Unit-2

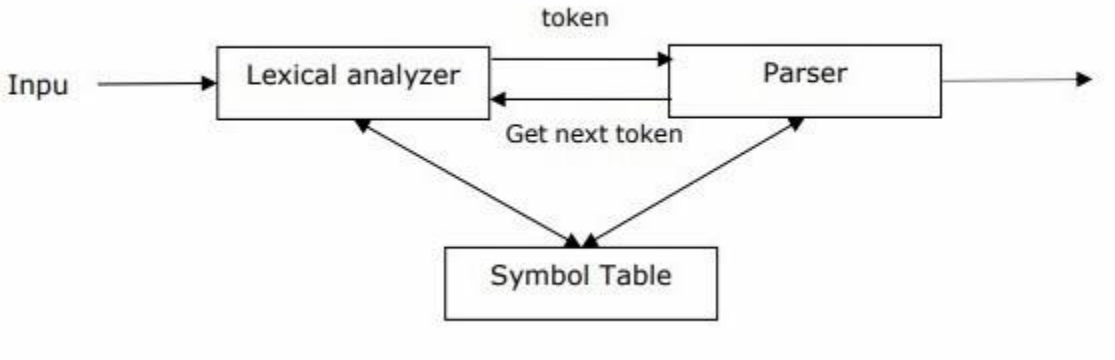
1. Syntax Parsing

Syntactic analysis or parsing or syntax analysis is the third phase of NLP. The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text. Syntax analysis checks the text for meaningfulness comparing to the rules of formal grammar. For example, the sentence like "hot ice-cream" would be rejected by semantic analyzer.

In this sense, syntactic analysis or parsing may be defined as the process of analyzing the strings of symbols in natural language conforming to the rules of formal grammar. The origin of the word ***'parsing'*** is from Latin word ***'pars'*** which means ***'part'***.

Concept of Parser

It is used to implement the task of parsing. It may be defined as the software component designed for taking input data (text) and giving structural representation of the input after checking for correct syntax as per formal grammar. It also builds a data structure generally in the form of parse tree or abstract syntax tree or other hierarchical structure.



The main roles of the parse include −

- To report any syntax error.
- To recover from commonly occurring error so that the processing of the remainder of program can be continued.
- To create parse tree.
- To create symbol table.
- To produce intermediate representations (IR).

Types of Parsing

Derivation divides parsing into the followings two types −

- Top-down Parsing
- Bottom-up Parsing

Top-down Parsing

In this kind of parsing, the parser starts constructing the parse tree from the start symbol and then tries to transform the start symbol to the input. The most common form of top-down parsing uses recursive procedure to process the input. The main disadvantage of recursive descent parsing is backtracking.

Bottom-up Parsing

In this kind of parsing, the parser starts with the input symbol and tries to construct the parser tree up to the start symbol.

Concept of Derivation

In order to get the input string, we need a sequence of production rules. Derivation is a set of production rules. During parsing, we need to decide the non-terminal, which is to be replaced along with deciding the production rule with the help of which the non-terminal will be replaced.

Types of Derivation

Left-most Derivation

In the left-most derivation, the sentential form of an input is scanned and replaced from the left to the right. The sentential form in this case is called the left-sentential form.

Right-most Derivation

In the left-most derivation, the sentential form of an input is scanned and replaced from right to left. The sentential form in this case is called the right-sentential form.

Concept of Parse Tree

It may be defined as the graphical depiction of a derivation. The start symbol of derivation serves as the root of the parse tree. In every parse tree, the leaf nodes are terminals and interior nodes are non-terminals. A property of parse tree is that in-order traversal will produce the original input string.

Concept of Grammar

Grammar is very essential and important to describe the syntactic structure of well-formed programs. In the literary sense, they denote syntactical rules for conversation in natural languages. Linguistics have attempted to define grammars since the inception of natural languages like English, Hindi, etc.

The theory of formal languages is also applicable in the fields of Computer Science mainly in programming languages and data structure. For example, in 'C' language, the precise grammar rules state how functions are made from lists and statements.

A mathematical model of grammar was given by **Noam Chomsky** in 1956, which is effective for writing computer languages.

