**What is computational linguistics (CL)?**

Computational linguistics (CL) is the application of computer science to the analysis and comprehension of written and spoken language. As an interdisciplinary field, CL combines linguistics with computer science and artificial intelligence ([AI](https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence)) and is concerned with understanding language from a computational perspective. Computers that are linguistically competent help facilitate human interaction with machines and software.

Computational linguistics is used in tools such as instant machine translation, [speech recognition](https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition) systems, [parsers](https://www.techtarget.com/searchapparchitecture/definition/parser), text-to-speech synthesizers, [interactive voice response](https://www.techtarget.com/searchcustomerexperience/definition/Interactive-Voice-Response-IVR) systems, search engines, text editors and language instruction materials.

The term *computational linguistics* is also closely linked to natural language processing ([NLP](https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP)), and these two terms are often used interchangeably.

**Applications of computational linguistics**

Most work in computational linguistics -- which has both theoretical and applied elements -- is aimed at improving the relationship between computers and basic language. It involves building [artifacts](https://www.techtarget.com/searchsoftwarequality/definition/artifact-software-development) that can be used to process and produce language. Building such artifacts requires [data scientists](https://www.techtarget.com/searchenterpriseai/definition/data-scientist) to analyze massive amounts of written and spoken language in both [structured](https://www.techtarget.com/whatis/definition/structured-data) and [unstructured](https://www.techtarget.com/searchbusinessanalytics/definition/unstructured-data) formats.

Applications of CL typically include the following:

* **Machine translation.** This is the process of [using AI to translate](https://www.techtarget.com/searchenterpriseai/feature/Where-are-we-with-machine-translation-in-AI) one human language to another.
* **Application clustering.**This is the process of turning multiple computer [servers](https://www.techtarget.com/whatis/definition/server) into a cluster.
* **Sentiment analysis.** [Sentiment analysis](https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining) is an important approach to NLP that identifies the emotional tone behind a body of text.
* **Chatbots.** These software or computer programs simulate human conversation or *chatter* through text or voice interactions.
* **Information extraction.** This is the creation of knowledge from structured and unstructured text.
* **Natural language interfaces.** These are computer-human interfaces where words, phrases or clauses act as user interface controls.
* **Content filtering.** This process blocks various language-based web content from reaching users.
* **Text mining.** [Text mining](https://www.techtarget.com/searchbusinessanalytics/definition/text-mining) is the process of extracting useful information from massive amounts of unstructured textual data. [Tokenization](https://www.techtarget.com/searchsecurity/definition/tokenization), part-of-speech tagging -- named entity recognition and sentiment analysis -- are used to accomplish this process.

**Approaches and methods of computational linguistics**

There have been many different approaches and methods of computational linguistics since its beginning in the 1950s. Examples of some CL approaches include the following:

* The **corpus-based**approach, which is based on the language as it's practically used.
* The **comprehension**approach, which enables the NLP engine to interpret naturally written commands in a simple rule-governed environment.
* The **developmental**approach, which adopts the language acquisition strategy of a child by acquiring language over time. The developmental process has a statistical approach to studying language and doesn't take grammatical structure into account.
* The **structural**approach, which takes a theoretical approach to the structure of a language. This approach uses large samples of a language run through computational models to gain a better understanding of the underlying language structures.
* The **production**approach focuses on a CL model to produce text. This has been done in a number of ways, including the construction of [algorithms](https://www.techtarget.com/whatis/definition/algorithm) that produce text based on example texts from humans. This approach can be broken down into the following two approaches:
  + The **text-based interactive**approach uses text from a human to generate a response by an algorithm. A computer can recognize different patterns and reply based on user input and specified keywords.
  + The **speech-based interactive**approach works similarly to the text-based approach, but user input is made through speech recognition. The user's speech input is recognized as sound waves and is interpreted as patterns by the CL system.

## CL vs. NLP

Computational linguistics and natural language processing are similar concepts, as both fields require formal training in computer science, linguistics and machine learning ([ML](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML)). Both use the same tools, such as ML and AI, to accomplish their goals and many NLP tasks need an understanding or interpretation of language.

NLP plays an important role in [creating language technologies](https://www.techtarget.com/searchbusinessanalytics/tip/Natural-language-processing-augments-analytics-and-data-use), including chatbots, speech recognition systems and [virtual assistants](https://www.techtarget.com/searchcustomerexperience/definition/virtual-assistant-AI-assistant), such as [Siri](https://www.techtarget.com/searchmobilecomputing/definition/Siri), Alexa and [Cortana](https://www.techtarget.com/searchenterprisedesktop/definition/Cortana). Meanwhile, CL lends its expertise to topics such as preserving languages, analyzing historical documents and building dialogue systems, such as [Google Translate](https://www.techtarget.com/searchenterpriseai/tip/What-do-large-language-models-do-in-AI).

