# Natural Language Processing (A Classical Approach)

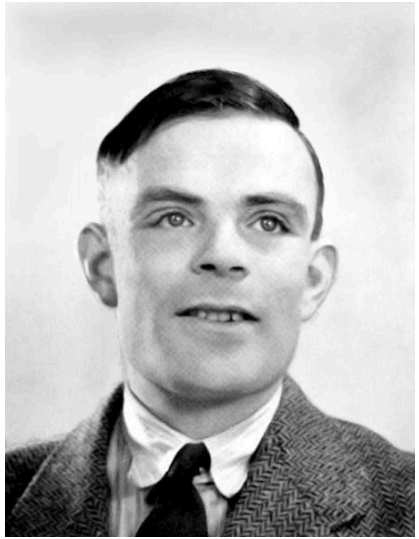
# Natural Language Processing

**Natural-language processing (NLP)**

- An area of computer science and artificial intelligence for
- Interactions between computers and human languages
- Program computers to process natural language data

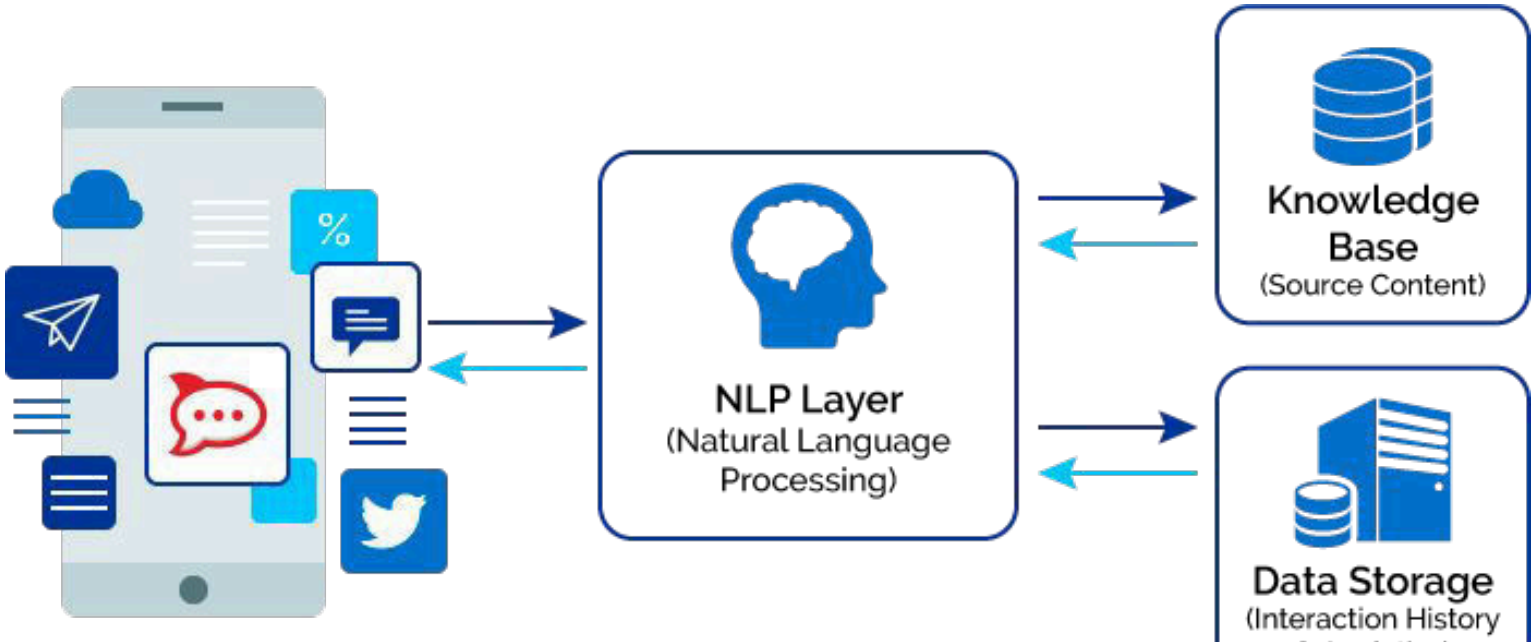
# Natural Language Processing

In 1950, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence.