Mathematically, a grammar G can be formally written as a 4-tuple (N, T, S, P) where –

- **N** or $\mathbf{V_N}$ = set of non-terminal symbols, i.e., variables.
- **T** or $\sum$ = set of terminal symbols.
- **S** = Start symbol where S N
- **P** denotes the Production rules for Terminals as well as Non-terminals. It has the form $\alpha \rightarrow \beta$, where α and β are strings on $V_N \sum$ and least one symbol of α belongs to V_N

Phrase Structure or Constituency Grammar

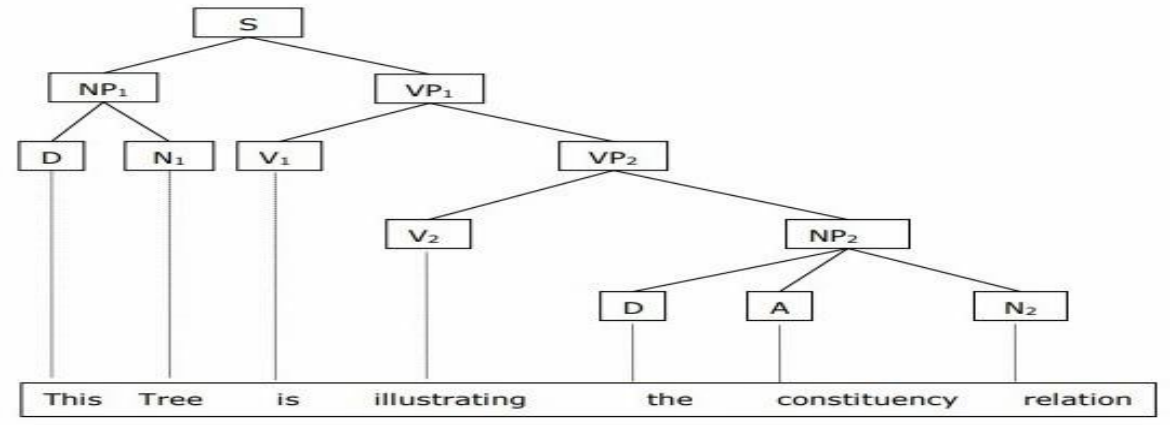
Phrase structure grammar, introduced by Noam Chomsky, is based on the constituency relation. That is why it is also called constituency grammar. It is opposite to dependency grammar.

Example

Before giving an example of constituency grammar, we need to know the fundamental points about constituency grammar and constituency relation.

- All the related frameworks view the sentence structure in terms of constituency relation.
- The constituency relation is derived from the subject-predicate division of Latin as well as Greek grammar.
- The basic clause structure is understood in terms of **noun phrase NP** and **verb phrase VP**.

We can write the sentence **"This tree is illustrating the constituency relation"** as follows –



Dependency Grammar

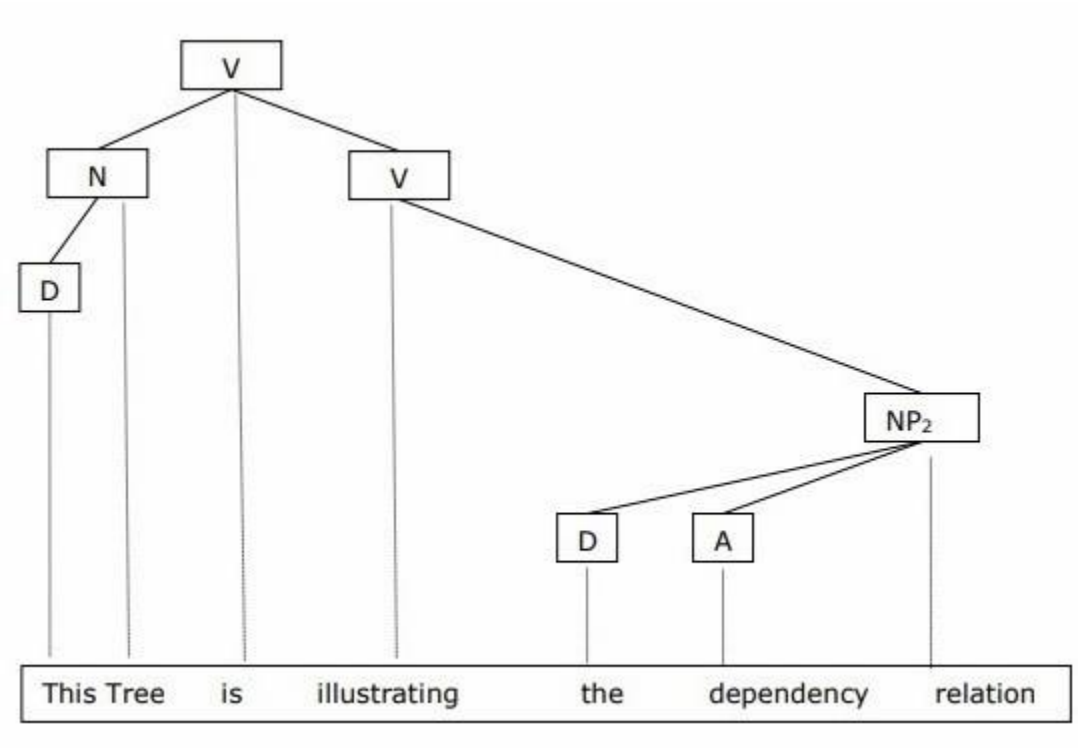
It is opposite to the constituency grammar and based on dependency relation. It was introduced by Lucien Tesniere. Dependency grammar (DG) is opposite to the constituency grammar because it lacks phrasal nodes.

