Question 1. [2+2+1= 5 Marks]

Given the following simplified transition and emission probabilities for an HMM-based Part-of-Speech (POS) tagger:

Tag Transition Probabilities	Noun	Verb	Adj
Noun	0.4	0.3	0.3
Verb	0.1	0.8	0.1
Adj	0.6	0.3	0.1

Emission Probabilities	Noun	Verb	Adj
"run"	0.8	0.2	-
"fast"	0.6	-	0.4
"dog"	0.7	0.3	-

Question:

Find the most likely sequence of POS tags for the sentence "run fast" using the Viterbi algorithm.

Solution:

For transition probabilities:
P(Noun → Noun) = 0.4, P(Noun → Verb) = 0.3, P(Noun → Adj) = 0.3
P(Verb → Noun) = 0.1, P(Verb → Verb) = 0.8, P(Verb → Adj) = 0.1
P(Adj → Noun) = 0.6, P(Adj → Verb) = 0.3, P(Adj → Adj) = 0.1

For Emission probabilities:
For run: P(run | Noun) = 0.8, P(run | Verb) = 0.2, P(run | Adj) = 0
For fast: P(fast | Noun) = 0.6, P(fast | Verb) = -, P(fast | Adj) = 0.4

Now using Viterbi Algorithm,

Calculate the probability for word "run" for each tag (t=1):
V(Noun, "run") = P(Noun) × P(run | Noun) = 1/3 × 0.8=0.2667
V(Verb, "run") = P(Verb) × P(run | Verb) = 1/3 × 0.2=0.0667
V(Adj, "run") = P(Adj) × P(run | Adj) = 1/3 × 0=0

Calculate the probability for word "fast" for each tag (t=2):
For Noun:
From Noun: 0.2667 ×0.4 ×0.6=0.0640
From Verb: 0.0667 ×0.1 ×0.6=0.0040
From Adj: 0 ×0.6 ×0.6=0
Maximum value for Noun is 0.0640 (From Noun)
For Verb:
From Noun: 0.2667 ×0.3 ×Undefined (-)=0
From Verb: 0.0667 ×0.8 ×Undefined (-)=0
From Adj: 0 ×0.3 ×Undefined (-)=0
Maximum value for Verb is 0 (No valid emission probability from Verb)
For Adj:
From Noun: 0.2667 ×0.3 ×0.4=0.0320
From Verb: 0.0667 ×0.1 ×0.4=0.0027
From Adj: 0 ×0.1 ×0.4=0
Maximum value for Adj is 0.0320 (From Noun)

From this we can infer that the highest probability for the last word "fast" is for Noun (0.0640) followed by Adj (0.0320).

The backtracking path shows that both the words “run” and “fast” are most likely tagged as “Noun”.

Hence the most likely sequence of POS tags for the sentence “run fast” is [Noun, Noun]

Q2. Given the following transition and emission probabilities for an HMM:

Transition Probabilities	S1	S2
S1	0.6	0.4
S2	0.3	0.7

Emission Probabilities	S1	S2
“A”	0.5	0.5
“B”	0.4	0.6

Using the Viterbi algorithm, calculate the most probable hidden state sequence for the observation sequence “A B”.
SOLUTION:
Step 1
Initialization: Calculate initial probabilities for states S1 and S2 for the first observation 'A'. P(S1)=0.5×0.5=0.25 P(S2)=0.5×0.5=0.25
Explanation:
The initial probabilities are calculated by multiplying the initial state probabilities (assumed equal here for simplicity) with the emission probabilities for 'A'. Both states have an equal initial probability of 0.25.
Step 2
Recursion: Calculate probabilities for the next observation 'B'. For state S1: P(S1)=max(0.25×0.6×0.4,0.25×0.3×0.4)=0.06 For state S2: P(S2)=max(0.25×0.4×0.6,0.25×0.7×0.6)=0.105
Explanation:
The probabilities are calculated by considering the maximum probability path to each state. For S1, the paths are from S1 and S2 from the previous observation. The same applies for S2.
Step 3
Termination: Determine the most probable state sequence. Since P(S2)=0.105 is greater than P(S1)=0.06, the most probable state for the last observation is S2.
Explanation:
The termination step involves choosing the state with the highest probability for the last observation. Here, S2 has the highest probability.

Question 3. [6 Marks]

Consider the statement: “The flying fish jumped out of the water”
Using Depth First, Simple Top Down Parsing method, work out the steps only till the noun-phrase “The flying fish” is successfully parsed. Provide one line explanation for each step.
The grammar rules to be applied and the lexicon are given below.

Rule No	Rule	Rule No	Rule
1	S → NP VP	8	PP → P PP
2	NP → Det N	9	Det → 'the'
3	NP → Det Adj N	10	Adj → 'flying'
4	VP → V	11	N → 'fish' | 'water'
5	VP → V NP	12	V → 'jumped'
6	VP → V PP	13	P → 'out' | 'of'
7	PP → P NP		