# Natural Language Processing

**Basic Structure of a NLP application (chatbot considered below)**

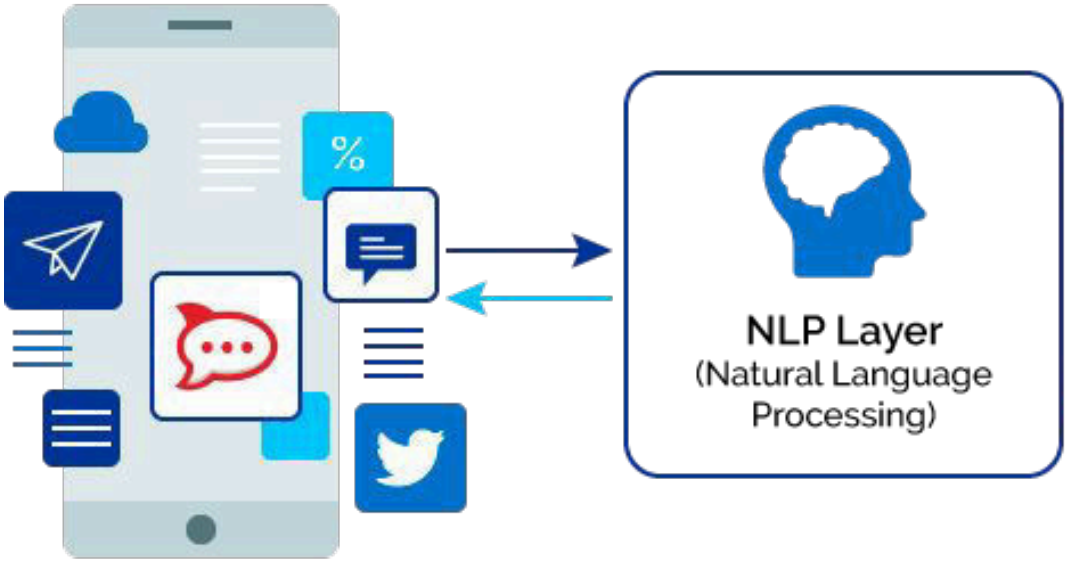


# Natural Language Processing


**Knowledge Base** – A database used to equip chatbots with the information needed to respond to queries of customers request.


**Data Store** – Contains interaction history of chatbot with users.

# Natural Language Processing




**NLP Layer** – Translates users' queries (free form) into information that can be used for appropriate responses.

**Application Layer** – The application interface that is used to interact with the user.

# Natural Language Processing – Applications

**Speech Recognition**

# Natural Language Processing – Applications

## Text Classification

# Natural Language Processing – Applications

## Caption Generation using VQA (Visual Question Answering)

1. Choose an Image and a Question

Kitchen

Image:

Question:

What is in the bowls on the island?

2. Run a model

Predicted Answers

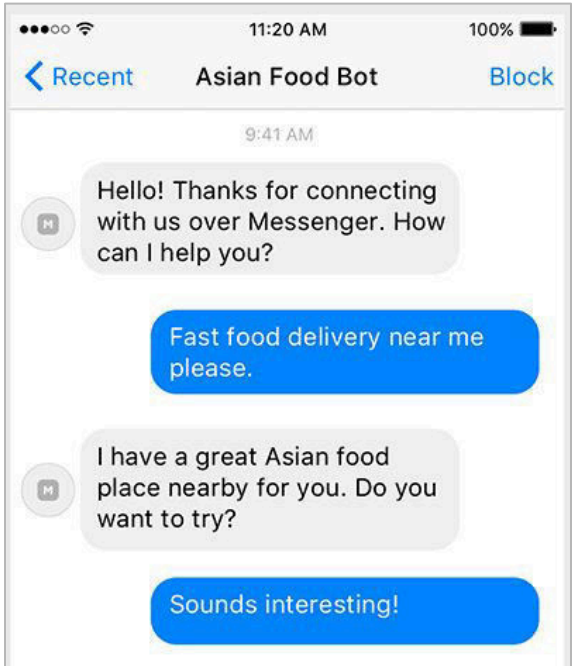
| Score | Answer |
|---|---|
| 45.7% | bread |
| 36.7% | fruit |
| 5.1% | food |
| 3.2% | cereal |
| 1.9% | bananas |

# Natural Language Processing – Applications

## Machine Translation

# Natural Language Processing – Applications

## Question Answering



# Natural Language Processing – Tools

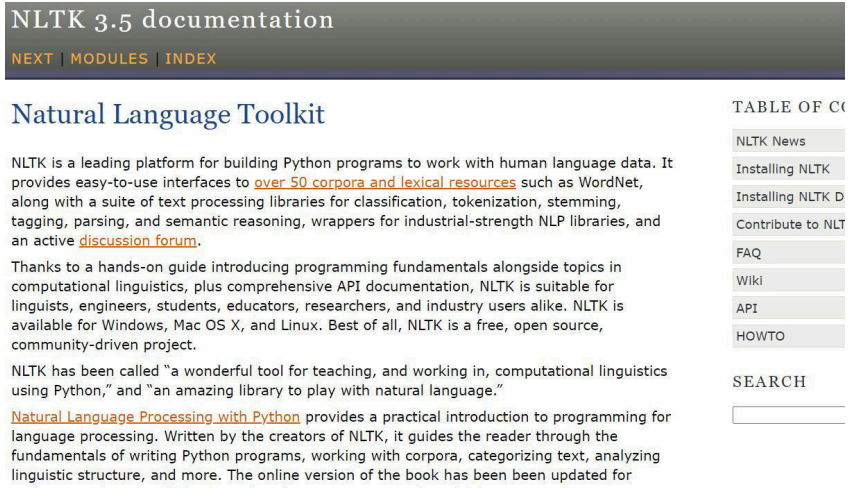
Here are some of the most popular Natural Language Processing Tools.