Example

Before giving an example of Dependency grammar, we need to know the fundamental points about Dependency grammar and Dependency relation.

- In DG, the linguistic units, i.e., words are connected to each other by directed links.
- The verb becomes the center of the clause structure.
- Every other syntactic unit is connected to the verb in terms of directed link. These syntactic units are called ***dependencies***.

We can write the sentence **"This tree is illustrating the dependency relation"** as follows;

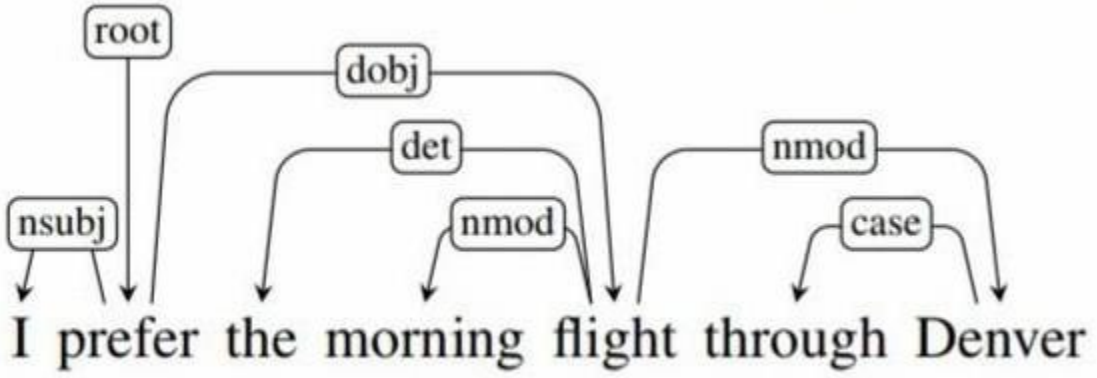


Parse tree that uses Constituency grammar is called constituency-based parse tree; and the parse trees that uses dependency grammar is called dependency-based parse tree.

2. **Dependency Parsing**

   Dependency Parsing (DP) refers to examining the dependencies between the words of a sentence to analyze its grammatical structure. Based on this, a sentence is broken into several components. The mechanism is based on the concept that there is a direct link between every linguistic unit of a sentence. These links are termed dependencies.

Let's take for example the sentence "I prefer the morning flight through Denver." The diagram below explains the dependency structure of the sentence:



Dependency Parsing using NLTK

Dependency Parsing can be carried out using the Natural Language Toolkit (NLTK) package which is a collection of libraries and codes used in the statistical Natural Language Processing (NLP) of human language.

We can use NLTK to achieve dependency parsing through one of the following methods:

1. **Probabilistic, projective dependency parser**: These parsers use the knowledge of human language gleaned from hand-parsed sentences to predict new sentences. They are known to make mistakes and work with a restricted set of training data.
2. **Stanford parser**: This is a natural language parser implemented on Java. You need the Stanford Core NLP parser to perform dependency parsing. The parser includes several languages including English, Chinese, German, and Arabic.

