1. Natural Language Processing — Introduction

Language is a method of communication with the help of which we can speak, read and write. For example, we think, we make decisions, plans and more in natural language; precisely, in words. However, the big question that confronts us in this AI era is that can we communicate in a similar manner with computers. In other words, can human beings communicate with computers in their natural language? It is a challenge for us to develop NLP applications because computers need structured data, but human speech is unstructured and often ambiguous in nature.

In this sense, we can say that Natural Language Processing (NLP) is the sub-field of Computer Science especially Artificial Intelligence (AI) that is concerned about enabling computers to understand and process human language. Technically, the main task of NLP would be to program computers for analyzing and processing huge amount of natural language data.

History of NLP

We have divided the history of NLP into four phases. The phases have distinctive concerns and styles.

First Phase (Machine Translation Phase) – Late 1940s to late 1960s

The work done in this phase focused mainly on machine translation (MT). This phase was a period of enthusiasm and optimism.

Let us now see all that the first phase had in it:

- The research on NLP started in early 1950s after Booth & Richens' investigation and Weaver's memorandum on machine translation in 1949.
- 1954 was the year when a limited experiment on automatic translation from Russian to English demonstrated in the Georgetown-IBM experiment.
- In the same year, the publication of the journal MT (Machine Translation) started.
- The first international conference on Machine Translation (MT) was held in 1952 and second was held in 1956.
- In 1961, the work presented in Teddington International Conference on Machine Translation of Languages and Applied Language analysis was the high point of this phase.

Second Phase (AI Influenced Phase) – Late 1960s to late 1970s

In this phase, the work done was majorly related to world knowledge and on its role in the construction and manipulation of meaning representations. That is why, this phase is also called AI-flavored phase.

The phase had in it, the following:

- In early 1961, the work began on the problems of addressing and constructing data or knowledge base. This work was influenced by AI.

- In the same year, a BASEBALL question-answering system was also developed. The input to this system was restricted and the language processing involved was a simple one.

- A much advanced system was described in Minsky (1968). This system, when compared to the BASEBALL question-answering system, was recognized and provided for the need of inference on the knowledge base in interpreting and responding to language input.

Third Phase (Grammatico-logical Phase) – Late 1970s to late 1980s

This phase can be described as the grammatico-logical phase. Due to the failure of practical system building in last phase, the researchers moved towards the use of logic for knowledge representation and reasoning in AI.

The third phase had the following in it:

- The grammatico-logical approach, towards the end of decade, helped us with powerful general-purpose sentence processors like SRI's Core Language Engine and Discourse Representation Theory, which offered a means of tackling more extended discourse.

- In this phase we got some practical resources & tools like parsers, e.g. Alvey Natural Language Tools along with more operational and commercial systems, e.g. for database query.

- The work on lexicon in 1980s also pointed in the direction of grammatico-logical approach.

Fourth Phase (Lexical & Corpus Phase) – The 1990s

We can describe this as a lexical & corpus phase. The phase had a lexicalized approach to grammar that appeared in late 1980s and became an increasing influence. There was a revolution in natural language processing in this decade with the introduction of machine learning algorithms for language processing.

Study of Human Languages

Language is a crucial component for human lives and also the most fundamental aspect of our behavior. We can experience it in mainly two forms – written and spoken. In the written form, it is a way to pass our knowledge from one generation to the next. In the spoken form, it is the primary medium for human beings to coordinate with each other in their day-to-day behavior. Language is studied in various academic disciplines. Each discipline comes with its own set of problems and a set of solution to address those.

Consider the following table to understand this:

Discipline	Problems	Tools
Linguists	How phrases and sentences can be formed with words? What curbs the possible meaning for a sentence?	Intuitions about well-formedness and meaning. Mathematical model of structure. For example, model theoretic semantics, formal language theory.
Psycholinguists	How human beings can identify the structure of sentences? How the meaning of words can be identified? When does understanding take place?	Experimental techniques mainly for measuring the performance of human beings. Statistical analysis of observations.
Philosophers	How do words and sentences acquire the meaning? How the objects are identified by the words? What is meaning?	Natural language argumentation by using intuition. Mathematical models like logic and model theory.
Computational Linguists	How can we identify the structure of a sentence How knowledge and reasoning can be modeled? How we can use language to accomplish specific tasks?	Algorithms Data structures Formal models of representation and reasoning. AI techniques like search & representation methods.

