# Instruction to annotate the Ground Truth

Prepared by: Chadni Islam

**Before Starting:** The annotator needs to have a basic understanding of English language linguistic features to understand the syntactic and semantic structure of a sentence. One way to gain that understanding is the use of dependency parsing.

i. Dependency parsing extracts the dependency parse tree of a sentence that represents the grammatical structure of a sentence and defies the relationship between root words of a sentence and words which modify the root.
   a. The dependency parsing will also provide the syntactical and semantic parsing of a sentence structure.
   b. Syntactical parsing provides the parse tree
   c. The semantic analysis provides the subject, object and different attributes of a sentence.
   d. Several types of dependency (i.e., exist between different words of a sentence

ii. One way to automatically identify the dependency parse tree is to use the *spaCy* or NLTK *library* to identify the dependency and part of speech (POS) of different words in a sentence.

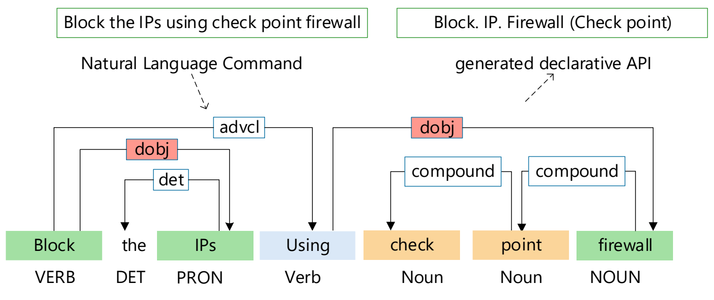
## 1. Preliminary knowledge required to annotate the data

If the annotator required preliminary knowledge, ***Figure 1*** provides a detailed example of some of the key dependencies. Following are some of the key terminologies are described that need to be considered for annotating.

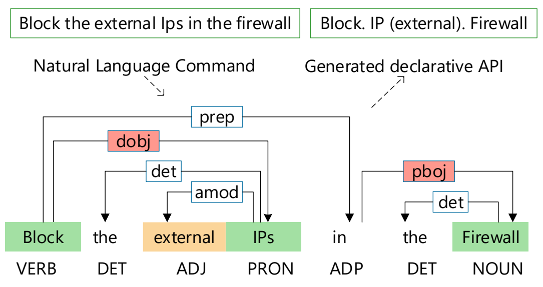
- **Root:** The root of a dependency parse tree is considered as an ancestor of all the other nodes of the parse tree. Usually, the root word is used to identify the main verb of the sentence. (for example, See **Figure 1**)
- **Direct object:** The direct object of a VP is the noun phrase which is the (accusative) object of the verb. (for example, see ***Figure 1***)
- **Prepositional object:** The object of a preposition is the head of a noun phrase following the preposition, or the adverbs "here" and "there". (The preposition, in turn, may be modifying a noun, verb, etc.) (for example, see ***Figure 1***)
- **Nominal subject:** A nominal subject is a noun phrase which is the syntactic subject of a clause. Nominal subject modifies a non-verbal predicate of a sentence. Example.
    - "Clinton defeated Dole" nsubj(defeated, **Clinton**)

  Here Clinton is the nominal subject of defeat.
- **Adverbial clause modifier (advcl):** An adverbial clause modifier of a verb phrase or subject is a clause modifying the verb (temporal clause, consequence, conditional clause, purpose clause, etc.). (for example, see ***Figure 1***)
- **Adverb modifier:** An adverb modifier of a word is a (non-clausal) adverb or adverb-headed phrase that serves to modify the meaning of the word. Example
    - "Genetically modified food" advmod(modified, **genetically**)

  Here **genetically** is the adverb modifier of modified.
- **Adjectival modifier (amod):** An adjectival modifier of a noun phrase is an adjectival phrase that serves to modify the meaning of the NP. (for example, see ***Figure 1***)
- **Clausal complement:** A clausal complement of a verb or adjective is a dependent clause with an internal subject which functions like an object of the verb, or adjective. Clausal complements for nouns are limited to complement clauses with a subset of nouns like "fact" or "report". Example:



(a) dependency tree for "block the IPs using check point firewall"



(b) dependency tree showing the dependency and POS for "block the external IPs in the firewall"

| Token | Token dep | POS of Token | Token head | Token head dep | POS of Token head | Token child |
|---|---|---|---|---|---|---|
| Block | root | VERB | Block | root | VERB | IP, In |
| The | det | DET | IPs | dobj | PROP | - |
| External | amod | ADJ | IPs | dobj | PROPN | - |
| IPs | dobj | PROP | Block | root | VERB | The, External |
| In | Prep | ADP | Block | root | VERB | Firewall |
| The | det | DET | Firewall | pboj | NOUN | - |
| Firewall | pobj | NOUN | In | prep | ADP | the |

(c) The different linguistic associated with a words (here token) of sentence "block the external IPs in the firewall"

**Figure 1 Example of dependency parsing for the sentence (a) block the IPs using checkpoint Firewall and (b) block the external IP in the Firewall and (c) other linguistic features of a token (each word is considered as a token) such as token head, token**

| | |
|---|---|
| ○ "He says that you like to swim" | ccomp(says, **like**) |
| ○ "I am certain that he did it" | ccomp(certain, **did**) |
| ○ "I admire the fact that you are honest" | ccomp(fact, **honest**) |

- **Noun compound modifier (compound)**: A noun compound modifier of a noun phrase is any noun that serves to modify the head noun. (Note that in the current system for dependency extraction, all nouns modify the rightmost noun of the NP – there is no intelligent noun compound analysis. This is likely to be fixed once the Penn Treebank represents the branching structure of NPs.) (for example, see ***Figure 1***)

## 2. Annotation Instruction

### 2.1. Annotate the different parts of an API element from the task description

The annotator needs to take each task description from the "*raw_file.xls*" and annotate the different parts (i.e., API element) of an API element. Certain things need to consider

- Each annotated API will have three parts.
- The **first part**, consists of the key abstract functions such as *block, scan, verify* and *detonate*.
- The **second part** provides details about the task and compromises of the object (e.g., *IP* and *capability*) on which the task needs to be performed.
- The **third part** is to identify the specific components (i.e., *endpoint* and *ontology*) on which the task needs to be done or the types of tools (i.e., *firewall*) need to be used.
- The last two parts of an API take **parameters** that provide more fine-grained detail about a task. For example, the task "*block external IPs in the Firewall*" specifies the types of IPs. The API for the task will be "*block. IP (external). firewall*" where the second part takes *"external"* as an input parameter. Follow the annotation instruction below to label the task with the corresponding API.

- The third part and the parameters of the second and third part of an API might be empty depending on the task description.

**Following are the step by step annotation instruction**

**Step 1:** Identify the Part of speech of a sentence from the list of the task description and the relationship between different words of a sentence or build the dependency parse tree as shown in **Figure 1**.

**Step 2: Annotation of the first part of the API**

i. Identify the root word of a sentence and consider the **root of a sentence as the first part** of an API.
ii. If a word is the nominal subject of the root of a sentence and the part of speech of the word is verb or noun then consider the **word as a first part** of the.

**Step 3: Annotation of the second part of the API**

i. If the root has a **nominal subject** where the parts of speech of the nominal subject are either verb or noun only then consider the **root** as a **second part** of the API
ii. If a word is a **direct object** of a root consider it as the **second part** of the API
iii. If a word is a modifier (adverbial modifier or clausal complement) of the root consider the child of the word that is a nominal subject of the word
    a. If the child is a **nominal subject** of the word and the part of speech of the child is a **verb** then consider it as the **second part** of the API

For annotation of the parameter of the second part of the API follow step 5

**Step 4: Annotation of the third part of the API**

i. If a word is a modifier (adverbial modifier or clausal complement) of the root consider the child of the word that is a nominal subject of the word
    a. If a child is a **direct object** of the word and the part of speech of the child is a **verb** then consider it as the **third part** of the API
ii. If a word is a preposition of the root and the part of speech of the word is the adverbial position then consider the child of the word that is a prepositional object of the word
    a. If the child of the word is a **preposition object** and its **part of speech is a verb or noun** consider the **child** of the word as a **third part** of the API

For annotation of the parameter of the third part of the API follow step 5

**Step 5: Annotation of the parameter of the second and third part of the API**

Each object and subject of the second and third part of the API (also referred to as API element) have further modifiers (e.g., adverbial modifier, adjective modifier and noun compound modifier).

ii. Consider the modifier of an API element as the parameter of that API element. For example, *block the external IPs in the Firewall,* here *external* is a modifier (i.e., **adjective modifier**) of *IP*.
iii. Several cases exist where a single object or subject has multiple dependencies, for example, "*send a message to the source user email address*", here the *"source user email"* is the modifier of the object *addres*s and parameter of the address API element. Identify the multiple modifiers of the API element and consider them as the list of parameters for that part. Example: **address (source user email)**

**Save the file as "annotated_API_*AnnotatorName*.xls"**

**Tips:** Annotate each part for all the task together than annotating all the part for a single task. It helps to speed up the process.

## 2.2. Combine the semantically similar words

Some of the annotated API element can be combined together into a single API element based on their semantics and the task they are performing. For example, "*quarantine the endpoint*" and "*isolate the endpoint*" referring to the same task. With the above annotation, there will be two different API elements in the first part: ***quarantine*** and ***isolate***. The job of the annotator here is to combine the API elements and present them with a single API element. For example: use ***quarantine*** instead of *isolate* or vice versa. In the new annotated API element, the API element *quarantine* refers to the task related to isolate and quarantine that is quarantine -> (quarantine, isolate)

- Quarantine the endpoint -> quarantine. endpoint
- Isolate the endpoint -> quarantine. endpoint

Similarity ***retrieve*** can be used instead of ***acquire***, and ***get***

**Instruction:**

- Take each API element and consider the rest of the API elements and see whether they seem to be the synonym of each other or tends to have a semantically similar meaning in the context of the task.

The annotator can provide the API element list that can be combined together in the following form and save the file with the name "**semanticAPI_AnnotatorName.docs**" / **"semantics_api_annotatorName.pdf"**

- quarantine -> (quarantine, isolate)
- retrieve -> (acquire, get, retrieve)

Also, change the annotation of the **"annotated_API_*AnnotatorName*.xls"** with the semantically similar API and save the new annotated file as "**annotate_Semantic_API_AnnotatorName.xls**".