   Here's how you can use the parser:

```
from nltk.parse.stanford import StanfordDependencyParser
path_jar = 'path_to/stanford-parser-full-2014-08-27/stanford-parser.jar'
path_models_jar = 'path_to/stanford-parser-full-2014-08-27/stanford-parser-3.4.1-models.jar'
dep_parser = StanfordDependencyParser(
  path_to_jar = path_jar, path_to_models_jar = path_models_jar
)
result = dep_parser.raw_parse('I shot an elephant in my sleep')
dependency = result.next()
list(dependency.triples())
```

   The output of the above program is as follows:

```
[
  ((u'shot', u'VBD'), u'nsubj', (u'I', u'PRP')),
 ((u'shot', u'VBD'), u'dobj', (u'elephant', u'NN')),
```

```
  ((u'elephant', u'NN'), u'det', (u'an', u'DT')),
  ((u'shot', u'VBD'), u'prep', (u'in', u'IN')),
  ((u'in', u'IN'), u'pobj', (u'sleep', u'NN')),
  ((u'sleep', u'NN'), u'poss', (u'my', u'PRP$'))
]
```

3. **Semantic Parsing**
   The purpose of semantic analysis is to draw exact meaning, or you can say dictionary meaning from the text. The work of semantic analyzer is to check the text for meaningfulness.

Studying meaning of individual word

It is the first part of the semantic analysis in which the study of the meaning of individual words is performed. This part is called lexical semantics.

Studying the combination of individual words

In the second part, the individual words will be combined to provide meaning in sentences.

The most important task of semantic analysis is to get the proper meaning of the sentence. For example, analyze the sentence **"Ram is great."** In this sentence, the speaker is talking either about Lord Ram or about a person whose name is Ram. That is why the job, to get the proper meaning of the sentence, of semantic analyzer is important.

Elements of Semantic Analysis

Followings are some important elements of semantic analysis –

Hyponymy

It may be defined as the relationship between a generic term and instances of that generic term. Here the generic term is called hypernym and its instances are called hyponyms. For example, the word color is hypernym and the color blue, yellow etc. are hyponyms.

Homonymy

It may be defined as the words having same spelling or same form but having different and unrelated meaning. For example, the word "Bat" is a homonymy word because bat can be an implement to hit a ball or bat is a nocturnal flying mammal also.

Polysemy

Polysemy is a Greek word, which means "many signs". It is a word or phrase with different but related sense. In other words, we can say that polysemy has the same spelling but different and related meaning. For example, the word "bank" is a polysemy word having the following meanings –

- A financial institution.

- The building in which such an institution is located.
- A synonym for "to rely on".

Difference between Polysemy and Homonymy

Both polysemy and homonymy words have the same syntax or spelling. The main difference between them is that in polysemy, the meanings of the words are related but in homonymy, the meanings of the words are not related. For example, if we talk about the same word "Bank", we can write the meaning 'a financial institution' or 'a river bank'. In that case it would be the example of homonym because the meanings are unrelated to each other.

Synonymy

It is the relation between two lexical items having different forms but expressing the same or a close meaning. Examples are 'author/writer', 'fate/destiny'.

Antonymy

It is the relation between two lexical items having symmetry between their semantic components relative to an axis. The scope of antonymy is as follows –

- **Application of property or not** – Example is 'life/death', 'certitude/incertitude'
- **Application of scalable property** – Example is 'rich/poor', 'hot/cold'
- **Application of a usage** – Example is 'father/son', 'moon/sun'.

Meaning Representation

Semantic analysis creates a representation of the meaning of a sentence. But before getting into the concept and approaches related to meaning representation, we need to understand the building blocks of semantic system.

Building Blocks of Semantic System

In word representation or representation of the meaning of the words, the following building blocks play an important role –

- **Entities** – It represents the individual such as a particular person, location etc. For example, Haryana. India, Ram all are entities.
- **Concepts** – It represents the general category of the individuals such as a person, city, etc.
- **Relations** – It represents the relationship between entities and concept. For example, Ram is a person.
- **Predicates** – It represents the verb structures. For example, semantic roles and case grammar are the examples of predicates.

Approaches to Meaning Representations

Semantic analysis uses the following approaches for the representation of meaning –

- First order predicate logic (FOPL)
- Semantic Nets

- Frames
- Conceptual dependency (CD)
- Rule-based architecture
- Case Grammar
- Conceptual Graphs

Need of Meaning Representations

A question that arises here is why do we need meaning representation? Followings are the reasons for the same –

Linking of linguistic elements to non-linguistic elements

The very first reason is that with the help of meaning representation the linking of linguistic elements to the non-linguistic elements can be done.

