

# **Thapar Summer School 2022**

**(4<sup>th</sup> July 2022)**

## **NLP with NLTK and PYTHON**

**Dr. Nitin Arvind Shelke**

# Contents (Day 1)

- **NLP Introduction**
- **NLP Applications**
- **Lifecycle of NLP**
- **Libraries Required for NLP**
- **NLP Operations with NLTK and PYTHON**
- **Basic Strings Method for NLP**
- **Making Decisions and Taking Control**
- **NLTK Methods for NLP**
  - Counting Vocabulary
  - Searching Text (Concordance)
  - Searching Text (Similar)
  - Searching Text (common\_contexts)
  - Collocation

# Contents (Day 1)

- **Frequency Distributions and Conditional frequency distribution**
- **Extraction of Text Data from PDF and DOC file**
- **Text Corpus**
  - Gutenberg Corpus
  - Web and Chat Text Corpus
  - Brown Corpus
- **Lexical Resources**
  - Wordlist Corpora
  - Name Corpus
  - Comparative Wordlists
  - WordNet
  - Stopword
- **Hands On Case Study**

# About NLTK

- The Natural Language Toolkit, or more commonly NLTK, is the most popular library for natural language processing (NLP) which was written in the Python programming language.
- A software package for manipulating linguistic data and performing NLP tasks
- It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania.

.

# What is NLP?

- Natural Language Processing (NLP) is a field of artificial intelligence that allows computers to analyze and understand human language.
- Study of interaction between computers and human languages
- Process of deriving meaningful information from natural language text

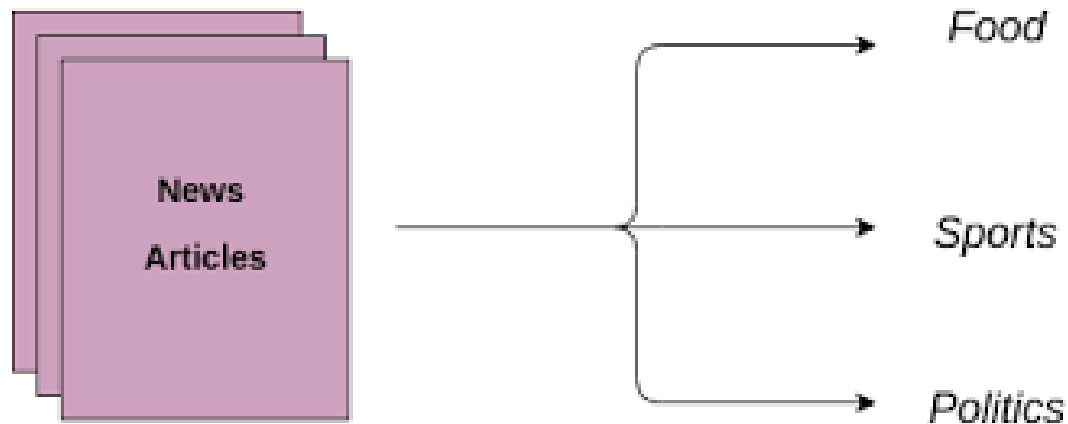


# Need of NLP

- Text communication is one of the most popular forms of day to day conversion. We chat, message, tweet, share status, email, write blogs, share opinion and feedback in our daily routine.
- By utilizing NLP and its components, one can organize the massive chunks of text data, perform numerous automated tasks and solve a wide range of problems in the following applications,
- Amazon can understand user feedback or review on the specific product.
- BookMyShow can discover people's opinion about the movie.
- Use in Search Engine, Social Website, Spam Filters, Automatic summarization, E-Commerce Websites, chatbot and many....

# NLP Applications

**Text Classification:** It also is the process of categorizing text into organized groups. By using NLP, text classifiers can automatically analyze text and then assign a set of pre-defined tags or categories based on its content.



# NLP Applications

## Fake news detection:

Since all the news we encounter in our day-to-day life is not authentic, how do we categorize if the news is fake or real?