Ambiguity and Uncertainty in Language

Ambiguity, generally used in natural language processing, can be referred as the ability of being understood in more than one way. In simple terms, we can say that ambiguity is the capability of being understood in more than one way. Natural language is very ambiguous. NLP has the following types of ambiguities:

Lexical Ambiguity

The ambiguity of a single word is called lexical ambiguity. For example, treating the word **silver** as a noun, an adjective, or a verb.

Syntactic Ambiguity

This kind of ambiguity occurs when a sentence is parsed in different ways. For example, the sentence "The man saw the girl with the telescope". It is ambiguous whether the man saw the girl carrying a telescope or he saw her through his telescope.

Semantic Ambiguity

This kind of ambiguity occurs when the meaning of the words themselves can be misinterpreted. In other words, semantic ambiguity happens when a sentence contains an ambiguous word or phrase. For example, the sentence "The car hit the pole while it was moving" is having semantic ambiguity because the interpretations can be "The car, while moving, hit the pole" and "The car hit the pole while the pole was moving".

Anaphoric Ambiguity

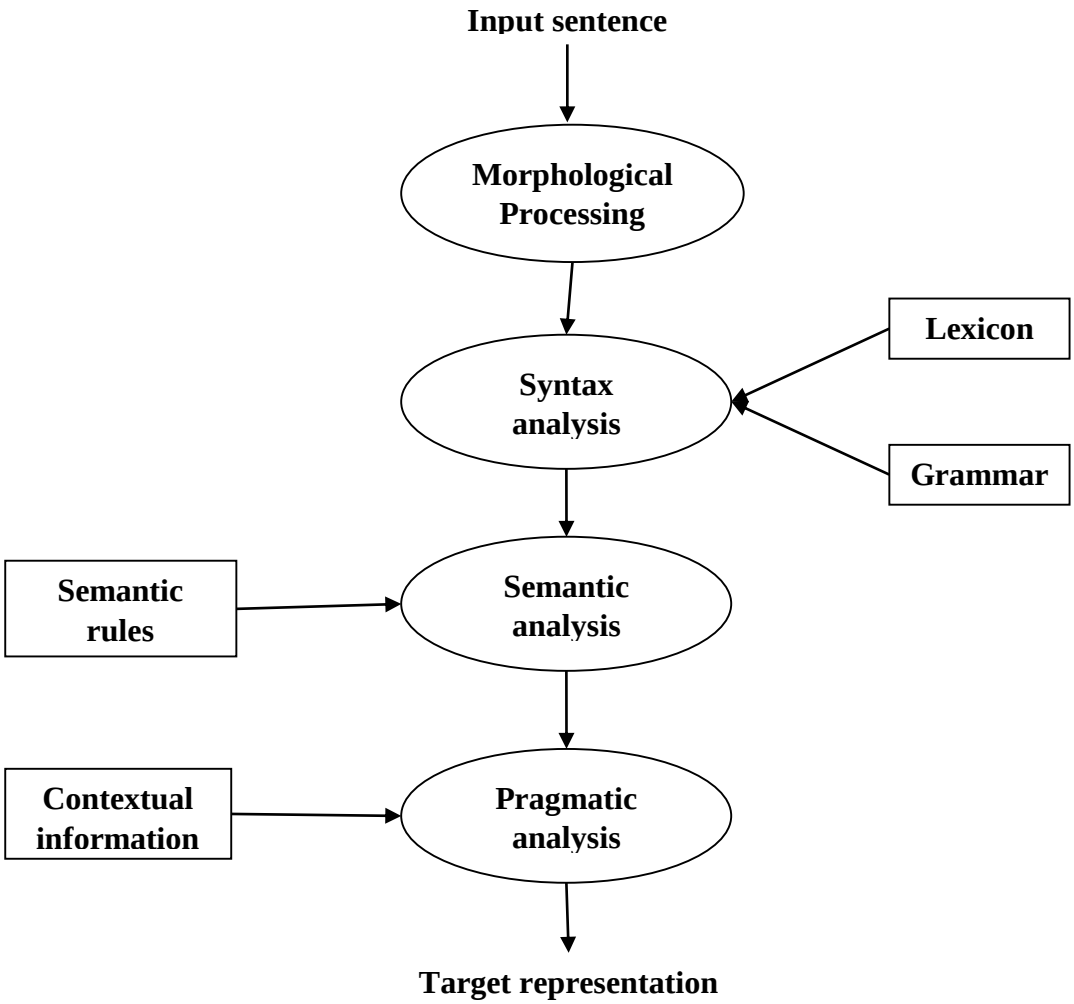
This kind of ambiguity arises due to the use of anaphora entities in discourse. For example, the horse ran up the hill. It was very steep. It soon got tired. Here, the anaphoric reference of "it" in two situations cause ambiguity.

