DATA 641 - HW3

Name: Precious Worgu
Student ID: 119343890

Problem 1

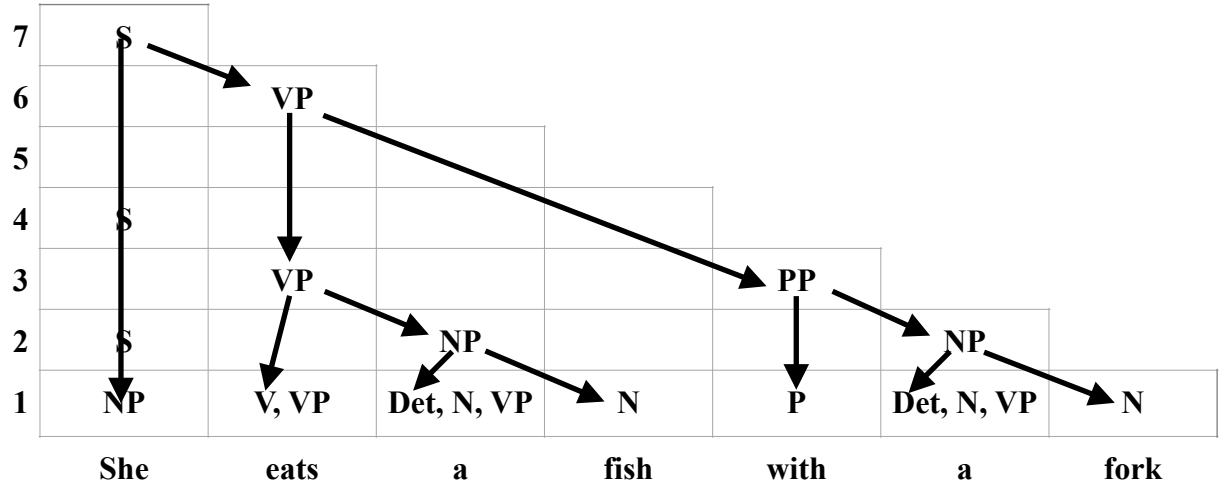
A.

1. CKY as a recognizer;

7	**S**						
6		**VP**					
5							
4	**S**						
3		**VP**			**PP**		
2	**S**		**NP**			**NP**	
1	**NP**	**V, VP**	**Det, N, VP**	**N**	**P**	**Det, N, VP**	**N**
	She	**eats**	**a**	**fish**	**with**	**a**	**fork**

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → She
V → eats
P → with
N → fish
N → fork
Det → a
N → a
NP → N N

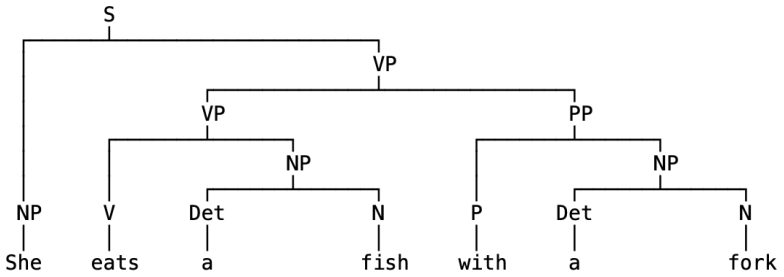
2. CKY as a parser;



B. There are four valid parse trees for the given sentence with the updated grammar.

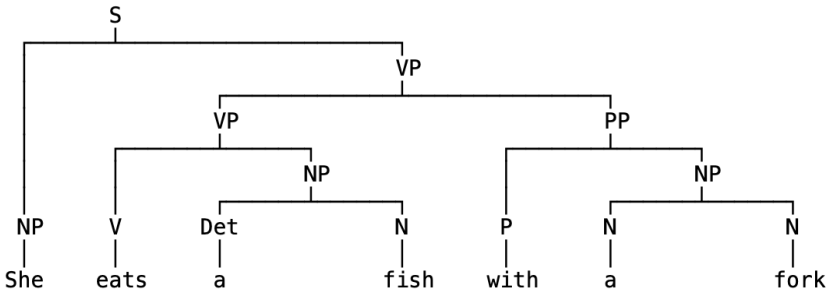
1. The first parse tree is:

(S
(NP She)
(VP
(VP (V eats) (NP (Det a) (N fish)))
(PP (P with) (NP (Det a) (N fork)))))