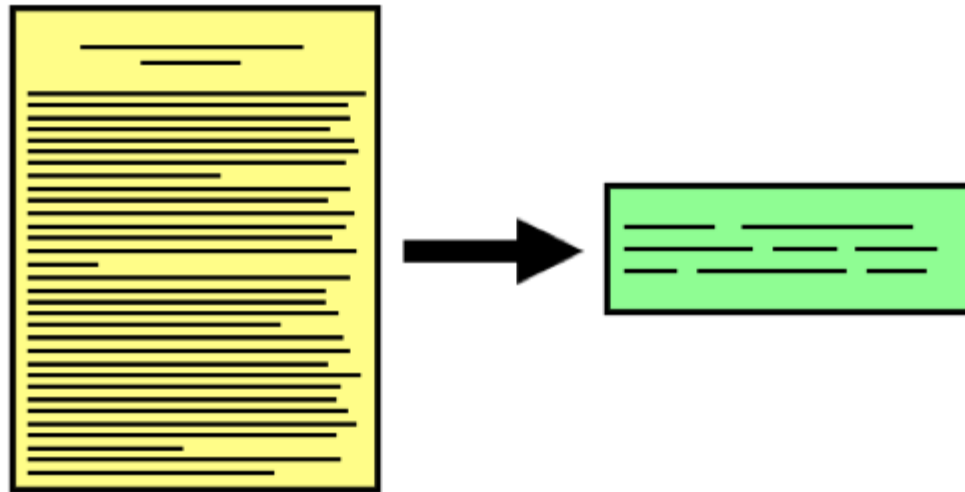
Build a system to identify to identify when an article might be fake news.





# NLP Applications

**Automatic Summarization:** Text summarization can be defined as the technique to create short, accurate summary of longer text documents.



# NLP Applications

**Sentiment Analysis** : Sentiment analysis is the automated process of classifying opinions in a text as positive, negative, or neutral.

- Monitor sentiments in social media
- Analyze sentiment of product reviews
- Analyze emotions in survey responses