# Natural Language Processing – Tools

- Stanford's Core NLP Suite – a set of stable and well-tested natural language processing tools, widely used by various groups in academia, industry, and government.

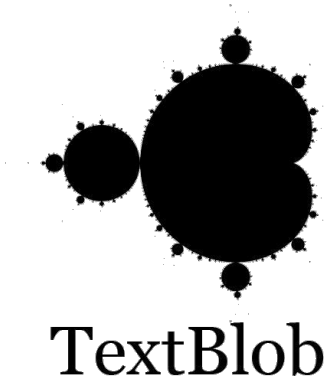
# Natural Language Processing – Tools

- [Natural Language Toolkit (NLTK)](#) – A suite of libraries and programs for symbolic and statistical natural language processing for English.



# Natural Language Processing - Tools

- Text Blob - A Python library for processing textual data and NLP tasks like part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.



# Natural Language Processing - Tools

- SpaCy - An open-source software library for advanced natural language processing.

# Natural Language Processing - Tools

- Pytorch-NLP - Library for NLP in Python with pre-trained embeddings, samplers, dataset loaders, metrics, neural network modules and text encoders.

# Natural Language Processing – Tools

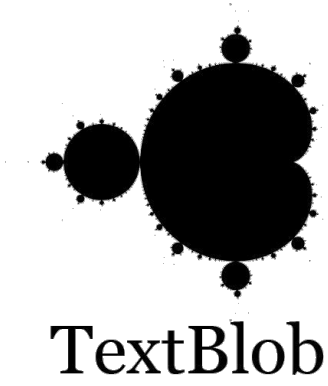
- OpenNLP by Apache – A machine learning based toolkit for the processing of natural language text.

