natural', 'Language', 'Processing', 'as', 'part', 'of
', 'Fundamentals', 'of', 'Machine', 'Learning', 'in', 'our', 'second', 'year', '
B.Tech', '.']
['We', 'learning', 'Natural', 'Language', 'Processing', 'part', 'Fundamentals',
'Machine', 'Learning', 'second', 'year', 'B.Tech', '.']
```


Write a simple script to
remove punctuations.

Text Normalization

- Before almost any natural language processing of a text, the text has to be normalized.
- At least three tasks are commonly applied as part of any normalization process:
  1. Tokenizing (segmenting) words
  2. Normalizing word formats
  3. Segmenting sentences

Tokenization

- Common task in NLP
- Foremost step
- It is a way of separating a piece of text into smaller units called Tokens.
- Tokens are the building blocks of Natural Language
  - Tokens can be words, characters or subwords
  - Divided into 3 types
    - Word
    - Character
    - subword
- Goal – the creation of Vocabulary
  - Vocabulary – set of unique tokens in the corpus

Example

- word tokenization for the sentence: "Never give up"
  - ["Never", "give", "up"]
- Character tokenization for "smarter" is
  - ['s','m','a','r','t','e','r’]
- Subword tokenization for "smarter" is
  - ["smart", "er"]

Word Tokenization

- Most commonly used tokenization
- Splits a piece of text into individual words based on a certain delimiter
- Methods to perform tokenization
  - Using python's split() function
  - Using regular expressions
  - Using NLTK

Using Python's split() function

```
text = '''Once there lived a greedy man in a small town. He was very rich, and he loved gold and all
things fancy. But he loved his daughter more than anything. One day, he chanced upon a fairy. The
fairy's hair was caught in a few tree branches. He helped her out, but as his greediness took over, he
realised that he had an opportunity to become richer by asking for a wish in return (by helping her
out). The fairy granted him a wish. He said, "All that I touch should turn to gold." And his wish was
granted by the grateful fairy.'''

#word tokenization
tokens = text.split()
print(tokens)
print("No.of tokens : ", len(tokens))

#sentence tokenization
sentences = text.split('.')
print(sentences)
print("No.of sentences : ", len(sentences))
```

Tokenization using Regular Expressions

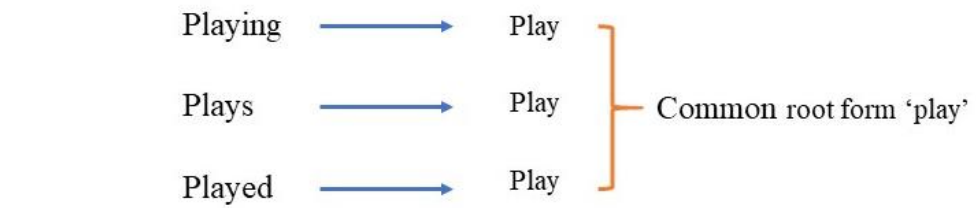
```
import re

#word tokenization
tokens = re.findall("[\w']+", text)
print(tokens)
print("No.of tokens : ", len(tokens))

#sentence tokenization
sentences = re.compile('[.?!] ').split(text)
print(sentences)
print("No.of sentences : ", len(sentences))
```

Tokenization using NLTK

```
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text)
print(tokens)
print("No.of tokens : ", len(tokens))

from nltk.tokenize import sent_tokenize
sentences = sent_tokenize(text)
print(sentences)
print("No.of sentences : ", len(sentences))
```

Word Normalization, Stemming and Lemmatization

- used to prepare text, words, and documents for further processing
- Stemming and Lemmatization helps us to achieve the root forms of inflected words



am, are, is → be

Car cars, car's, cars' → car

Using above mapping a sentence could be normalized as follows:

the boy's cars are different colors → the boy car be differ color

Stemming

- helps us to achieve the root forms of inflected words.
- Stem (root) is the part of the word to which you add inflectional (changing/deriving) affixes such as (-ed,-ize, -s,-de,mis).
- stemming a word or sentence may result in words that are not actual words. Stems are created by removing the suffixes or prefixes used with a word.
- A computer program that stems word is called a stemming program, or stemmer
- PorterStemmer is stemming algorithm present in NLTK which uses Suffix Stripping
- It does not follow linguistics rather a set of 5 rules for different cases that are applied in phases to generate stems.

```
from nltk.stem import PorterStemmer
```

```
#create an object of class PorterStemmer
porter = PorterStemmer()
print(porter.stem("cats"))
print(porter.stem("troubles"))
```

```
cat
troubl
```

create a function which takes a sentence and returns the stemmed sentence.

```
from nltk.stem import PorterStemmer
porter = PorterStemmer()

sentence="Pythoners are very intelligent and work very pythonly and now they are pythoning their way to
success."

from nltk.tokenize import sent_tokenize, word_tokenize
def stemSentence(sentence):
    token_words=word_tokenize(sentence)
    print(token_words)
    stem_sentence=[]
    for word in token_words:
        stem_sentence.append(porter.stem(word))
        stem_sentence.append(" ")
    return "".join(stem_sentence)

x=stemSentence(sentence)
print("Sentence after stemming :", x)
```

```
['Pythoners', 'are', 'very', 'intelligent', 'and', 'work', 'very', 'pythonly', 'and', 'now', 'they', 'are', 'pythoning', 'their', 'way', 'to', 'success', '.']
Sentence after stemming : python are veri intellig and work veri pythonli and now they are python their way to success .
```

Lemmatization

- Lemmatization reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma
- For example, **runs, running, ran** are all forms of the word **run**, therefore run is the **lemma** of all these words.
- As lemmatization returns an actual word of the language, it is used where it is necessary to get valid words.
- Python NLTK provides **WordNetLemmatizer** that uses the **WordNet** Database to lookup lemmas of words.

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

print(wordnet_lemmatizer.lemmatize("cats"))
print(wordnet_lemmatizer.lemmatize("troubles"))
```

```
cat
trouble
```

Sentence Segmentation

- Sentence segmentation is another important step in text processing.
- The most useful cues for segmenting a text into sentences are punctuation, like periods, question marks, and exclamation points.

Standardization of Data

The common operations performed to standardize the data are

- Removal of duplicate whitespaces and punctuation.
- Accent removal
- Capital letter removal
- Removal or substitution of special characters/emojis (e.g.: remove hashtags).
- Substitution of contractions (very common in English; e.g.: 'I'm'→'I am').
- Transform word numerals into numbers (eg.: 'twenty three'→'23').
- Substitution of values for their type (e.g.: '$50'→'MONEY').
- Acronym normalization (e.g.: 'US'→'United States'/'U.S.A') and abbreviation normalization (e.g.: 'btw'→'by the way').
- Normalize date formats, social security numbers
- Spell correction — this is very important if you're dealing with open user inputs, such as tweets, IMs and emails.
- Removal of gender/time/grade variation with Stemming or Lemmatization.
- Substitution of rare words for more common synonyms.
- Stop word removal (more a dimensionality reduction technique than a normalization technique).

Refer to the below link for basic text cleaning

HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2022/01/TEXT-CLEANING-METHODS-IN-NLP/