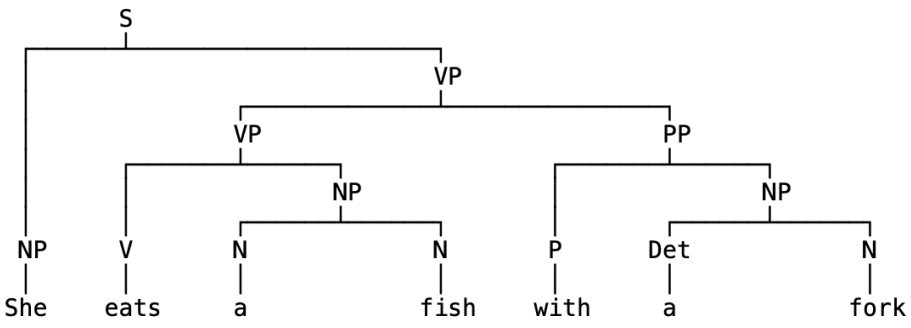
2. The second parse tree is:

(S
(NP She)
(VP
(VP (V eats) (NP (Det a) (N fish)))
(PP (P with) (NP (N a) (N fork)))))

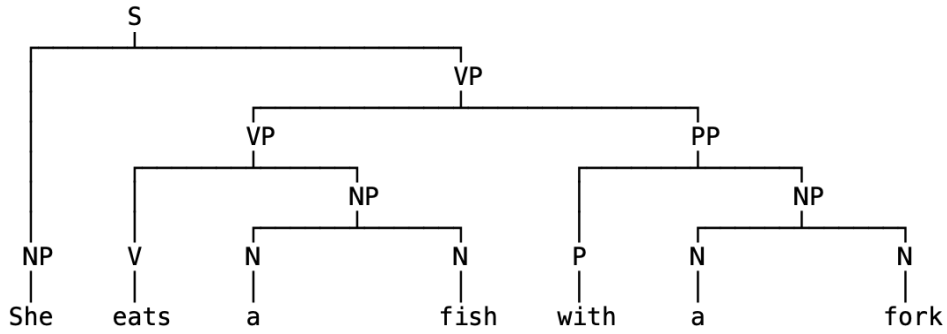


3. The third parse tree is:

(S
(NP She)
(VP
(VP (V eats) (NP (N a) (N fish)))
(PP (P with) (NP (Det a) (N fork)))))



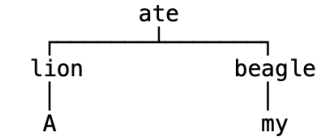
4. The fourth parse tree is:
   (S
    (NP She)
    (VP
     (VP (V eats) (NP (N a) (N fish)))
     (PP (P with) (NP (N a) (N fork)))))



Problem 2

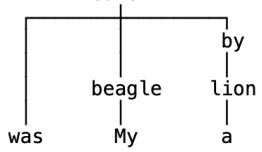
A. Syntactic dependency tree;
   1. A lion ate my beagle

```
Token    dependence head text  Dependency explained
A        det =>       lion       determiner
lion     nsubj =>     ate        nominal subject
ate      ROOT =>      ate        root
my       poss =>      beagle     possession modifier
beagle   dobj =>      ate        direct object
```



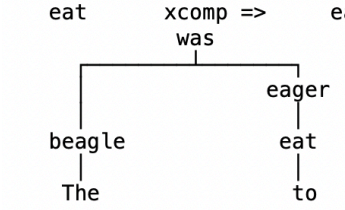
   2. My beagle was eaten by a lion

```
Token    dependence head text  Dependency explained
My       poss =>      beagle     possession modifier
beagle   nsubjpass =>   eaten      nominal subject (passive)
was      auxpass =>   eaten      auxiliary (passive)
eaten    ROOT =>      eaten      root
by       agent =>     eaten      agent
a        det =>       lion       determiner
lion     pobj =>      by         object of preposition
```



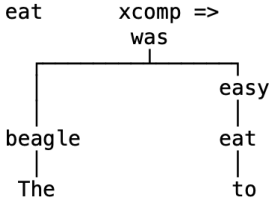
3. The beagle was eager to eat

```
Token    dependence head text  Dependency explained
The      det =>       beagle     determiner
beagle   nsubj =>     was        nominal subject
was      ROOT =>      was        root
eager    acomp =>     was        adjectival complement
to       aux =>       eat        auxiliary
eat      xcomp =>     eager      open clausal complement
```



