Biruk Mamo

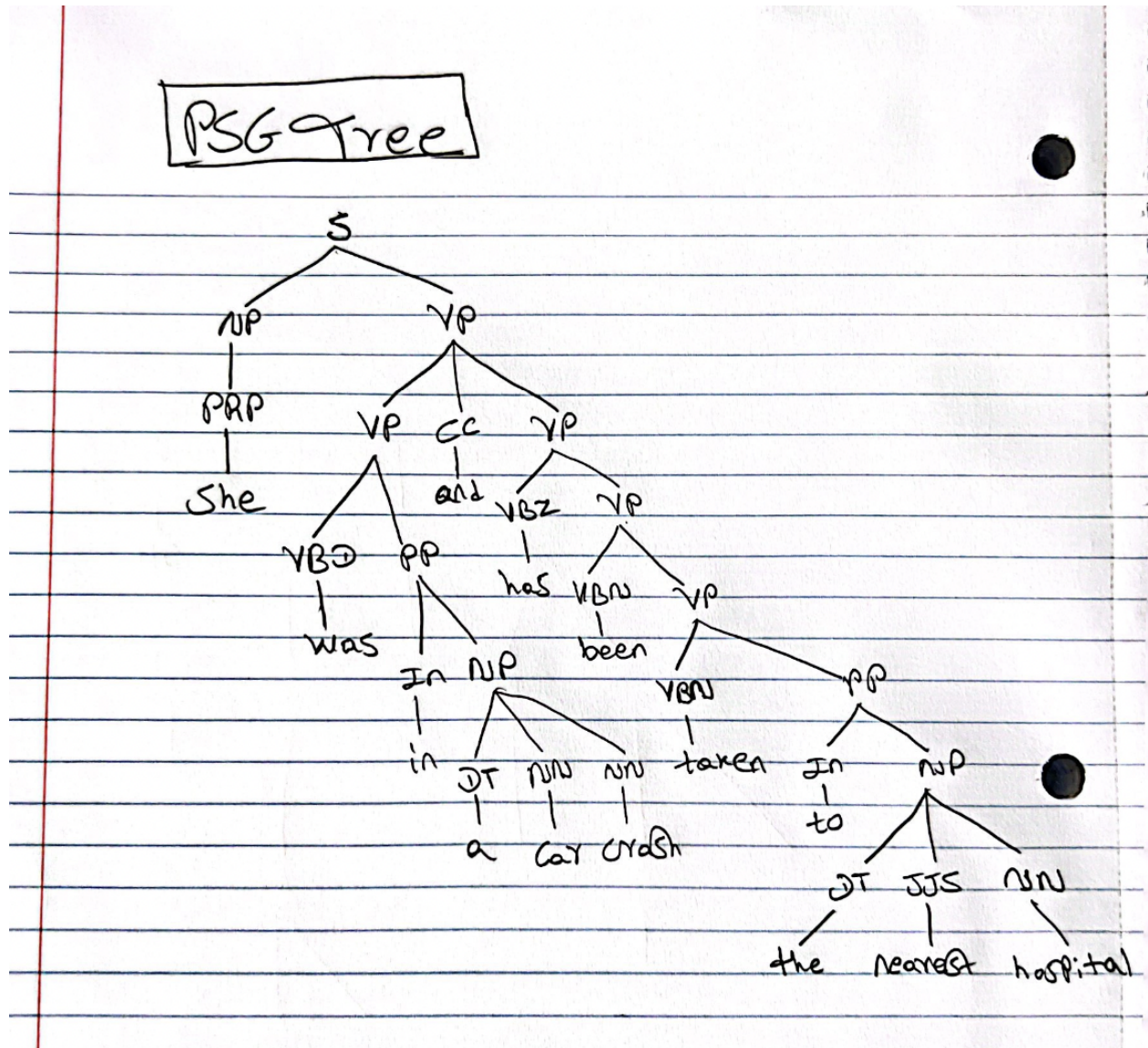**Sentence**: She was in a car crash and has been taken to the nearest hospital
PSG Tree:

Phase Terms and their Definitions:

CC: Coordinating Conjunction
DT: Determiner
IN: Preposition
JJS: Adjective, superlative
NN: Noun, singular
NP: Noun Phrase
PRP: Personal Pronoun
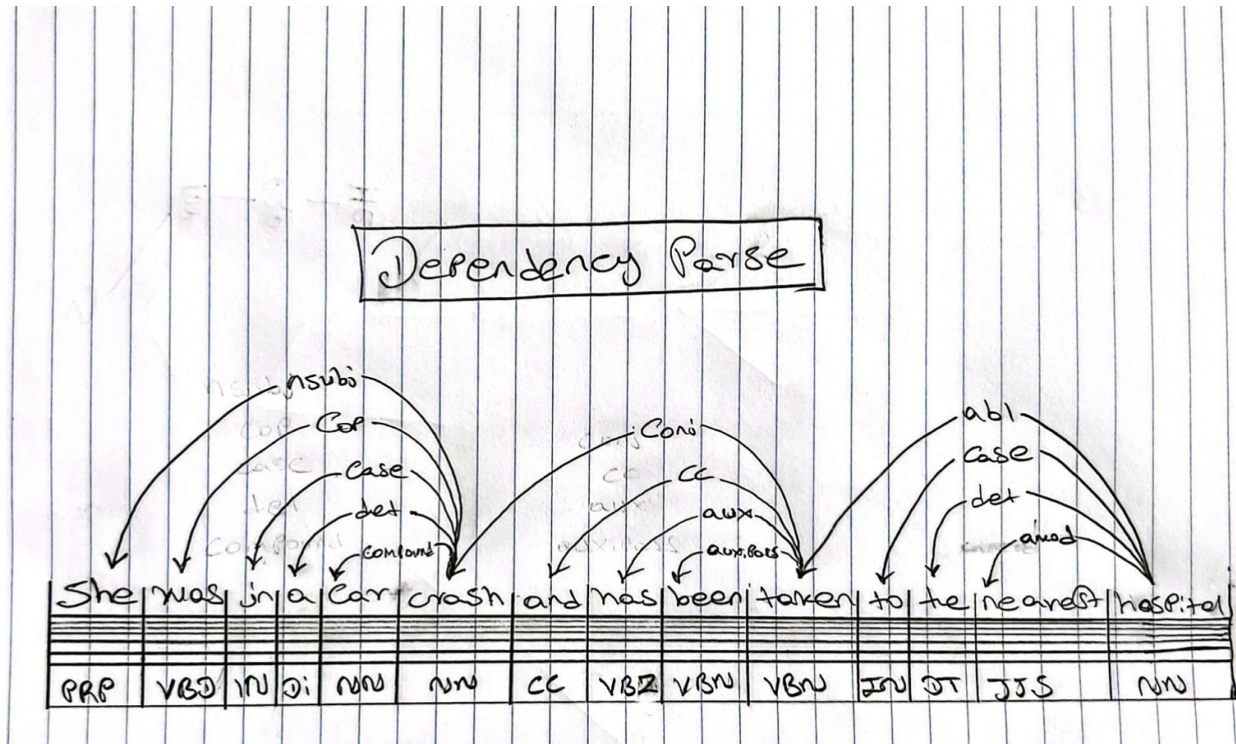**PP**: Prepositional Phrase
VBD: Verb, past tense
VBN: Verb, past participle
VBZ: Verb, 3rd person singular present
VP: Verb Phrase

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: She
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: She
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 view, the PSG tree was the simplest parser compared to the other two, and it provided us with a valuable list of parts of speech that is applicable to many machine learning algorithms. However, the PSG tree does not analyze the grammatical structure of the sentence. Dependency parsing, on the other hand, was challenging to determine the relationships between each word, but it gave us a list of parts of speech and explored the dependencies between the words, which is superior to the PSG tree and is more beneficial for applications requiring grammatical analysis. Lastly, SRL parsing is simple to implement, but I'm unsure if it's as valuable in machine learning algorithms

because it only recognizes different actors and aspects of an action without providing information on the role of each word.