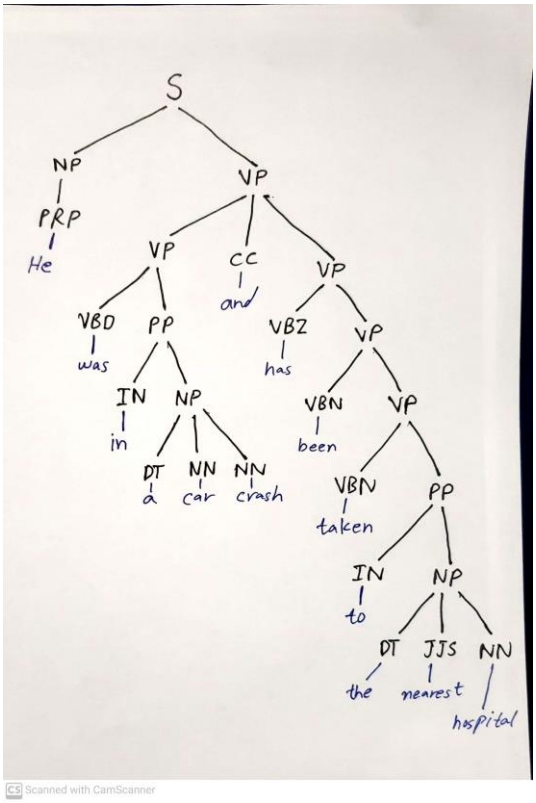
My Sentence: He was in a car crash and has been taken to the nearest hospital

PSG Tree:



Phase Terms and their Definitions:

CC: Coordinating Conjunction

DT: Determiner

IN: Preposition

JJS: Adjective, superlative

NN: Noun, singular

NP: Noun Phrase

PRP: Personal Pronoun

PP: Prepositional Phrase

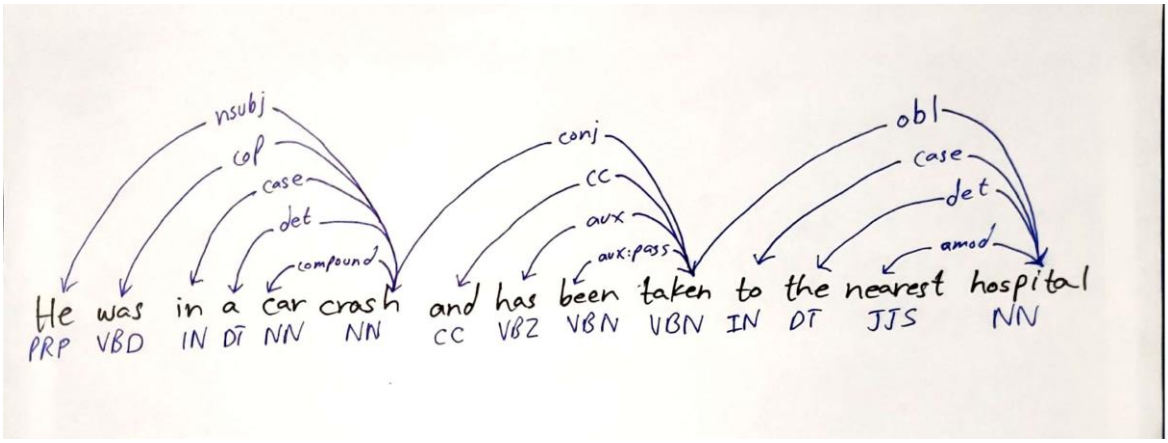
VBD: Verb, past tense

VBN: Verb, past participle

VBZ: Verb, 3rd person singular present

VP: Verb Phrase

Dependency Parse:



Phase Terms and their Definitions:

NSUBJ: Nominal Subject

COP: Copula

COMPOUND: Compound

CONJ: Conjunct

CC: Coordination

AUX: Auxiliary

AUX:PASS: Passive Auxiliary

OBL: Indirect Nominal

CASE: Case Marking

DET: Determiner

AMOD: Adjectival Modifier

SRL Parse:

Verb 1: was

Predicate: was in a car crash

ARG0: He

ARG1: in a car crash

Relation between the arguments: Arg0 is the agent of the sentence and Arg1 is something that happened to Arg0.

Verb 2: taken

Predicate: taken to the nearest hospital

ARG0: He

ARGM: to the nearest hospital

Tag of the Modifier: GOL

GOL: final destination of motion verbs

Relation between the argument and modifier: Arg0 is the agent of the sentence and ARGM is where the agent ended up.

Pros and Cons of Parsers:

In my opinion, PSG tree was easier to perform compared to the other two parsers, and it gave us a list of parts of speech which is very useful for many machine learning algorithms; on the other hand, PSG tree does not analyze the grammatical structure of the sentence. For the dependency parsing, it was hard to figure out the dependency relations between each word, but this gave us a list of POS and explored the dependencies between the words, which is better than PSG tree and is more useful for applications that require grammatical analysis. Lastly, when it comes to SRL parsing, it is very easy to implement, but I don’t know if it is as useful in machine learning algorithms, since it only identifies different actors and aspects of an action, but does not give us any information on the role of each word.