# **Levels/ Stages of Natural Language Processing**



The process of Natural Language Processing is divided into 5 major stages or phases, starting from basic word-level processing up to finding complex meanings of sentences.

1. **Morphological Analysis/ Lexical Analysis**

Morphological or Lexical Analysis deals with text at the individual word level. It looks for *morphemes*, the smallest unit of a word. For example, *irrationally* can be broken into *ir* (prefix), *rational* (root) and *-ly* (suffix). Lexical Analysis finds the relation between these morphemes and converts the word into its root form. A lexical analyzer also assigns the possible Part-Of-Speech (POS) to the word. It takes into consideration the dictionary of the language.

For example, the word “character” can be used as a noun or a verb.

#### **Syntax Analysis**

Syntax Analysis ensures that a given piece of text is correct structure. It tries to parse the sentence to check correct grammar at the sentence level. Given the possible POS generated from the previous step, a syntax analyzer assigns POS tags based on the sentence structure.

For example:

**Correct Syntax:** Sun rises in the east.

**Incorrect Syntax:** Rise in sun the east.

#### **3.Semantic Analysis**

Consider the sentence: “The apple ate a banana”. Although the sentence is syntactically correct, it doesn’t make sense because apples can’t eat. Semantic analysis looks for meaning in the given sentence. It also deals with combining words into phrases.

For example, “red apple” provides information regarding one object; hence we treat it as a single phrase. Similarly, we can group names referring to the same category, person, object or organisation. “Robert Hill” refers to the same person and not two separate names – “Robert” and “Hill”

#### **Discourse**

Discourse deals with the effect of a previous sentence on the sentence in consideration. In the text, “Albert is a bright student. He spends most of the time in the library.” Here, discourse assigns “he” to refer to “Albert”.

#### **Pragmatics**

The final stage of NLP, Pragmatics interprets the given text using information from the previous steps. Given a sentence, “Turn off the lights” is an order or request to switch off the lights.

# Tokenization in Natural Language Processing

To let machines understand the natural language, we first need to divide the input text in smaller chunks. Breaking paragraphs into sentences and then into individual words can help machines interpret meanings easily. This is where the concept of tokenization comes in Natural Language Processing.

### **Tokenization**

Tokenization is one of the most common tasks in text processing. It is the process of separating a given text into smaller units called tokens.

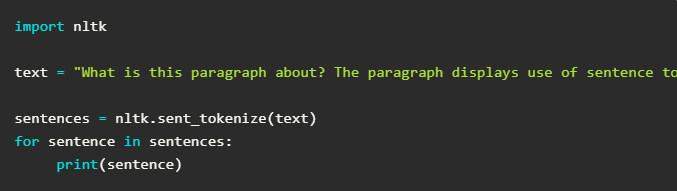
An input text is a group of multiple words which make a sentence. We need to break the text in such a way that machines can understand this text and tokenization helps us to achieve that.

It can be classified into 2 types:

#### **Sentence Tokenization**

Sentence tokenization is the process of dividing the text into its component sentence. The method is very simple. In layman’s term: split the sentences wherever there is an end-of-sentence punctuation mark. For example, the English language has 3 punctuations that indicate the end of a sentence: !, . and ?. Similarly, other languages have different closing punctuations.

While we can manually break sentences on these punctuations, python’s NLTK library provides us with the necessary tools and we need not worry about splitting sentences.



#### **Word Tokenization**

Word tokenization is the process of dividing a text into its component word. We need to split the text after every space is seen. Also, we need to take care of punctuation marks. It is easier to deal with individual words than to deal with a sentence. Thus, we need to further tokenize sentences into words.

A screen shot of a computer code

Description automatically generated

## ****Stopwords****

Stop words refers to common words in a language. These are words that do not contain major information but are necessary for making the sentence complete. Some examples of stop words are “in”, “the”, “is”, “an”, etc. We can safely ignore these words without losing the meaning of the content.



## ****Stemming****

Stemming refers to the crude chopping of words to reduce into their stem words. A Stemmer follows a set of pre-defined rules to remove affixes from inflectional words. For example: connects, connected, connection can be converted to connect.

#### **Porter’s Stemmer**

There are multiple stemming algorithms to chose from, Porter’s Stemmer being one of the most used. NLTK provides this algorithm as Porter Stemmer. To use this stemmer, we need to download it through Python Shell



## ****Lemmatization****

Lemmatization is similar to Stemming, however, a Lemmatizer always returns a valid word. Stemming uses rules to cut the word, whereas a Lemmatizer searched for the root word, also called as Lemma from the WordNet. Moreover, lemmatization takes care of converting a word into its base form; i.e. words like am, is, are will be converted to “be”.

#### **WordNetLemmatizer**

Again, NLTK provides a WordNetLemmatizer to use off-the-shelf. However, this requires the POS tags of the word for correct results. For now, we manually provide the POS tags.