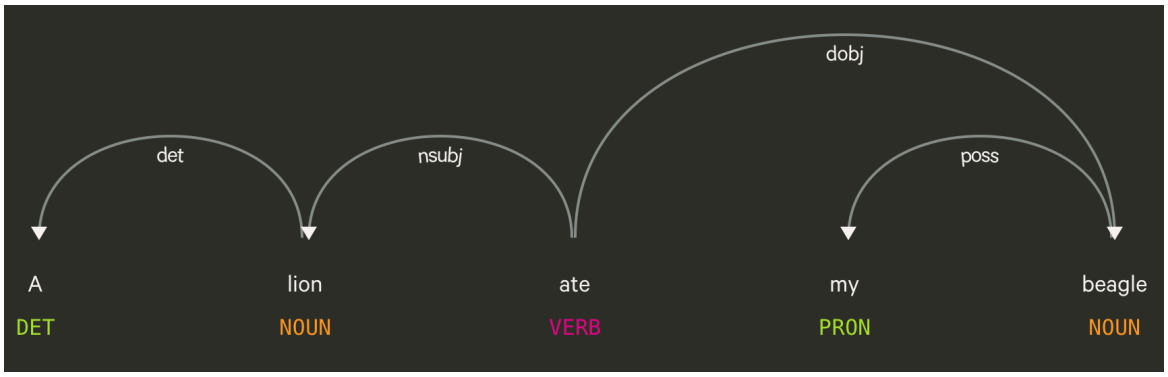
4. The beagle was easy to eat

```
Token    dependence head text  Dependency explained
The      det =>       beagle     determiner
beagle   nsubj =>     was        nominal subject
was      ROOT =>      was        root
easy     acomp =>     was        adjectival complement
to       aux =>       eat        auxiliary
eat      xcomp =>     easy       open clausal complement
```