Pragmatic ambiguity

Such kind of ambiguity refers to the situation where the context of a phrase gives it multiple interpretations. In simple words, we can say that pragmatic ambiguity arises when the statement is not specific. For example, the sentence "I like you too" can have multiple interpretations like I like you (just like you like me), I like you (just like someone else dose).

NLP Phases

Following diagram shows the phases or logical steps in natural language processing:



Morphological Processing

It is the first phase of NLP. The purpose of this phase is to break chunks of language input into sets of tokens corresponding to paragraphs, sentences and words. For example, a word like **"uneasy"** can be broken into two sub-word tokens as **"un-easy"**.

Syntax Analysis

It is the second phase of NLP. The purpose of this phase is two folds: to check that a sentence is well formed or not and to break it up into a structure that shows the syntactic relationships between the different words. For example, the sentence like **"The school goes to the boy"** would be rejected by syntax analyzer or parser.

Semantic Analysis

It 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. The text is checked for meaningfulness. For example, semantic analyzer would reject a sentence like "Hot ice-cream".

Pragmatic Analysis

It is the fourth phase of NLP. Pragmatic analysis simply fits the actual objects/events, which exist in a given context with object references obtained during the last phase (semantic analysis). For example, the sentence "Put the banana in the basket on the shelf" can have two semantic interpretations and pragmatic analyzer will choose between these two possibilities.

2. Natural Language Processing — Linguistic Resources

In this chapter, we will learn about the linguistic resources in Natural Language Processing.

Corpus

A corpus is a large and structured set of machine-readable texts that have been produced in a natural communicative setting. Its plural is *corpora*. They can be derived in different ways like text that was originally electronic, transcripts of spoken language and optical character recognition, etc.

Elements of Corpus Design

Language is infinite but a corpus has to be finite in size. For the corpus to be finite in size, we need to sample and proportionally include a wide range of text types to ensure a good corpus design.

Let us now learn about some important elements for corpus design:

Corpus Representativeness

Representativeness is a defining feature of corpus design. The following definitions from two great researchers – Leech and Biber, will help us understand corpus representativeness:

- **According to Leech (1991),** "A corpus is thought to be representative of the language variety it is supposed to represent if the findings based on its contents can be generalized to the said language variety".

- **According to Biber (1993),** "Representativeness refers to the extent to which a sample includes the full range of variability in a population".

In this way, we can conclude that representativeness of a corpus are determined by the following two factors:

- **Balance –** The range of genre include in a corpus.
- **Sampling –** How the chunks for each genre are selected.

Corpus Balance

Another very important element of corpus design is corpus balance – the range of genre included in a corpus. We have already studied that representativeness of a general corpus depends upon how balanced the corpus is. A balanced corpus covers a wide range of text categories, which are supposed to be representatives of the language. We do not have any reliable scientific measure for balance but the best estimation and intuition works in this concern. In other words, we can say that the accepted balance is determined by its intended uses only.

Sampling

Another important element of corpus design is sampling. Corpus representativeness and balance is very closely associated with sampling. That is why we can say that sampling is inescapable in corpus building.

- According to **Biber(1993),** "Some of the first considerations in constructing a corpus concern the overall design: for example, the kinds of texts included, the number of texts, the selection of particular texts, the selection of text samples from within texts, and the length of text samples. Each of these involves a sampling decision, either conscious or not."

While obtaining a representative sample, we need to consider the following:

- **Sampling unit:** It refers to the unit which requires a sample. For example, for written text, a sampling unit may be a newspaper, journal or a book.
- **Sampling frame:** The list of al sampling units is called a sampling frame.
- **Population:** It may be referred as the assembly of all sampling units. It is defined in terms of language production, language reception or language as a product.

Corpus Size

Another important element of corpus design is its size. How large the corpus should be? There is no specific answer to this question. The size of the corpus depends upon the purpose for which it is intended as well as on some practical considerations as follows:

- Kind of query anticipated from the user.
- The methodology used by the users to study the data.
- Availability of the source of data.