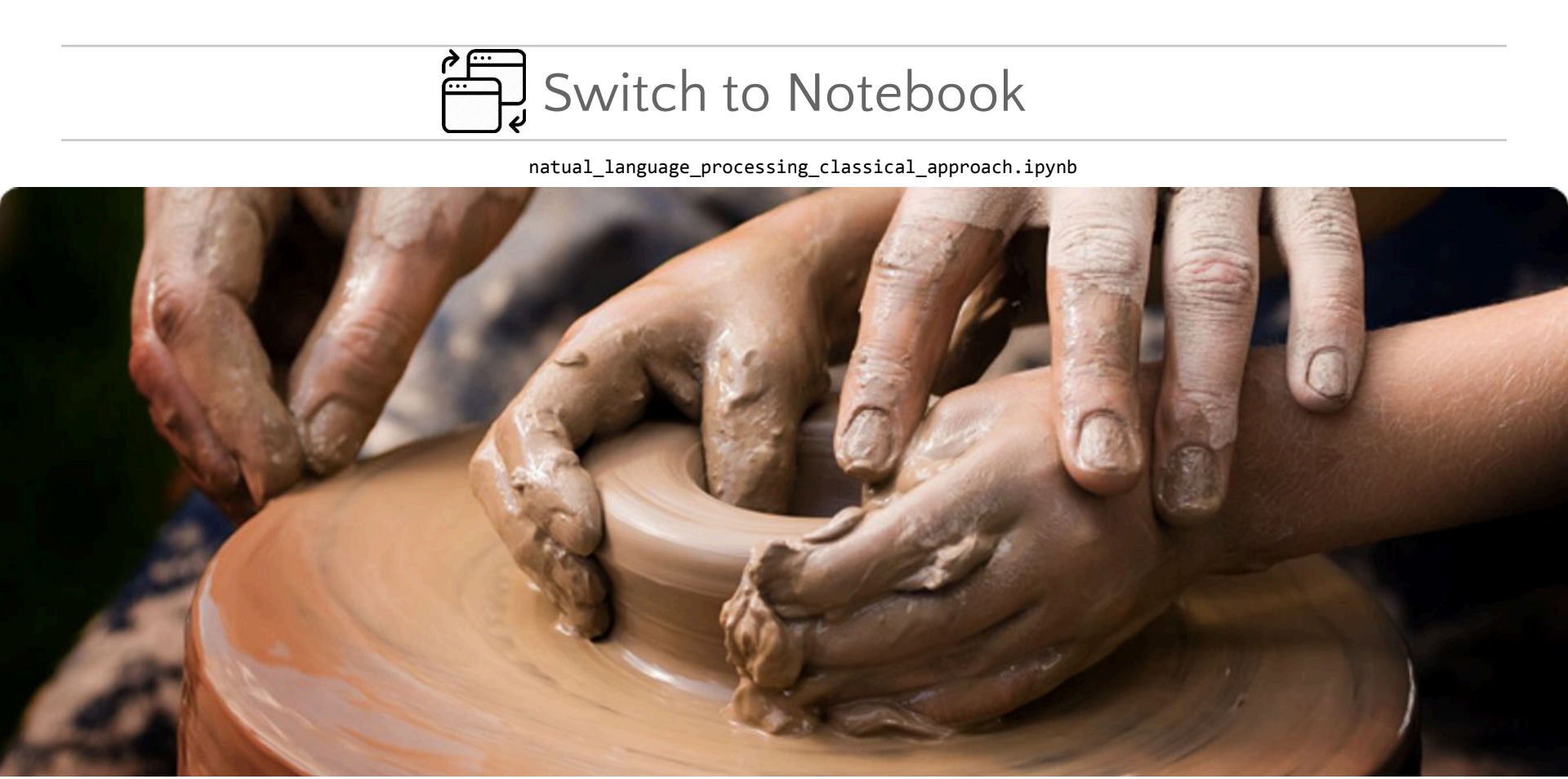
# Create a Quiz using TextBlob

# Create a Quiz using TextBlob

Let us use the TextBlob library of Python to Build a program that makes a quiz out of a provided text.

It is basically a usage of **NER** - Named-Entity Recognition.





# Switch to Notebook

natual_language_processing_classical_approach.ipynb

# Create a Quiz using TextBlob

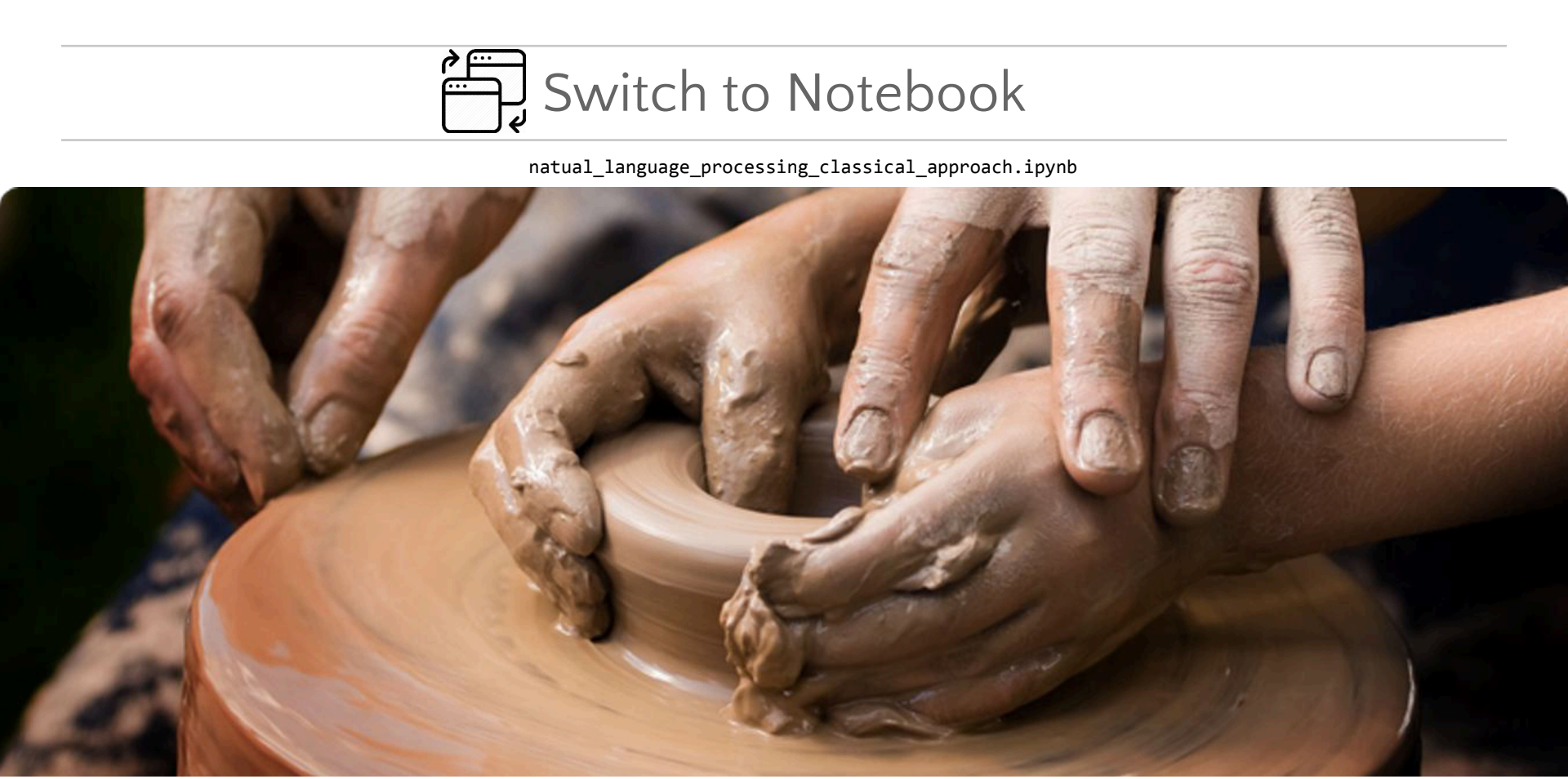
Let us understand a few things about the TextBlob API:

