1. What are Corpora?

In the context of Natural Language Processing (NLP), corpora refer to large collections of text or speech data that are specifically gathered and curated for language processing tasks. NLP corpora serve as essential resources for training and evaluating various NLP models and algorithms.

1. What are Tokens?

In Natural Language Processing (NLP), tokens refer to the individual units or elements into which a text is divided for processing. These tokens can be words, subwords, characters, or even larger units, depending on the specific tokenization approach used. Tokenization is the process of splitting a text into tokens.

Here are a few commonly used types of tokens in NLP:

1. Word tokens: In many NLP tasks, the most common form of tokenization is splitting the text into individual words. Each word in the text is considered a separate token. For example, the sentence "I love NLP!" would be tokenized into the following word tokens:

["I", "love", "NLP", "!"].

2.Subword tokens: Subword tokenization splits the text into smaller units that are not necessarily whole words. This approach is often used in languages with complex morphology or in scenarios with limited training data. Subword tokens can capture meaningful subunits of words, such as prefixes, suffixes, or stems. For example, the word "unhappiness" might be tokenized into ["un", "happiness"].

3.Character tokens: In some cases, individual characters within a text can be treated as tokens. This approach is useful when dealing with character-level models or tasks that require fine-grained analysis at the character level.

4. Byte-pair encoding (BPE) tokens: BPE is a popular subword tokenization technique that identifies common subword units based on the frequency of their occurrence in a corpus. BPE tokenization splits the text into variable-length subword tokens, allowing for effective representation of rare words or out-of-vocabulary (OOV) terms.

Tokenization is an important step in NLP preprocessing pipelines as it enables subsequent analysis and processing tasks. Tokenized representations of text are often used as input for various NLP models, such as language models, sequence-to-sequence models, or deep learning architectures. Tokens serve as the fundamental units for training and inference, allowing the models to process and understand the structure and meaning of the text.

1. What are Unigrams, Bigrams, Trigrams?

Unigrams, bigrams, and trigrams are different types of n-grams used in Natural Language Processing (NLP) for analyzing the sequential structure of text. N-grams are contiguous sequences of n items, where an item can be a word, subword, or character, depending on the chosen tokenization level.

1.Unigrams: Unigrams are n-grams of size 1, representing individual items, typically words. In unigram analysis, each word in a text is treated as a separate unit, disregarding any information about the surrounding context. For example, in the sentence "I love NLP," the unigrams would be ["I", "love", "NLP"].

2.Bigrams: Bigrams are n-grams of size 2, representing adjacent pairs of items, usually words. Bigrams capture the sequential relationship between two consecutive words in a text. For example, given the sentence "I love NLP," the bigrams would be ["I love", "love NLP"].

3.Trigrams: Trigrams are n-grams of size 3, representing triplets of consecutive items, typically words. Trigrams provide a more context-rich view of the text compared to unigrams and bigrams. Continuing with the previous example, the trigrams for the sentence "I love NLP" would be ["I love NLP"].

N-grams are useful in several NLP tasks, including language modeling, information retrieval, and machine translation. By examining the frequencies and distributions of n-grams in a corpus, we can gain insights into the language patterns and relationships between words or subword units. N-grams can help in tasks such as predicting the next word in a sequence, identifying collocations or phrases, detecting language errors, or even generating text.

It's important to note that the choice of n in n-grams depends on the specific task and the desired level of context considered. Unigrams provide basic word-level information, bigrams capture local dependencies between adjacent words, and trigrams offer a slightly broader context. However, higher n-grams, such as 4-grams or 5-grams, can also be used to capture longer-range dependencies, although they come with increased sparsity and data requirements.

1. How to generate n-grams from text?

Generating n-grams from text involves splitting the text into contiguous sequences of n items. These items can be characters, words, or any other unit of text, depending on the desired level of granularity. Here's a general approach to generating n-grams from text:

1. Preprocess the text: Remove any unnecessary characters or symbols, and convert the text to lowercase if needed.
2. Tokenize the text: Split the text into individual units, such as words or characters. The choice of tokenization depends on the desired level of granularity. For example, you can use whitespace-based tokenization to split text into words, or simply treat each character as a token.
3. Generate n-grams: Iterate over the tokens and create n-grams by selecting consecutive sequences of n tokens. You can use a sliding window approach to extract n-grams.

* If you want to generate character n-grams, set n to the desired number of characters and slide the window over the text.
* If you want to generate word n-grams, set n to the desired number of words and slide the window over the tokenized words.

1. Store the n-grams: As you generate each n-gram, store it in a suitable data structure, such as a list, array, or dictionary, depending on your specific needs.
2. Explain Lemmatization

The method of mapping all the various forms of a word to its base word (also called “lemma”) is known as Lemmatization. Although this may appear close to the definition of stemming, these are actually different. For instance, the word “better,” after stemming, remains the same. However, upon lemmatization, this should become “good,”. Lemmatization needs greater linguistic knowledge. Modelling and developing efficient lemmatizers still remains an open problem in NLP research.

1. Explain Stemming

When we remove the suffixes from a word so that the word is reduced to its base form, this process is called stemming. When the word is reduced to its base form, all the different variants of that word can be represented by the same form (e.g., “bird” and “birds” are both reduced to “bird”).

We can do this by using a fixed set of rules. For instance:  if a word ends in “-es,” we can remove the “-es”).

Even though these rules might not really make sense as a linguistically correct base form, stemming is usually carried out to match user queries in search engines to relevant documents. And in text classification, is done to reduce the feature space to train our machine learning (ML) models.

