AI:

Simulating human intelligence to perform tasks that would require human intelligence at the first place.

AI Sub Fields:

* Machine learning
  + Completing tasks by using previous data.
  + Predicting output for a given data.
* Robotics
  + Using algorithms to solve questions about robot automation.
* NLP
  + Generation
    - Produce natural language output
    - Content determination
    - Document structuring
    - Aggregation: putting similar sentences together
    - Lexical choice
    - Referring expression generation: expression that helps identifying an object
    - Realisation: creating and optimizing correct text for grammar
  + Understanding
    - Understanding natural language and tagging each element with its properties
* Speech Recognition
  + Text to speech
    - Transforming text documents to speech.
  + Speech to text
    - Understanding speech and turning it into text.
* Expert systems
  + Systems that emulate decision making processes of experts
* Computer vision
  + Derive meaning from images
* Planning
  + Use AI to solve schedule problems

NLP Sub-fields

* Sentiment Analysis
  + Finding out if a text is positive negative or neutral
* Named Entity Recognition
  + Finding the types of words
* Summarization
  + Making a summary of a passage
* Topic Modeling
  + Connecting words with topics
* Part of Speech
  + Tagging each word as subject, verb, etc.
* Lemmatization and stemming
  + Stemming
    - Removing the suffixes
  + Lemmatization
    - Taking the root while looking at the context

Transformers:

Homogenized models that can be distributed among many servers. They can perform self-supervised learning and they can work with unlabeled data.

NLTK:

* Tokenizing
  + Dividing a given text into smaller parts called tokens in NLP tokens are words and punctuation.
  + Example Code:

|  |
| --- |
| import nltk  from nltk.tokenize import word\_tokenize  # Text to be tokenized  text = "NLTK provides various tokenization methods."  # Tokenize the text  tokens = word\_tokenize(text)  # Print the tokens  print(tokens) |

* Taking out the stopwords
  + Stopwords are words that carry no significant weight in the analysis
  + Example Code:

|  |
| --- |
| import nltk  from nltk.tokenize import word\_tokenize  from nltk.corpus import stopwords  # Text to be tokenized  text = "NLTK provides various tokenization methods."  # Tokenize the text  tokens = word\_tokenize(text)  # Set the list of stop words  stop\_words = set(stopwords.words('english'))  # Remove stop words from the tokens  filtered\_tokens = [token for token in tokens if token.lower() not in stop\_words]  # Print the filtered tokens  print(filtered\_tokens) |

* Lemmatizing
  + Lemmatizing removes the suffixes while looking at the context
  + Example Code:

|  |
| --- |
| import nltk  from nltk.stem import WordNetLemmatizer  # Initialize the lemmatizer  lemmatizer = WordNetLemmatizer()  # Example sentence  sentence = "The cats were playing in the garden"  # Tokenize the sentence into individual words  tokens = nltk.word\_tokenize(sentence)  # Lemmatize each word  lemmatized\_words = [lemmatizer.lemmatize(word) for word in tokens]  # Print the lemmatized words  print(lemmatized\_words) |

* Stemming
  + Stemming just removes the suffixes
  + Example Code:

|  |
| --- |
| import nltk  from nltk.stem import PorterStemmer  # Initialize the stemmer  stemmer = PorterStemmer()  # Example sentence  sentence = "The cats were playing in the garden"  # Tokenize the sentence into individual words  tokens = nltk.word\_tokenize(sentence)  # Stem each word  stemmed\_words = [stemmer.stem(word) for word in tokens]  # Print the stemmed words  print(stemmed\_words) |

* Pos tagging
  + POS tagging is the process of finding the types of words in a grammatical way
  + Example Code:

|  |
| --- |
| import nltk  # Example sentence  sentence = "The cats were playing in the garden"  # Tokenize the sentence into individual words  tokens = nltk.word\_tokenize(sentence)  # Perform POS tagging  pos\_tags = nltk.pos\_tag(tokens)  # Print the POS tags  print(pos\_tags) |

* Sentiment analysis
  + Sentiment analysis is for finding the intent
  + Example Code:

|  |
| --- |
| import nltk  from nltk.sentiment import SentimentIntensityAnalyzer  # Initialize the sentiment analyzer  sia = SentimentIntensityAnalyzer()  # Example sentence  sentence = "I loved the movie! It was fantastic."  # Perform sentiment analysis  sentiment\_scores = sia.polarity\_scores(sentence)  # Print the sentiment scores  print(sentiment\_scores) |

spaCy:

nlp() takes care of everything

Example Code:

|  |
| --- |
| import spacy  # Load the pre-trained model  nlp = spacy.load("en\_core\_web\_sm")  # Example sentence  sentence = "The cats were playing in the garden"  # Process the sentence  doc = nlp(sentence)  # Perform POS tagging  pos\_tags = [(token.text, token.pos\_) for token in doc]  # Perform lemmatization  lemmas = [token.lemma\_ for token in doc]  # Perform named entity recognition (NER)  ner\_entities = [(ent.text, ent.label\_) for ent in doc.ents]  # Perform sentiment analysis  sentiment\_score = doc.sentiment  # Print the results  print("POS Tags:")  print(pos\_tags)  print()  print("Lemmas:")  print(lemmas)  print()  print("Named Entities:")  print(ner\_entities)  print()  print("Sentiment Score:")  print(sentiment\_score) |

NLTK vs spaCy:

|  |  |
| --- | --- |
| Different methods for each operation | Nlp() can do the most |
| has sentiment analysis | Does not have sentiment analysis |
| Not object oriented | Object oriented |