- **text.sentences** - gives the sentences in a text
- **sentences.tags** - gives the tags for each of the word in sentence

# Create a Quiz using TextBlob

Now to generate our quiz we will:

- Extract each sentence
- Replace some the nouns and proper nouns with a blank
- And remove only after the fourth word in the sentence



# Switch to Notebook

natual_language_processing_classical_approach.ipynb

# Find Related Posts using Scikit-Learn

# Find Related Posts using Scikit-Learn

To find related posts from a bunch of posts, we first need to learn how to turn text into something on which we can calculate similarity.

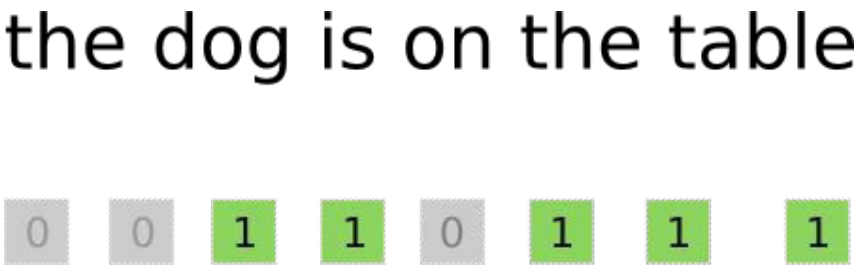
How to do it ??

By using the Bag of Words Approach

# Find Related Posts using Scikit-Learn

**Bag of Words Approach** - It ignores order of words, and uses word counts as their basis.

In this model, a text (sentence or document) is represented as bag (multiset) of its words, disregarding grammar and word order but keeping multiplicity.



# Find Related Posts using Scikit-Learn

**Vectorization -** For each word in the post, its occurrence is counted and noted in a vector. This step is called vectorization.

The vector is typically huge as it contains as many elements as words occur in the whole dataset.

# Find Related Posts using Scikit-Learn

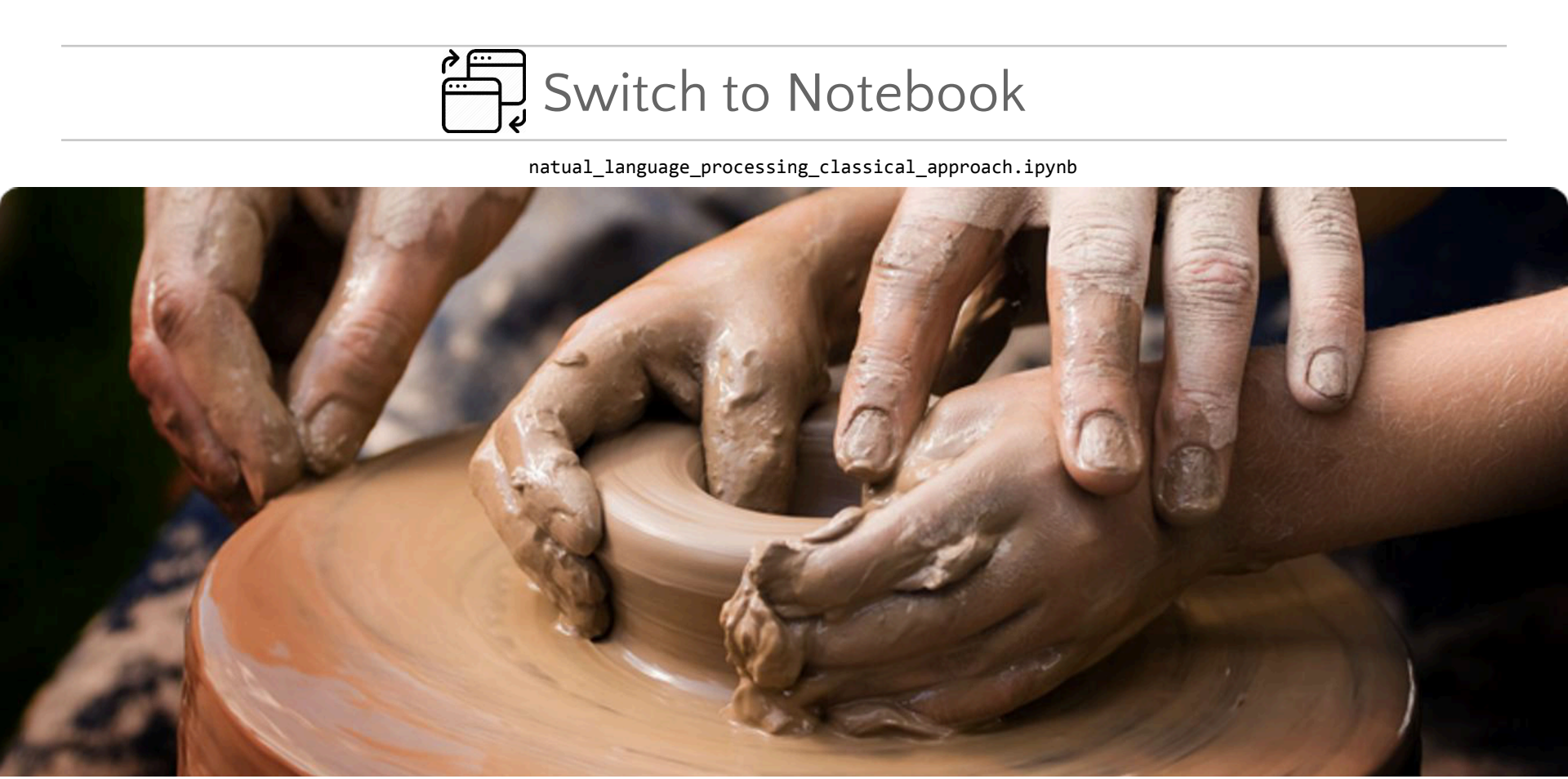
**Vectorization example -** For the two statements "How to format my hard disk" and " Hard disk format problems " the vectors are shown below in a