With the advancement in technology, the corpus size also increases. The following table of comparison will help you understand how the corpus size works:

Year	Name of the Corpus	Size (in words)
1960s-70s	Brown and LOB	1 Million words
1980s	The Birmingham corpora	20 Million words
1990s	The British National corpus	100 Million words
Early 21st century	The Bank of English corpus	650 Million words

In our subsequent sections, we will look at a few examples of corpus.

TreeBank Corpus

It may be defined as linguistically parsed text corpus that annotates syntactic or semantic sentence structure. Geoffrey Leech coined the term 'treebank', which represents that the most common way of representing the grammatical analysis is by means of a tree structure. Generally, Treebanks are created on the top of a corpus, which has already been annotated with part-of-speech tags.

Types of TreeBank Corpus

Semantic and Syntactic Treebanks are the two most common types of Treebanks in linguistics. Let us now learn more about these types -

Semantic Treebanks

These Treebanks use a formal representation of sentence's semantic structure. They vary in the depth of their semantic representation. Robot Commands Treebank, Geoquery, Groningen Meaning Bank, RoboCup Corpus are some of the examples of Semantic Treebanks.

Syntactic Treebanks

Opposite to the semantic Treebanks, inputs to the Syntactic Treebank systems are expressions of the formal language obtained from the conversion of parsed Treebank data. The outputs of such systems are predicate logic based meaning representation. Various syntactic Treebanks in different languages have been created so far. For example, **Penn Arabic Treebank**, **Columbia Arabic Treebank** are syntactic Treebanks created in Arabia language. **Sininca** syntactic Treebank created in Chinese language. **Lucy**, **Susane** and **BLLIP WSJ** syntactic corpus created in English language.

Applications of TreeBank Corpus

Followings are some of the applications of TreeBanks:

In Computational Linguistics

If we talk about Computational Linguistic then the best use of TreeBanks is to engineer state-of-the-art natural language processing systems such as part-of-speech taggers, parsers, semantic analyzers and machine translation systems.

In Corpus Linguistics

In case of Corpus linguistics, the best use of Treebanks is to study syntactic phenomena.

In Theoretical Linguistics and Psycholinguistics

The best use of Treebanks in theoretical and psycholinguistics is interaction evidence.

PropBank Corpus

PropBank more specifically called "Proposition Bank" is a corpus, which is annotated with verbal propositions and their arguments. The corpus is a verb-oriented resource; the annotations here are more closely related to the syntactic level. Martha Palmer et al., Department of Linguistic, University of Colorado Boulder developed it. We can use the term PropBank as a common noun referring to any corpus that has been annotated with propositions and their arguments.

In Natural Language Processing (NLP), the PropBank project has played a very significant role. It helps in semantic role labeling.

VerbNet(VN)

VerbNet(VN) is the hierarchical domain-independent and largest lexical resource present in English that incorporates both semantic as well as syntactic information about its contents. VN is a broad-coverage verb lexicon having mappings to other lexical resources such as WordNet, Xtag and FrameNet. It is organized into verb classes extending Levin classes by refinement and addition of subclasses for achieving syntactic and semantic coherence among class members.

Each VerbNet (VN) class contains:

A set of syntactic descriptions or syntactic frames

For depicting the possible surface realizations of the argument structure for constructions such as transitive, intransitive, prepositional phrases, resultatives, and a large set of diathesis alternations.

A set of semantic descriptions such as animate, human, organization

For constraining, the types of thematic roles allowed by the arguments, and further restrictions may be imposed. This will help in indicating the syntactic nature of the constituent likely to be associated with the thematic role.

WordNet

WordNet, created by Princeton is a lexical database for English language. It is the part of the NLTK corpus. In WordNet, nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called **Synsets**. All the synsets are linked with the help of conceptual-semantic and lexical relations. Its structure makes it very useful for natural language processing (NLP).

In information systems, WordNet is used for various purposes like word-sense disambiguation, information retrieval, automatic text classification and machine translation. One of the most important uses of WordNet is to find out the similarity among words. For this task, various algorithms have been implemented in various packages like Similarity in Perl, NLTK in Python and ADW in Java.

3. Natural Language Processing — Word Level Analysis

In this chapter, we will understand world level analysis in Natural Language Processing.