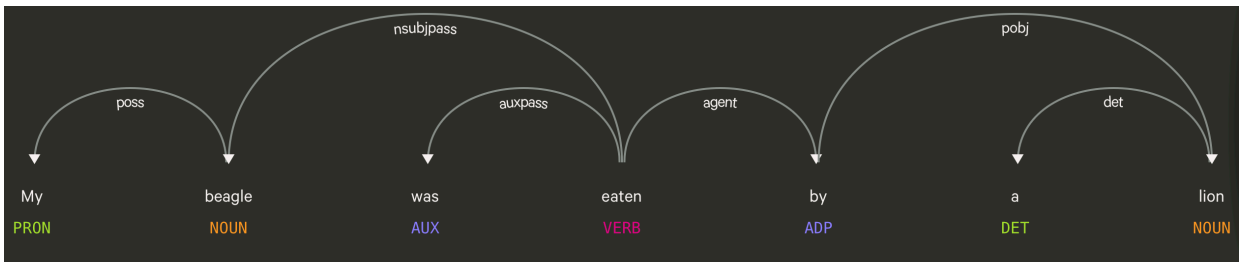


B. Online spaCy dependency parser demo;

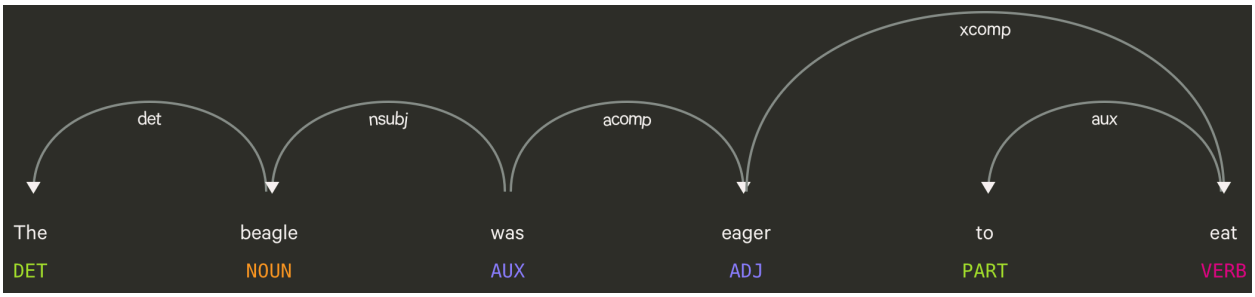
1. A lion ate my beagle



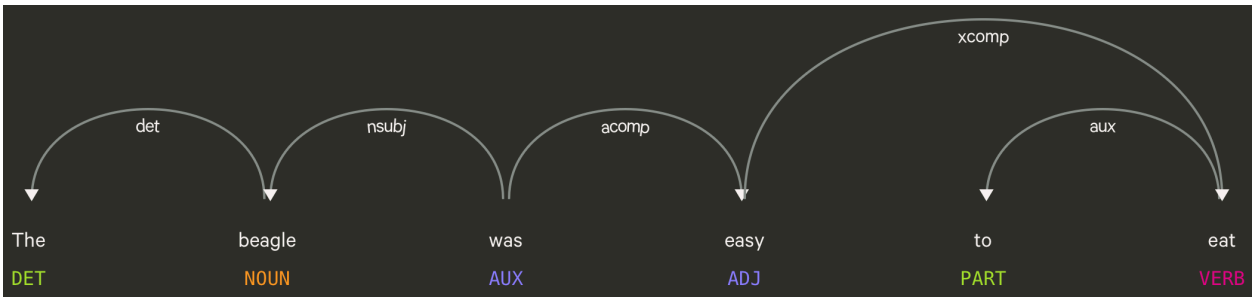
2. My beagle was eaten by a lion



3. The beagle was eager to eat



4. The beagle was easy to eat



- The difference between the syntactic dependency tree for the sentences and the output of the online spaCy dependency parser demo is mainly in the representation format.

  In the syntactic dependency tree for “A lion ate my beagle”, the words are arranged in a hierarchical structure where the main verb 'ate' is the root of the tree and the other words are its dependents. The subject 'lion' is connected to the verb, indicating that it is the agent performing the action. The object 'beagle' is connected to the verb, indicating that it is the patient affected by the action. The possessive pronoun 'my' is attached to the object, indicating that it modifies the noun ‘beagle’. In contrast, the output of the spaCy dependency parser demo represents the dependencies in a visual and interactive format. Each word is represented as a node with its part of speech and a label indicating its syntactic relationship to other words in the sentence. The parser correctly identifies the verb 'ate' as the root of the tree and assigns the subject 'lion' and the object 'beagle' as its dependents. Additionally, it recognizes the modifier 'my' as a possessive determiner that modifies the noun ‘beagle'.

  Furthermore, comparatively, the syntactic dependency tree and the output of the spaCy dependency parser demo are quite similar for "The beagle was eager to eat”, “My beagle was eaten by a lion”, and “The beagle was easy to eat”. Overall, both representations convey the same syntactic structure of the sentence, but they differ in their visual format and the level of detail provided; the spaCy dependency parser provides a more detailed and complex representation of the sentence's syntactic structure.

C. When consulting for a news agency with a large database of news reports and helping a reporter find stories about pets being eaten by wild animals, there are potential issues that need to be considered based on the observations from part b. One of the primary concerns is the possibility of false positives. This means that the spaCy parser may identify sentences as relevant even if they do not meet the criteria of pets being eaten by wild animals. For instance, a sentence such as "The lion was

killed by the hunter's dog" may be flagged as relevant due to the mention of a dog and a lion, but it does not actually fit the criteria.

Additionally, it is important to note that the spaCy parser has limitations in flagging relevant sentences. It may not identify sentences that do not contain the exact keywords or phrases that the reporter is searching for. For instance, a sentence such as "The lion attacked a small dog" which pertains to a domestic animal being preyed on by a wild animal may not be considered relevant if it does not contain the exact keywords or phrases being searched for. Therefore, it is crucial to keep in mind that the spaCy parser is not infallible and may miss out on some relevant information.

When facing these challenges, it would be beneficial to consider the following strategies:

- Employ human judgment: Despite the implementation of advanced search techniques, it is crucial to manually review the search results to guarantee their relevance to the search criteria. By having a human reviewer quickly scan through the results, any erroneous positives or overlooked opportunities can be eliminated.

- To enhance the efficiency and effectiveness of the search algorithm, it is essential to keep improving it. This can be achieved by continuously collecting and analyzing more data to refine the algorithm, which in turn leads to increased accuracy and reduced occurrences of false positives and missed opportunities. The process may entail re-training the spaCy parser on a more comprehensive dataset or integrating feedback from human reviewers to optimize the search algorithm.