This gives “bike” as the stemmed version for “bikes,” but “revolut” as the stemmed form of “revolution,” even though the latter is not linguistically correct. Even if this might not affect the performance of the search engine, a derivation of the correct linguistic form becomes useful in some other cases. This can be done by another process that is closer to stemming, known as lemmatization.

1. Explain Part-of-speech (POS) tagging

A Part-Of-Speech Tagger (POS Tagger) reads the text in a language and assigns parts of speech to each word (and other tokens), such as noun, verb, adjective, and so on. To label terms in text bodies, PoS taggers employ an algorithm. With tags like "nounplural" or even more complicated labels, these taggers create more complex categories than those stated as basic PoS.

1. Explain Chunking or shallow parsing

Chunking, also known as shallow parsing, is a natural language processing (NLP) technique that involves grouping and segmenting words into meaningful syntactic units called "chunks." It aims to identify and extract phrases or noun phrases from sentences based on their linguistic patterns and grammatical structures.

In chunking, the focus is on identifying the structure of a sentence without assigning detailed syntactic labels to each word. It provides a higher-level representation of the text compared to full syntactic parsing. Chunking typically operates on parts of speech (POS) tagged sentences, where each word in a sentence is assigned a specific POS tag (e.g., noun, verb, adjective) indicating its grammatical role.

The most common type of chunking is called noun phrase (NP) chunking, which involves identifying and extracting noun phrases from a sentence. A noun phrase is a group of words centered around a noun that can include determiners, adjectives, and other modifiers. For example, in the sentence "The big cat is sleeping," the noun phrase "The big cat" can be chunked.

Chunking follows certain linguistic patterns or rules to identify and extract chunks. These patterns can be defined using regular expressions or rule-based approaches. Common patterns for NP chunking include:

* An optional determiner (DT)
* Followed by zero or more adjectives (JJ)
* Followed by one or more nouns (NN)

Using these patterns, a chunker can recognize and extract noun phrases from sentences. For example, given the sentence "The black dog is running," the NP chunker can identify and extract the chunk "The black dog."

Chunking is often used as a pre-processing step in various NLP tasks, such as information extraction, named entity recognition, and text classification. By extracting meaningful chunks from a sentence, it helps in reducing the complexity of the text while retaining important syntactic and semantic information.

1. Explain Noun Phrase (NP) chunking

Noun Phrase (NP) chunking, also known as noun phrase extraction or noun phrase parsing, is a natural language processing (NLP) technique that focuses on identifying and extracting noun phrases from sentences. A noun phrase is a group of words centered around a noun that can include determiners, adjectives, and other modifiers.

NP chunking involves analyzing the grammatical structure of a sentence to identify the boundaries of noun phrases. It is typically performed as a part of shallow parsing or chunking, which aims to identify meaningful syntactic units without providing detailed syntactic labels for each word.

To perform NP chunking, the text is usually preprocessed by assigning part-of-speech (POS) tags to each word in the sentence. POS tagging involves labeling words with their grammatical categories, such as noun (NN), verb (VB), adjective (JJ), and so on. This POS-tagged sentence is then processed to identify noun phrases based on specific patterns or rules.

Common patterns used in NP chunking include:

1.Optional determiner (DT): A noun phrase can start with an optional determiner like "a," "an," "the," or "some."

2. Zero or more adjectives (JJ): A noun phrase can have zero or more adjectives that modify the noun.

3.One or more nouns (NN): A noun phrase must contain at least one noun as its core component.

Using these patterns, NP chunking algorithms scan through the POS-tagged sentence and identify sequences of words that match the defined patterns. These identified sequences are then labeled as noun phrases or chunks.

1. Explain Named Entity Recognition

Named Entity Recognition (NER) is a natural language processing (NLP) technique that focuses on identifying and classifying named entities in text. Named entities are real-world objects such as persons, organizations, locations, dates, quantities, and more, that have specific names or labels.

The goal of NER is to extract and categorize these named entities from text in order to understand the relevant information and relationships within the text. NER is a crucial component in various NLP applications, including information extraction, question answering, document classification, and sentiment analysis.

NER involves analyzing the grammatical and contextual features of text to recognize and classify named entities. The process typically involves the following steps:

1.Tokenization: The input text is divided into individual words or tokens.

2.Part-of-Speech (POS) Tagging: Each token is labeled with its corresponding part of speech (e.g., noun, verb, adjective).

3. Named Entity Recognition: The labeled tokens are analyzed to identify and classify named entities.

NER can be performed using different approaches, including rule-based methods, statistical models, and machine learning algorithms.

Rule-based methods rely on handcrafted patterns and heuristics to identify named entities. These rules are designed based on linguistic patterns, syntactic structures, and contextual cues. For example, a rule may state that a noun phrase following the word "President" is likely to be a person's name.

Statistical models and machine learning algorithms learn patterns and associations from annotated training data. They use features such as word context, part-of-speech tags, and syntactic structures to make predictions about named entity labels. Popular machine learning algorithms for NER include Conditional Random Fields (CRF), Hidden Markov Models (HMM), and deep learning-based models like Recurrent Neural Networks (RNN) and Transformer models.

The output of NER is typically a labeled sequence of tokens where each named entity is tagged with its corresponding label. Common labels include:

* PERSON: Referring to a person's name.
* ORGANIZATION: Referring to a company, institution, or organization name.
* LOCATION: Referring to a place or location name.
* DATE: Referring to a specific date or time.
* MONEY: Referring to monetary values.
* PERCENT: Referring to percentages.
* and others, depending on the specific application.