**Term Document Matrix**

| Word | Occurence in post 1 | Occurence in post 2 |
|---|---|---|
| disk | 1 | 1 |
| format | 1 | 1 |
| how | 1 | 0 |
| hard | 1 | 1 |
| my | 1 | 0 |
| problems | 0 | 1 |
| to | 1 | 0 |

# Find Related Posts using Scikit-Learn

Now let's see how to implement vectorization using Scikit-Learn



# Switch to Notebook

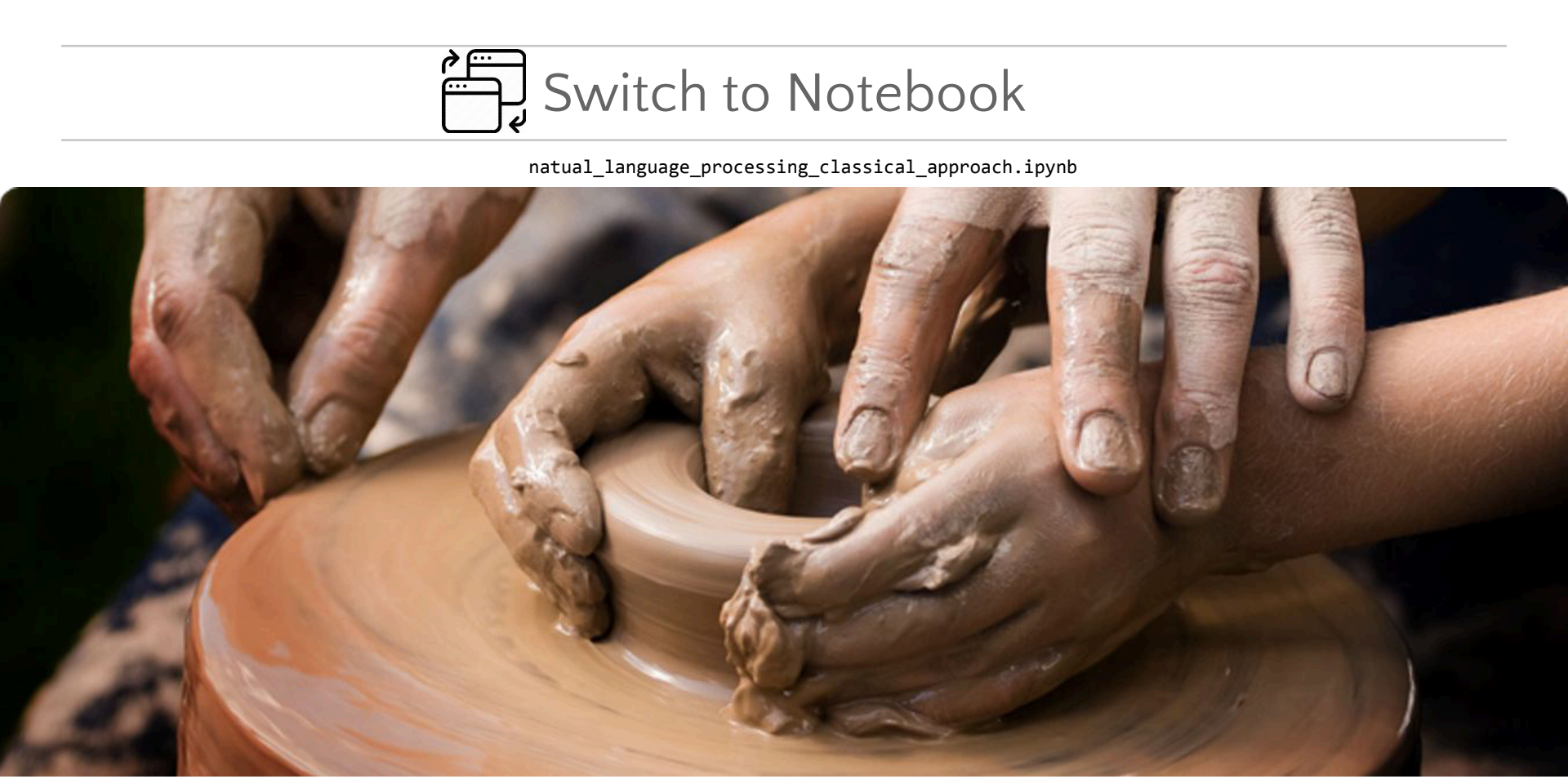
natual_language_processing_classical_approach.ipynb