A screen shot of a computer program

Description automatically generated

## ****Stemming**** ****vs Lemmatization****

Now that we know what Stemming and Lemmatization are, one may ask why to use Stemming at all if Lemmatization provides correct results?

A Stemmer is very fast in comparison to Lemmatization. Moreover, Lemmatization requires POS tags to perform correctly. In our example, we manually provided the POS tags. Although when dealing in an application, we need to perform this POS tagging. Then, each word is searched for its base form from the WordNet. This increases the computation time and may not be optimal.

In some cases, it might be better to use a Stemmer than to wait for Lemmatization. Whereas, if precision is important in an application, one can use Lemmatization over Stemming.

# **Part Of Speech Tagging – POS Tagging in NLP**



As discussed in [Stages of Natural Language Processing](https://byteiota.com/stages-of-nlp/), Syntax Analysis deals with the arrangement of words to form a structure that makes grammatical sense. A sentence is syntactically correct when the Parts of Speech of the sentence follow the rules of grammar. To achieve this, the given sentence structure is compared with the common language rules.

## ****Part of Speech****

Part of Speech is the classification of words based on their role in the sentence. The major POS tags are Nouns, Verbs, Adjectives, Adverbs. This category provides more details about the word and its meaning in the context. A sentence consists of words with a sensible Part of Speech structure.

For example: Book the flight!

This sentence contains Noun (Book), Determinant (the) and a Verb (flight).

**Part Of Speech Tagging**

POS tagging refers to the automatic assignment of a tag to words in a given sentence. It converts a sentence into a list of words with their tags. (word, tag). Since this task involves considering the sentence structure, it cannot be done at the Lexical level. A POS tagger considers surrounding words while assigning a tag.

For example, the previous sentence, *“Book the flight”*, will become a list of each word with its corresponding POS tag – [(“Book”, “Verb”), (“the”, “Det”), (“flight”, “Noun”)].

Similarly, *“I like to read book”* is represented as: [(“I”, “Preposition”), (“like”, “Verb”), (“to”, “To”), (“read”, “Verb”), (“books”, “Noun”)]. Notice how the word *Book*appears in both sentences. However, in the first example, it acts as a *Verb* but takes the role of a *Noun*in the latter.

Although we are using the generic names of the tags, in real practice, we refer a tagset for tags. The Penn TreeBank Tag Set is most used for the English language. Some examples from Penn Treebank:

|  |  |
| --- | --- |
| **Part Of Speech** | **Tag** |
| Noun (Singular) | NN |
| Noun (Plural) | NNS |
| Verb | VB |
| Determiner | DT |

Examples of Penn Treebank Tags

**Difficulties in POS Tagging**

Similar to most NLP problems, POS tagging suffers from ambiguity. In the sentences, “Book the flight” and “I like to read books”, we see that book can act as a Verb or Noun. Similarly, many words in the English dictionary has multiple possible POS tags.

* *This* *(Preposition)* is a car
* *This* *(Determiner)* car is red
* You can go *this* *(Adverb)* far only.

These sentences use the word “This” in various contexts. However, how can one assign the correct tag to the words?

## ****POS Tagging Approaches****

#### **Rule-Based POS Tag**

This is one of the oldest approaches to POS tagging. It involves using a dictionary consisting of all the possible POS tags for a given word. If any of the words have more than one tag, hand-written rules are used to assign the correct tag based on the tags of surrounding words.

For example, if the preceding of a word an article, then the word has to be a noun.

Consider the words: A Book

* + **Get all the possible POS tags for individual words:** A – Article; Book – Noun or Verb
  + **Use the rules to assign the correct POS tag:** As per the possible tags, “A” is an *Article* and we can assign it directly. But, a book can either be a Noun or a Verb. However, if we consider “A Book”, A is an article and following our rule above, Book has to be a *Noun*. Thus, we assign the tag of Noun to book.

**POS Tag:** [(“A”, “Article”), (“Book”, “Noun”)]

Similarly, various rules are written or machine-learned for other cases. Using these rules, it is possible to build a Rule-based POS tagger.

#### **Stochastic Tagger**

A Stochastic Tagger, a supervised model, involves using with frequencies or probabilities of the tags in the given training corpus to assign a tag to a new word. These taggers entirely rely on statistics of the tag occurrence, i.e. probability of the tags.

Based on the words used for determining a tag, Stochastic Taggers are divided into 2 parts:

* + **Word Frequency:**In this approach, we find the tag that is most assigned to the word. For example: Given a training corpus, “book” occurs 10 times – 6 times as Noun, 4 times as a Verb; the word book will always be assigned as “Noun” since it occurs the most in the training set. Hence, a Word Frequency Approach is not very reliable.
  + **Tag Sequence Frequency:** Here, the best tag for a word is determined using the probability the tags of N previous words, i.e. it considers the tags for the words preceding book. Although this approach provides better results than a Word Frequency Approach, it may still not provide accurate results for rare structures. Tag Sequence Frequency is also referred to as the N-gram approach.