SENTIMENT ANALYSIS



Discovering people opinions, emotions and feelings about  
a product or service

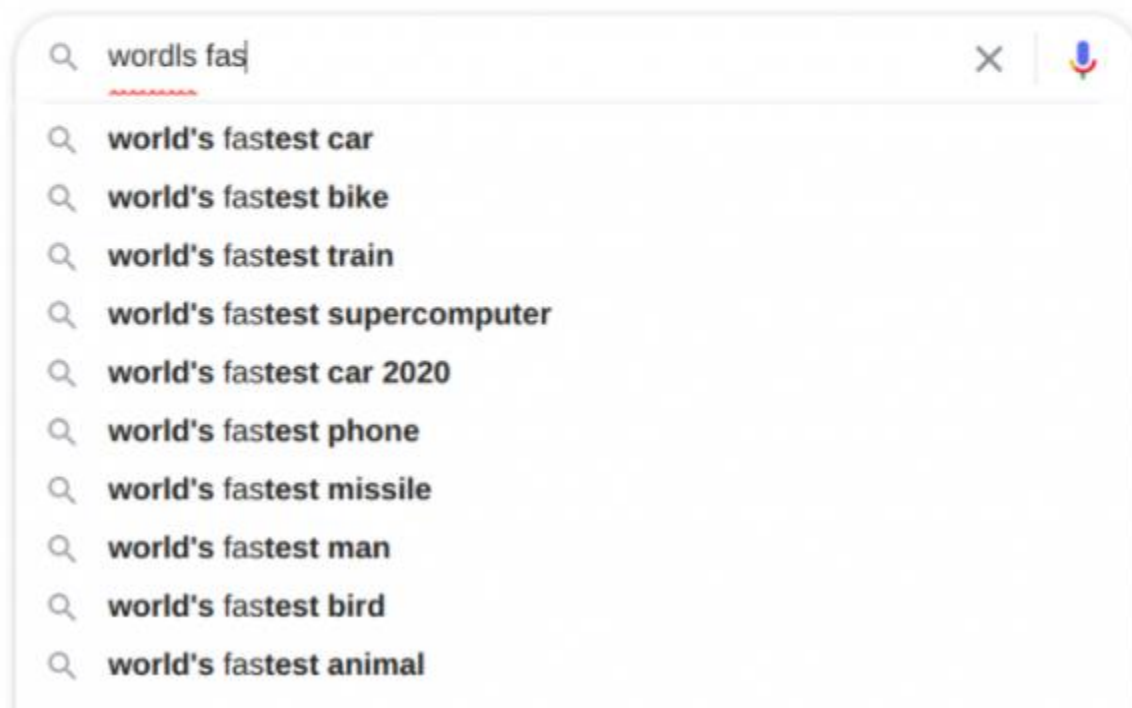
# NLP Applications

## Web Scrapping



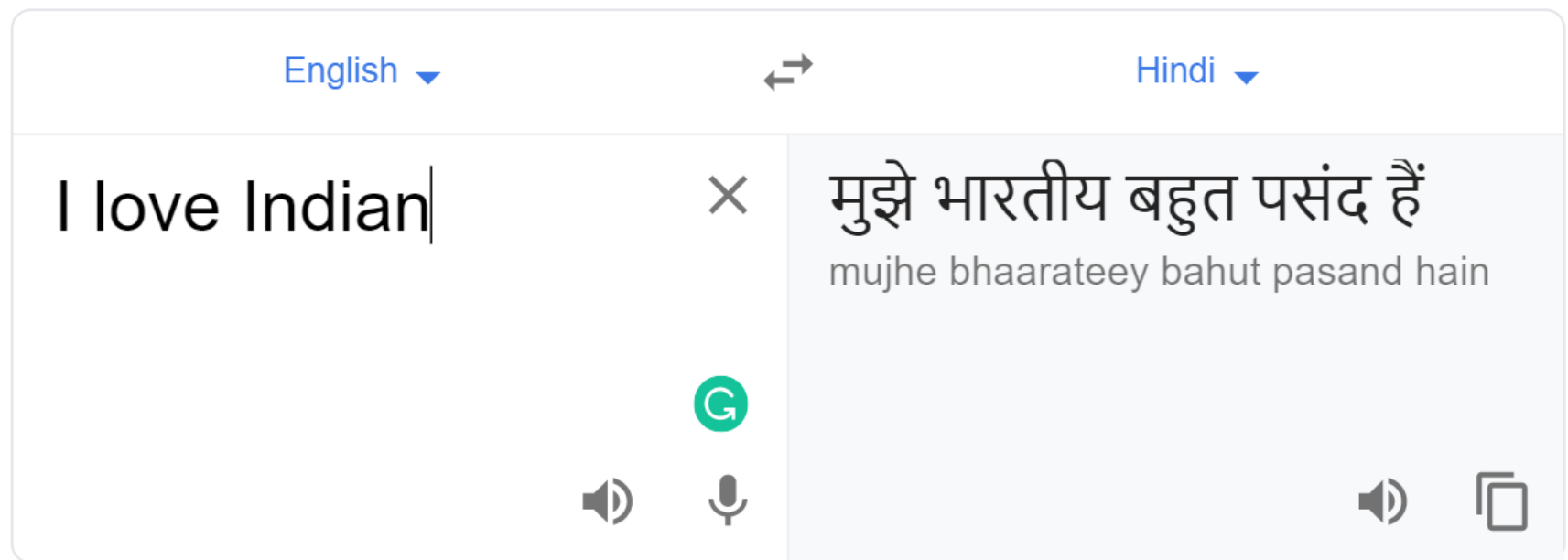
# NLP Applications

## Search Autocorrect and Autocomplete



# NLP Applications

**Machine Translation:** Machine Translation is the procedure of automatically converting the text in one language to another language while keeping the meaning intact.



[Open in Google Translate](#)

[Feedback](#)

# NLP Applications

**Chatbots:** A chatbot is a computer program that simulates human conversation. Chatbots use NLP to recognize the intent behind a sentence, identify relevant topics and keywords, verbs, and even emotions, and come up with the best response based on their interpretation of data.