# Analyse collection of Text Documents using Scikit Learn

# Analyse collection of Text Documents using Scikit Learn

Here is a quick checklist of what we would do to analyse these collection of text documents:

- Load file contents and categories
- Extract feature vectors suitable for machine learning
- Train linear model to perform categorization
- Use grid search to find good configuration of feature extraction components and the classifier



# Switch to Notebook

natual_language_processing_classical_approach.ipynb

# Analyse collection of Text Documents using Scikit Learn

- Occurrence count is a good start but there is an issue
  - Longer documents will have higher average count values than shorter documents

# Analyse collection of Text Documents using Scikit Learn

To avoid these potential discrepancies we:

- Divide the number of occurrences of each word in a document by the total number of words in the document:
    - These new features are called **tf for Term Frequencies**.

# Analyse collection of Text Documents using Scikit Learn

**How can we improve tf ?**

# Analyse collection of Text Documents using Scikit Learn

- Downscale weights for words that occur in many documents in the corpus and are therefore less informative than those that occur only in a smaller portion of the corpus.
    - This downscaling is called **tf–idf for “Term Frequency times Inverse Document Frequency”**.

# Analyse collection of Text Documents using Scikit Learn

- Tf-idf weight is often used in
  - Information retrieval and
  - Text mining
- This weight is a used to evaluate
  - How important a word is to a
  - Document in a collection or corpus

# Analyse collection of Text Documents using Scikit Learn

**Term Frequency**

- Measures how frequently a term occurs in a document
- It is possible that a term would appear
  - Much more times in long documents than shorter ones
  - This is why we normalize TF

**TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)**

# Analyse collection of Text Documents using Scikit Learn

**Inverse Document Frequency**

- Measures how important a term is
- In TF, all terms are considered equally important
- How ever some words and stop words appears lot of time
    - But have least importance
- In IDF we weight down frequent terms
    - And scale up rare terms

**IDF(t) = log_e(Total number of documents / Number of documents with term t in it)**

# Analyse collection of Text Documents using Scikit Learn

**Example**

- Consider a document containing 100 words and the word cat appears 3 times
- The term frequency(tf) for cat is
  - (3 / 100) = 0.03