Representing variety at lexical level

With the help of meaning representation, unambiguous, canonical forms can be represented at the lexical level.

Can be used for reasoning

Meaning representation can be used to reason for verifying what is true in the world as well as to infer the knowledge from the semantic representation.

Lexical Semantics

The first part of semantic analysis, studying the meaning of individual words is called lexical semantics. It includes words, sub-words, affixes (sub-units), compound words and phrases also. All the words, sub-words, etc. are collectively called lexical items. In other words, we can say that lexical semantics is the relationship between lexical items, meaning of sentences and syntax of sentence.

Following are the steps involved in lexical semantics –

- Classification of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.
- Decomposition of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.
- Differences as well as similarities between various lexical semantic structures are also analyzed.

4. Word Sense Disambiguation

Word sense disambiguation, in natural language processing (NLP), may be defined as the ability to determine which meaning of word is activated by the use of word in a particular context. Lexical ambiguity, syntactic or semantic, is one of the very first problems that any NLP system faces. Part-of-speech (POS) taggers with high level of accuracy can solve Word's syntactic ambiguity. On the other hand, the problem of resolving semantic

ambiguity is called WSD (word sense disambiguation). Resolving semantic ambiguity is harder than resolving syntactic ambiguity.

For example, consider the two examples of the distinct sense that exist for the word ***"bass"*** –

- I can hear bass sound.
- He likes to eat grilled bass.

The occurrence of the word **bass** clearly denotes the distinct meaning. In first sentence, it means **frequency** and in second, it means **fish**. Hence, if it would be disambiguated by WSD then the correct meaning to the above sentences can be assigned as follows –

- I can hear bass/frequency sound.
- He likes to eat grilled bass/fish.

Evaluation of WSD

The evaluation of WSD requires the following two inputs –

A Dictionary

The very first input for evaluation of WSD is dictionary, which is used to specify the senses to be disambiguated.

Test Corpus

Another input required by WSD is the high-annotated test corpus that has the target or correct-senses. The test corpora can be of two types:

- **Lexical sample** – this kind of corpora is used in the system, where it is required to disambiguate a small sample of words.
- **All-words** – this kind of corpora is used in the system, where it is expected to disambiguate all the words in a piece of running text.

Approaches and Methods to Word Sense Disambiguation (WSD)

Approaches and methods to WSD are classified according to the source of knowledge used in word disambiguation.

Four conventional methods to WSD –

Dictionary-based or Knowledge-based Methods

As the name suggests, for disambiguation, these methods primarily rely on dictionaries, treasures and lexical knowledge base. They do not use corpora evidences for disambiguation. The Lesk method is the seminal dictionary-based method introduced by Michael Lesk in 1986. The Lesk definition, on which the Lesk algorithm is based, is **"measure overlap between sense definitions for all words in context"**. However, in 2000, Kilgarriff and Rosensweig gave the simplified Lesk definition as **"measure overlap between sense definitions of word and current context"**, which further means identify the correct sense for one word at a time. Here the current context is the set of words in surrounding sentence or paragraph.

Supervised Methods

For disambiguation, machine learning methods make use of sense-annotated corpora to train. These methods assume that the context can provide enough evidence on its own to disambiguate the sense. In these methods, the words knowledge and reasoning are deemed unnecessary. The context is represented as a set of "features" of the words. It includes the information about the surrounding words also. Support vector machine and memory-based learning are the most successful supervised learning approaches to WSD. These methods rely on substantial amount of manually sense-tagged corpora, which is very expensive to create.

Semi-supervised Methods

Due to the lack of training corpus, most of the word sense disambiguation algorithms use semi-supervised learning methods. It is because semi-supervised methods use both labelled as well as unlabeled data. These methods require very small amount of annotated text and large amount of plain unannotated text. The technique that is used by semi-supervised methods is bootstrapping from seed data.

Unsupervised Methods

These methods assume that similar senses occur in similar context. That is why the senses can be induced from text by clustering word occurrences by using some measure of similarity of the context. This task is called word sense induction or discrimination. Unsupervised methods have great potential to overcome the knowledge acquisition bottleneck due to non-dependency on manual efforts.

Applications of Word Sense Disambiguation (WSD)

Word sense disambiguation (WSD) is applied in almost every application of language technology.

Machine Translation

Machine translation or MT is the most obvious application of WSD. In MT, Lexical choice for the words that have distinct translations for different senses is done by WSD. The senses in MT are represented as words in the target language. Most of the machine translation systems do not use explicit WSD module.