# NLP Applications


## Grammar Checkers

The most common type of marketing channel is the wholesale market.

Varies kinds of **produce** are supplied from different areas are assembled at one place and sold through smaller regional markets, etc. Fruits and vegetables through market handling and transport methods.

Replace the word

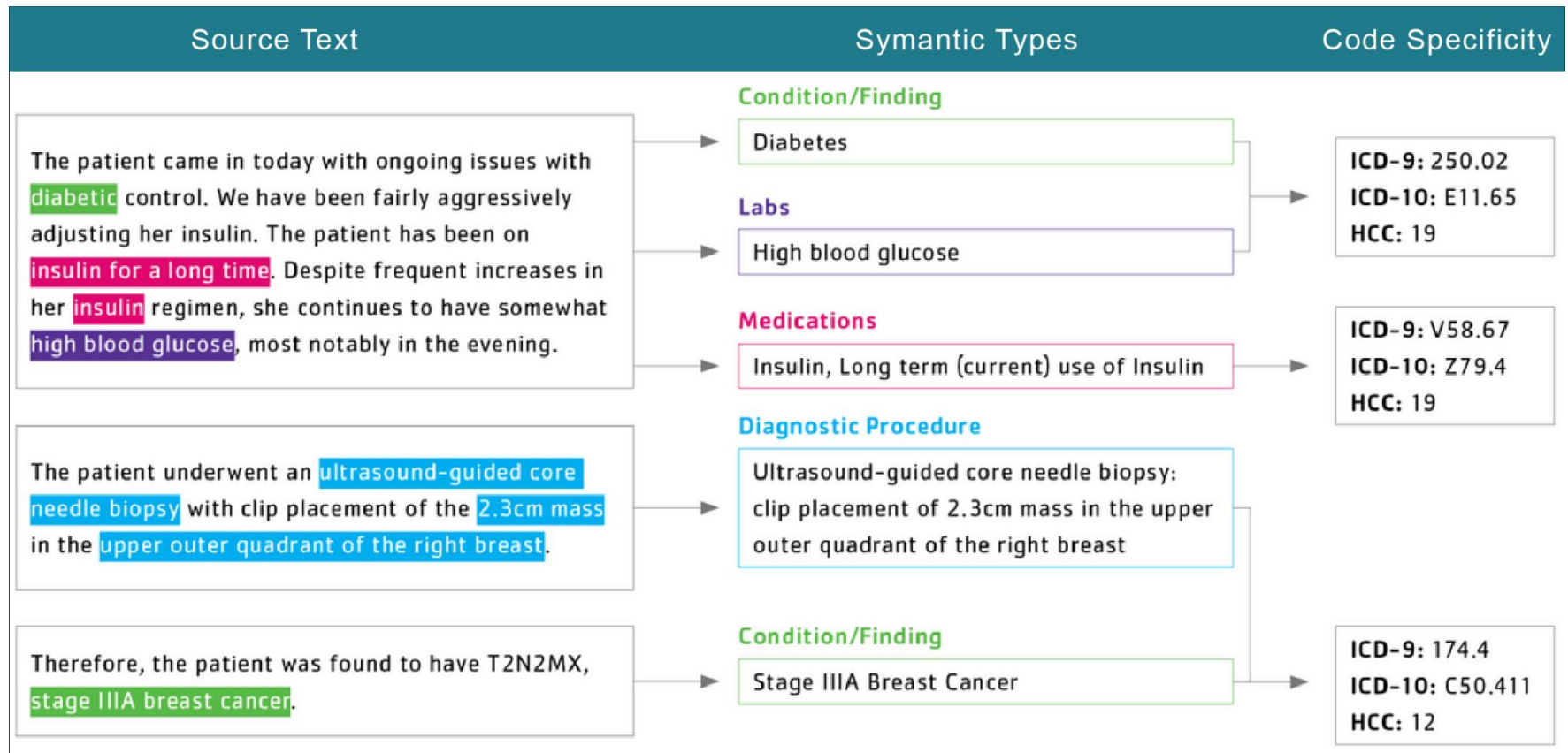
**products**

 Dismiss

Suggested by Grammarly

# NLP Applications

## Information Extraction from Clinical Note





# Input/Output for NLP

## ***Input / Output***



- Named entities
- Sentiment
- Category
- Information extraction
- Intent identification
- Translated text
- Summary

# Lifecycle of NLP

- Problem Identification
- Text Data Collection
- Text Data Pre-processing
- Exploratory Data Analysis (EDA)
- Feature Extraction
- Dataset Splitting
- Model Creation
- Model Testing
- Deployment

# Libraries Required for NLP

- **Primary Library**

NLTK (Mostly used python based)

Stanford NLP toolkit (Java based)

Genism

TextBlob

spaCy

- **Secondary Library**

NumPy

Matplotlib

Pandas

# NLP Operations with NLTK and PYTHON

- Operations on Text Data
- Accessing Text Corpora and Lexical Resources
- Text Data Preprocessing (Tokenization, Stop word Removal, Stemming and Lemmatization, regular Expression)
- POS Tagging
- NER
- Feature Extraction and Word Embeddings (BoW, TFIDF, Word2Vec, GLOVE)
- Accessing Text from the Web
- Web Scrapping
- Text Classification

# NLTK Installation

- NLTK requires Python versions above 2.7, 3.4, 3.5, or 3.6
- NLTK Library included in ANACONDA (No need to install separately)
- Alternatively, It can be directly installed using pip in terminal using following command  
`pip install nltk`

# NLTK Package Installation

Once you have installed NLTK, You have to download the NLTK packages.

```
>>>import nltk
```

```
>>>nltk.download( )
```



# Loading Book

Once the data is downloaded to our machine, You can load book module with the help of following line of code,

```
>>>>from nltk.book import *
```

# Basic Strings Method for NLP

# Slicing

#lower()

#upper()

#title()

#capitalize()

# split()

#join()

#count()

#find()

#strip()

#replace()

#startswith()

#islower()

#isupper()

#isalpha()

#isalnum()

#isdigit()

#istitle()

#in or not in

#len



# Counting Vocabulary

To find the length of a text,

```
>>>len(text3)
```

To find unique words in the text we use *set* command as follows

```
>>> set(text1)
```

# Counting Vocabulary

The list of unique terms can be sorted using *sorted* command as follows:

```
>>>sorted(set(text1))
```

The number of unique words can be counted by using *len* and *set* command as follows:

```
>>> len(set(text1))
```

# Counting Vocabulary

The frequency of any word in the text can be computed using *count* command as follows:

```
>>>text1.count("the")
```

The percentage of text taken up by a word can be computed as follows:

```
>>>100 * text1.count("the") / len(text1)
```

# Counting Vocabulary

Lexical diversity or richness of the text tells that how many times each word occurs on an average in the text. It is computed as follows:

```
>>> len(set(text3)) / len(text3)
```

# Counting Vocabulary

Lexical Diversity function is as follows,

```
>>> def lexical_diversity(text):  
...     return len(set(text)) / len(text)
```

```
...
```

```
>>> def percentage(count, total):  
...     return 100 * count / total
```

```
...
```

Call the function,

```
>>> lexical_diversity(text3)
```

```
>>> percentage(text4.count('a'), len(text4))
```

# Making Decisions and Taking Control

`s.startswith(t)` test if `s` starts with `t`

`s.endswith(t)` test if `s` ends with `t`

`t in s` test if `t` is contained inside `s`

`s.islower()` test if all cased characters in `s` are lowercase

`s.isupper()` test if all cased characters in `s` are uppercase

`s.isalpha()` test if all characters in `s` are alphabetic

`s.isalnum()` test if all characters in `s` are alphanumeric

`s.isdigit()` test if all characters in `s` are digits

`s.istitle()` test if `s` is titlecased (all words in `s` have initial capitals)