Regular Expressions

A regular expression (RE) is a language for specifying text search strings. RE helps us to match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are used to search texts in UNIX as well as in MS WORD in identical way. We have various search engines using a number of RE features.

Properties of Regular Expressions

Followings are some of the important properties of RE:

- American Mathematician Stephen Cole Kleene formalized the Regular Expression language.
- RE is a formula in a special language, which can be used for specifying simple classes of strings, a sequence of symbols. In other words, we can say that RE is an algebraic notation for characterizing a set of strings.
- Regular expression requires two things, one is the pattern that we wish to search and other is a corpus of text from which we need to search.

Mathematically, A Regular Expression can be defined as follows –

- **ε** is a Regular Expression, which indicates that the language is having an empty string.
- **φ** is a Regular Expression which denotes that it is an empty language.
- If **X** and **Y** are Regular Expressions, then
    - **X, Y**
    - **X.Y(Concatenation of XY)**
    - **X+Y (Union of X and Y)**
    - **X*, Y* (Kleen Closure of X and Y)**

are also regular expressions.

- If a string is derived from above rules then that would also be a regular expression.

Examples of Regular Expressions

The following table shows a few examples of Regular Expressions:

Regular Expressions	Regular Set
(0 + 10*)	{0, 1, 10, 100, 1000, 10000, … }
(0*10*)	{1, 01, 10, 010, 0010, …}
(0 + ε)(1 + ε)	{ε, 0, 1, 01}
(a+b)*	It would be set of strings of a's and b's of any length which also includes the null string i.e. {ε, a, b, aa , ab , bb , ba, aaa…….}
(a+b)*abb	It would be set of strings of a's and b's ending with the string abb i.e. {abb, aabb, babb, aaabb, ababb, ……..}
(11)*	It would be set consisting of even number of 1's which also includes an empty string i.e. {ε, 11, 1111, 111111, …….}
(aa)*(bb)*b	It would be set of strings consisting of even number of a's followed by odd number of b's i.e. {b, aab, aabbb, aabbbbb, aaaab, aaaabbb, ……..}
(aa + ab + ba + bb)*	It would be string of a's and b's of even length that can be obtained by concatenating any combination of the strings aa, ab, ba and bb including null i.e. {aa, ab, ba, bb, aaab, aaba, ……..}

Regular Sets & Their Properties

It may be defined as the set that represents the value of the regular expression and consists specific properties.

Properties of regular sets

- If we do the union of two regular sets then the resulting set would also be regular.
- If we do the intersection of two regular sets then the resulting set would also be regular.

- If we do the complement of regular sets, then the resulting set would also be regular.
- If we do the difference of two regular sets, then the resulting set would also be regular.
- If we do the reversal of regular sets, then the resulting set would also be regular.
- If we take the closure of regular sets, then the resulting set would also be regular.
- If we do the concatenation of two regular sets, then the resulting set would also be regular.

Finite State Automata

The term automata, derived from the Greek word "αὐτόματα" meaning "self-acting", is the plural of automaton which may be defined as an abstract self-propelled computing device that follows a predetermined sequence of operations automatically.

An automaton having a finite number of states is called a Finite Automaton (FA) or Finite State automata (FSA).

Mathematically, an automaton can be represented by a 5-tuple (Q, Σ, δ, q0, F), where –

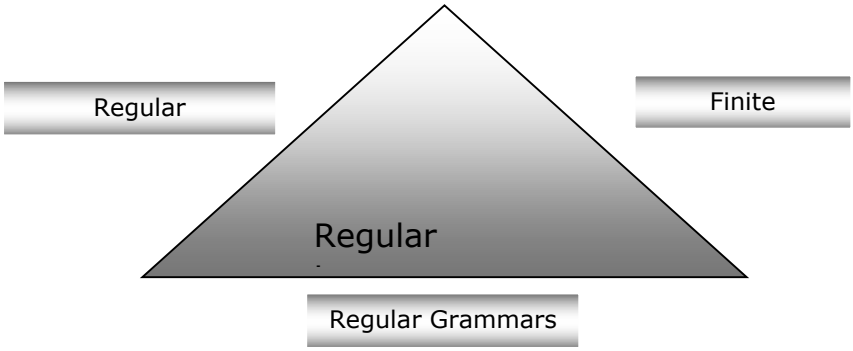
- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function.
- q0 is the initial state from where any input is processed (q0 ∈ Q).
- F is a set of final state/states of Q (F ⊆ Q).