Information Retrieval (IR)

Information retrieval (IR) may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information. The system basically assists users in finding the information they required but it does not explicitly return the answers of the questions. WSD is used to resolve the ambiguities of the queries provided to IR system. As like MT, current IR systems do not explicitly use WSD module and they rely on the concept that user would type enough context in the query to only retrieve relevant documents.

Text Mining and Information Extraction (IE)

In most of the applications, WSD is necessary to do accurate analysis of text. For example, WSD helps intelligent gathering system to do flagging of the correct words. For example, medical intelligent system might need flagging of "illegal drugs" rather than "medical drugs"

Lexicography

WSD and lexicography can work together in loop because modern lexicography is corpus based. With lexicography, WSD provides rough empirical sense groupings as well as statistically significant contextual indicators of sense.

Difficulties in Word Sense Disambiguation (WSD)

Followings are some difficulties faced by word sense disambiguation (WSD) –

Differences between dictionaries

The major problem of WSD is to decide the sense of the word because different senses can be very closely related. Even different dictionaries and thesauruses can provide different divisions of words into senses.

Different algorithms for different applications

Another problem of WSD is that completely different algorithm might be needed for different applications. For example, in machine translation, it takes the form of target word selection; and in information retrieval, a sense inventory is not required.

Inter-judge variance

Another problem of WSD is that WSD systems are generally tested by having their results on a task compared against the task of human beings. This is called the problem of inter judge variance.

Word-sense discreteness

Another difficulty in WSD is that words cannot be easily divided into discrete sub meanings.

5. **Structural Disambiguation**

Structural or syntactic ambiguity is the potential of multiple interpretations for a piece of written or spoken language because of the way words or phrases are organized. Linguistic ambiguity makes it difficult for a human or an AI system, such as a natural language processing (NLP) program; to determine meaning unless further information is available that clarifies the context.

Some structural ambiguity is the result of writing errors, such as misplaced modifiers. An example from Tom Sant's book *Persuasive Business Proposals*: "Featuring plug-in circuit boards, we can strongly endorse this server's flexibility and growth potential."

That sentence might be intended to mean that the server has plug-in circuit boards, and a human would be likely to understand that. However, the way it's organized, the sentence actually means that the writer features plug-in circuit boards, and software would be likely to require word sense disambiguation (WSD) to understand that is not the intended meaning.

The term *structural ambiguity* is often contrasted with lexical (word-related) ambiguity, which often arises because words can have multiple meanings. Both are examples of linguistic ambiguity, which also results from other things including figurative language and vagueness.

6. Context and Sentence-level Semantics

Context analysis in NLP involves breaking down sentences to extract the n-grams, noun phrases, themes, and facets present within.

The Foundations of Context Analysis

The foundation of context determination is the **noun**. Of course, this is true of named entity extraction as well. But while entity extraction deals with proper nouns, context analysis is based around more general nouns.

For example, where "Cessna" and "airplane" will be classified as entities, "transportation" will be considered a theme (more on themes later).

Lexalytics, an In Moment company, supports four methods of context analysis, each with its merits and disadvantages:

- N-grams
- Noun phrases
- Themes
- Facets

Using N-grams for Basic Context Analysis

N-grams are combinations of one or more words that represent entities, phrases, concepts, and themes that appear in text. N-grams form the basis of many text analytics functions, including other context analysis methods such as Theme Extraction. We'll discuss themes later, but first it's important to understand what an n-gram is and what it represents.

There are three common levels of n-gram:

- 1 word = mono-gram
- 2 words = bi-gram
- 3 words = tri-gram

To get an idea of the relative strengths and weaknesses of mono-grams, bi-grams, and tri-grams, let's analyze two phrases and a sentence:

- "crazy good"
- "stone cold crazy"
- "President Barack Obama did a great job with that awful oil spill."

	Mono-grams	Bi-grams	Tri-grams
Phrases Extracted (crazy good, stone cold crazy)	crazy (2) cold good	crazy good cold crazy stone cold	stone cold crazy
Phrases Extracted (President Obama)	a awful barack did great job obama oil president spill that with	a great awful oil barack obama did a great job job with obama did oil spill president barack that awful with that	a great job awful oil spill barack obama did did a great great job with job with that obama did a president barack obama that awful oil with that awful