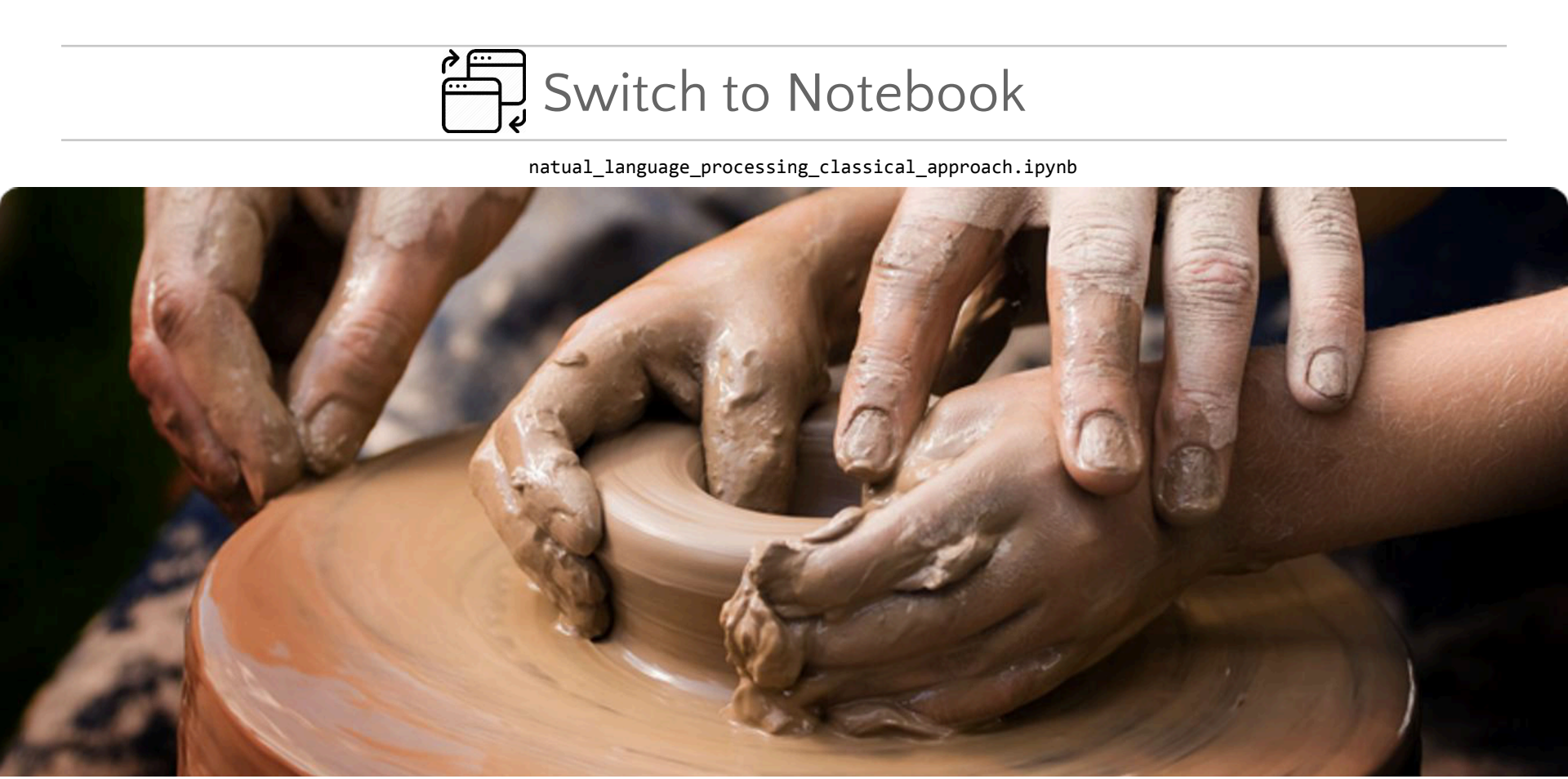
# Analyse collection of Text Documents using Scikit Learn

**Example**

- Now, assume we have 10 million documents and
  - The word cat appears in 1,000 of these
- The inverse document frequency(idf) is
  - log(10,000,000 / 1,000) = 4
- Tf-idf weight is the product of tf and idf
  - 0.03 * 4 = 0.12

# Analyse collection of Text Documents using Scikit Learn

Now let us apply tf-idf to our example



# Switch to Notebook

natual_language_processing_classical_approach.ipynb

# Natural Language Processing – Stanford NLP

# Natural Language Processing – Stanford NLP

## Overview of Stanford Core NLP

# Natural Language Processing – Stanford NLP

***Stanford CoreNLP provides a set of human language technology tools.***

It provides:

- Base forms of words,
- Parts of speech,
- Whether they are proper nouns,
- Mark up the structure of sentences in terms of phrases and syntactic dependencies,
- Indicate which noun phrases refer to the same entities, indicate sentiment

# Natural Language Processing – Stanford NLP

Choose Stanford CoreNLP if you need:

- Integrated NLP toolkit
- Fast, robust annotator for arbitrary texts
- Highest quality of text analytics
- Support for major languages
- Available APIs for most programming languages
- Ability to run as a simple web service

# Natural Language Processing – Stanford NLP

**Programming languages and operating systems**

Stanford CoreNLP can be used via:

- Command-line
- Object-oriented simple API,
- Third party APIs in other languages
- A web service

It works on Linux, macOS, and Windows.

Other NLP Tools and Techniques to explore

# Other NLP Tools and Techniques to explore

**Word2Vec**

The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.

Here is a Keras implementation of this technique by **Francois Chollet**.

# Other NLP Tools and Techniques to explore

**DialogFlow**

This is an advanced development suite for creating conversational AI applications, including chatbots, voicebots, and IVR bots from Google.

Create your own projects here.

# Other NLP Tools and Techniques to explore

**NLP using RNN**

A common and modern approach for natural language tasks is to use recurrent neural networks, we will explore them next!

# Questions?