Relation between Finite Automata, Regular Grammars and Regular Expressions

Following points will give us a clear view about the relationship between finite automata, regular grammars and regular expressions:

- As we know that finite state automata are the theoretical foundation of computational work and regular expressions is one way of describing them.
- We can say that any regular expression can be implemented as FSA and any FSA can be described with a regular expression.
- On the other hand, regular expression is a way to characterize a kind of language called regular language. Hence, we can say that regular language can be described with the help of both FSA and regular expression.
- Regular grammar, a formal grammar that can be right-regular or left-regular, is another way to characterize regular language.

Following diagram shows that finite automata, regular expressions and regular grammars are the equivalent ways of describing regular languages.



Types of Finite State Automation (FSA)

Finite state automation is of two types. Let us see what the types are.

Deterministic Finite automation (DFA)

It may be defined as the type of finite automation wherein, for every input symbol we can determine the state to which the machine will move. It has a finite number of states that is why the machine is called Deterministic Finite Automaton (DFA).

Mathematically, a DFA can be represented by a 5-tuple (Q, Σ, δ, q0, F), where –

- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function where δ: Q × Σ → Q .
- q0 is the initial state from where any input is processed (q0 ∈ Q).
- F is a set of final state/states of Q (F ⊆ Q).

Whereas graphically, a DFA can be represented by diagraphs called state diagrams where –

- The states are represented by **vertices**.
- The transitions are shown by labeled **arcs**.
- The initial state is represented by an **empty incoming arc**.
- The final state is represented by **double circle**.

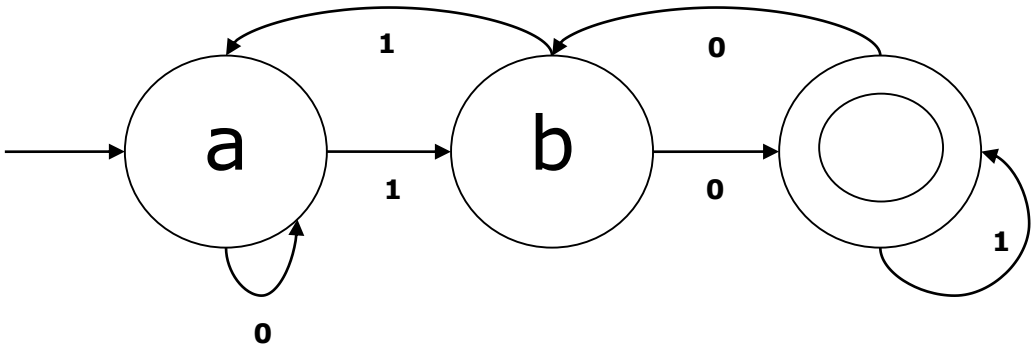
Example of DFA

Suppose a DFA be

- Q = {a, b, c},
- Σ = {0, 1},
- q_0 = {a},
- F = {c},
- Transition function δ is shown in the table as follows:-

Current State	Next State for Input 0	Next State for Input 1
A	a	B
B	c	A
C	b	C

The graphical representation of this DFA would be as follows:



Non-deterministic Finite Automation (NDFA)

It may be defined as the type of finite automation where for every input symbol we cannot determine the state to which the machine will move i.e. the machine can move to any combination of the states. It has a finite number of states that is why the machine is called Non-deterministic Finite Automation (NDFA).

Mathematically, NDFA can be represented by a 5-tuple (Q, Σ, δ, q0, F), where –

- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ :-is the transition function where δ: $Q \times \Sigma \rightarrow 2^Q$.
- q0 :-is the initial state from where any input is processed ($q0 \in Q$).
- F :-is a set of final state/states of Q ($F \subseteq Q$).

Whereas graphically (same as DFA), a NDFA can be represented by diagraphs called state diagrams where –

- The states are represented by **vertices**.
- The transitions are shown by labeled **arcs**.
- The initial state is represented by an **empty incoming arc**.
- The final state is represented by **double circle**.

Example of NDFA

Suppose a NDFA be

- Q = {a, b, c},
- Σ = {0, 1},
- q_0 = {a},
- F = {c},
- Transition function δ is shown in the table as follows: -

Current State	Next State for Input 0	Next State for Input 1
A	a, b	B
B	C	a, c
C	b, c	C

The graphical representation of this NDFA would be as follows: