First Assinment Natural Language Understanding

Wamiq Raza
Department of Information Engineering and Science
University of Trento
Trento, Italy
wamiq.raza@studenti.unitn.it

Giuseppe Riccardi
Department of Information Engineering and Science
University of Trento
Trento, Italy
giuseppe.riccardi@unitn.it

Evgeny Stepanov
Department of Information Engineering and Science
University of Trento
Trento, Italy
evgeny.stepanov@unitn.it

***Abstract*—In this report of assignment present a deterministic parsing function for dependency grammar. spaCy is the library used which is library for Natural language processing (NLP) in python with a lot of built-in capabilities. In total five function it covers how to extract the path of dependency relation from the root of token. For a given token subtree dependency is extracted and checked a segment of a sentence from subtree value in Boolean form. Moreover, head of token is identified and last function to extract subject, direct object and indirect object spans from a given sentence.**

Keywords: spacy, NLU, subtree, span, dependency graphs, subtree, head.

I. INTRODUCTION

Natural language understanding (NLU) is a field of computer science that studies how computers and humans interact. In the 1950, Alan Turing published an article that proposed a measure of intelligence, we called the Turing test. More modern techniques, such as deep learning, have produced results in the fields of language modeling, parsing, and natural-language tasks. The term 'dependency grammar' does not refer to a specific grammar formalism rather, it refers to a specific way to describe the syntactic structure of a sentence. Despite a long and esteemed heritage in language studies, dependency grammar has started playing a pretty minor role in natural language parsing until recently.

spaCy library is increasingly popular for processing and analyzing data in NLP. spaCy is written in Cpython and designed to build information extraction.

The report consists in five section in Section 2 description of first function, in Section 3 declaration details of second function, while in Section 4 description of third function, for Section 5 description of forth function and at last Section 6 description of last function.

II. FIRST FUNCTION

In the first function which is define as **(extract_Token_Path_Root)** for a given sentence function will run from head of token until the root and saving the path in path list which is declared. If it found root it will add the root and return the root. For each token in **spacy_doc** it will get its path using the function and once we call the function by passing sentence parameter it will return the ROOT. the syntactic

III. SECOND FUNCTION

For the second function which take one parameter and define as **(extracting_Subtree)** this function extracts the subtree and convert it into a list for every token in sentence.

IV. THRID FUNCTION

The third function looking for a given list of tokens from a subtree. It will check if the token list and subtree list are equal if equal it will return True. Function is defined as **(check_Subtree).**
After calling functions it give Boolean value for a given subtree output from second function.

V. FORTH FUNCTION

In this section you will find the description for forth function which is define as **(head_Of_List)** at the fist it join the words in the list of computer head of the span composed by the full parsed joined list of words. Then it convert the words to a parsed span and get its root and then it return the head of a sentence.

VI. FIFTH FUNCTION

In the last part of report function **(extract_Direct_Indirect_Span**) return subject, direct object, and indirect object spans from sentence. First it creates as dictionary of list for each dependency then it iterates over the dependency and all tokens. If it found dependency add its token to a corresponding list as a span object