# Making Decisions and Taking Control

Syntax [w for w in text if *condition*] // *with one condition*

```
>>> [w for w in set(text) if w.endswith('ab')]
```

```
>>> [term for term in set(text) if 'me' in term]
```

```
>>> [item for item in set(text) if item.istitle()]
```

```
>>> [item for item in set(sent) if item.isdigit()]
```

# Making Decisions and Taking Control

## Two Conditions

```
>>> [wd for wd in set(text) if wd.istitle() and len(wd) > 5]
```

```
>>> [w for w in set(text) if not w.islower()]
```

```
>>> [t for t in set(text) if 'th' in t or 'o' in t]
```

```
>>> [t for t in set(text) if 'th' in t and 'o' in t]
```

```
>>> [len(w) for w in text]
```

```
>>> [w.upper() for w in text]
```

```
>>> len(set([word.lower() for word in text]))
```

```
>>> len(set([word.lower() for word in text if word.isalpha()])))
```



# Fine Grained Selection of Words

Write a code to find the words from the vocabulary of the text 2 that are more than 10 characters long.

```
>>> vocb = set(text2)
```

```
>>> words_formed = [x for x in vocb if len(x) > 10 ]
```

```
>>> sorted(words_formed )
```

# Searching Text (Concordance)

Concordance : A concordance allows us to shows every occurrence of a given word, together with some context.

```
>>> text1.concordance("monstrous")
```

# Searching Text (Similar)

Similar: A similar method permits us to see what words appear in a similar range of contexts

```
>>> text1.similar("monstrous")
```

# common\_contexts()

The `common_contexts()` method allows us to examine the contexts that are shared by two or more words.

```
>>text2.common_contexts(["monstrous", "very"])  
a_pretty is_pretty am_glad be_glad a_lucky
```

# Frequency Distributions

frequency distribution tells us the frequency of each vocabulary item in the text (i.e., How many times particular word appear).

```
>>> from nltk import FreqDist
```

```
>>> fdist1 = FreqDist(text1)
```

```
>>> fdist1
```

```
>>> fdist1.keys()
```

```
>>> fdist1['whale']
```

# Frequency Distributions

## Extra Methods

`fdist['monstrous']`

`fdist.freq('monstrous')`

`fdist.N()`

`fdist.most_common()`

`fdist.keys()`

`fdist.values()`

`Fdist.items()`

`fdist.max()`

`fdist.tabulate()`

`fdist.plot(50,cumulative=True)`

# Collocation

**Collocation:** A collocation is a sequence of words that occur together unusually often. Thus red wine is a collocation, whereas the wine is not.

# Case Study (Sample Example)

- Input = “Game of Thrones is an American fantasy drama television series created by David Benioff and D. B. Weiss for HBO. It is an adaptation of A Song of Ice and Fire, George R. R. Martin's series of fantasy novels, the first of which is **A Game of Thrones**. The show was filmed in Belfast and elsewhere in the United Kingdom, Canada, Croatia, Iceland, Malta, Morocco, Spain, and the United States.[1] The series premiered on HBO in the United States on April 17, 2011, and concluded on May 19, 2019, with 73 episodes broadcast over eight seasons. Set on the fictional continents of Westeros and Essos, Game of Thrones has several plots and a large ensemble cast, and follows several story arcs. One arc is about the Iron Throne of the Seven Kingdoms, and follows **a web of** alliances and conflicts among the noble dynasties either vying to claim the throne or fighting for independence from it. Another focuses on the last descendant of the realm's deposed ruling dynasty, who has been exiled and is plotting a return to the throne, while another story arc follows the Night's Watch, a brotherhood defending the realm against the fierce peoples and legendary creatures of the North.”



# Case Study (Cont.)

To be able to apply nltk functions we need to convert our text of type 'str' to 'nltk.Text'.

```
>>text = nltk.Text( Input.split() )
```

Now use previously studied methods that we had seen earlier

```
>>text.common_contexts(['game', 'web']) #outputs a_of
```

**Thank  
You**

**[nitinashelke@gmail.com](mailto:nitinashelke@gmail.com)**