Lexicon

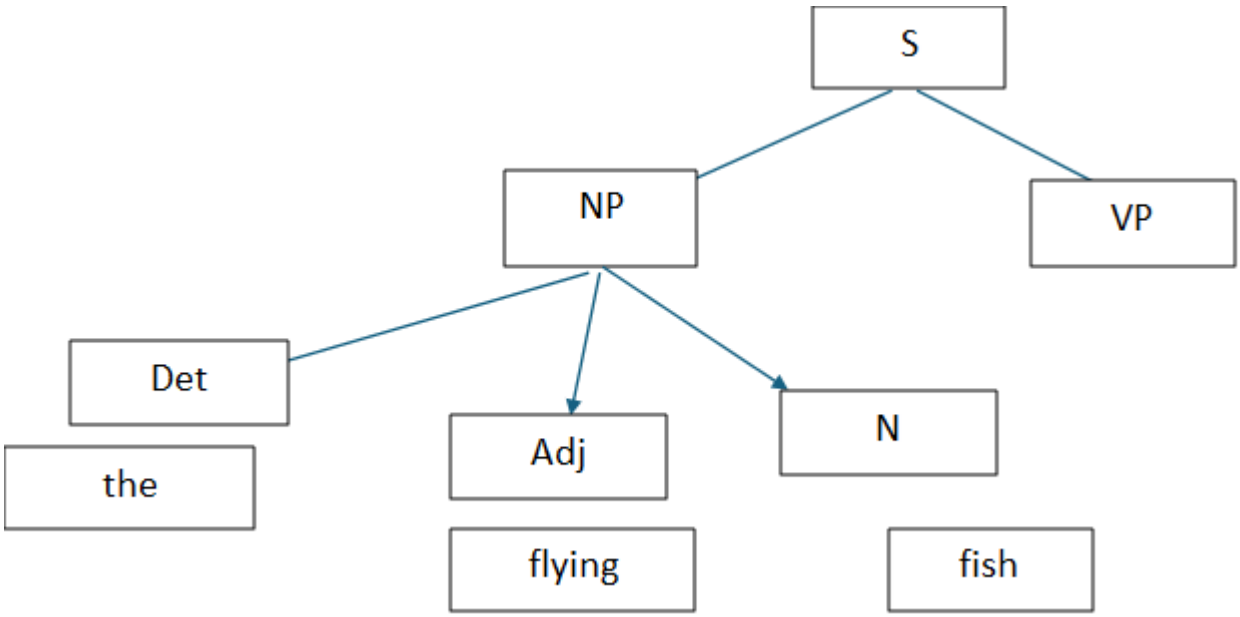
- **Det (Determiner):** the
- **Adj (Adjective):** flying
- **N (Noun):** fish, water
- **V (Verb):** jumped
- **P (Preposition):** out, of

Solution:

Sentence: "$_1$The $_2$flying $_3$fish $_4$jumped $_5$out $_6$of $_7$the $_8$water$_9$"

Step no	State	Backup	Comments
1	((S) 1)		
2	((NP VP) 1)		
3	((Det N VP) 1)	(Det Adj N VP) 1)	'the' is Det
4	~~((N VP) 2)~~	(Det Adj N VP) 1)	Second position does not have N, hence discard
5	(Det Adj N VP) 1)		Pop from backup. 'the' is Det
6	((Adj N VP) 2)		'flying' is Adj
7	((N VP) 3)		'fish' is N
8	((VP) 4)		'the flying fish' is fully parsed

Parsed tree :



Question 4. [3 + 3 = 6 Marks]

a. Highlight the differences between Constituency Parsing and Dependency Parsing based on the following aspects: i) Basic structure ii) Applications iii) Parsing methods

b. What will be the Constituency Parse Tree and the Dependency Parse Tree for the statement "**The man shouted loudly**"? (Only show the final tree, derivation steps are not expected).

In the Constituency Parse Tree, the relevant grammar rules among the following may be used.

```
S     -> NP VP
NP    -> Det N
VP    -> V AdvP
VP    -> V NP
VP    -> V PP
AdvP -> Adv
```

Question 5. [4+2=6 Marks]

a. Show how you would disambiguate the following sentence using the Simple Lesk approach. Describe the algorithm and show how it would apply in this instance. [4 marks]

The room was dark and there was no light though it was summer morning.

Sense 1

- (noun) The natural agent that stimulates sight and makes things visible.
  The morning light filtered softly through the curtains.

Sense 2

- (verb) To ignite something, causing it to start burning.
  He used a match to light the candle.

Sense 3

- (adjective) Having a considerable or sufficient amount of natural light; the opposite of dark.
  The room was light and airy, perfect for a summer's day.

Sense 4

- (adjective) Of low weight; not heavy.
  The suitcase was surprisingly light, making it easy to carry.

Solution :
LESK Algorithm - 1M
Identifying the overlaps - 2M
Identifying the correct Sense - 1M

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

  best-sense ← most frequent sense for word
  max-overlap ← 0
  context ← set of words in sentence
  for each sense in senses of word do
    signature ← set of words in the gloss and examples of sense
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
        max-overlap ← overlap
        best-sense ← sense
  end
  return(best-sense)
```

Overlap with Sense 1 (noun):
Overlap: morning
Sense 1 mentions morning in the example, which matches the context.

Overlap with Sense 2 (verb):
Overlap: None
There are no overlapping words between the context and this sense.

Overlap with Sense 3 (adjective):
Overlap: room, summer
Sense 3 mentions room, summer which match the context of the sentence.

Overlap with Sense 4 (adjective):
Overlap: None

The Simple Lesk algorithm would choose Sense 3 (adjective: *having a considerable or sufficient amount of natural light*) because it has the highest number of overlapping words

b. How can we use ontology and knowledge graph for a search application [2 marks]

Solution : Following points to be covered (0.5M each)

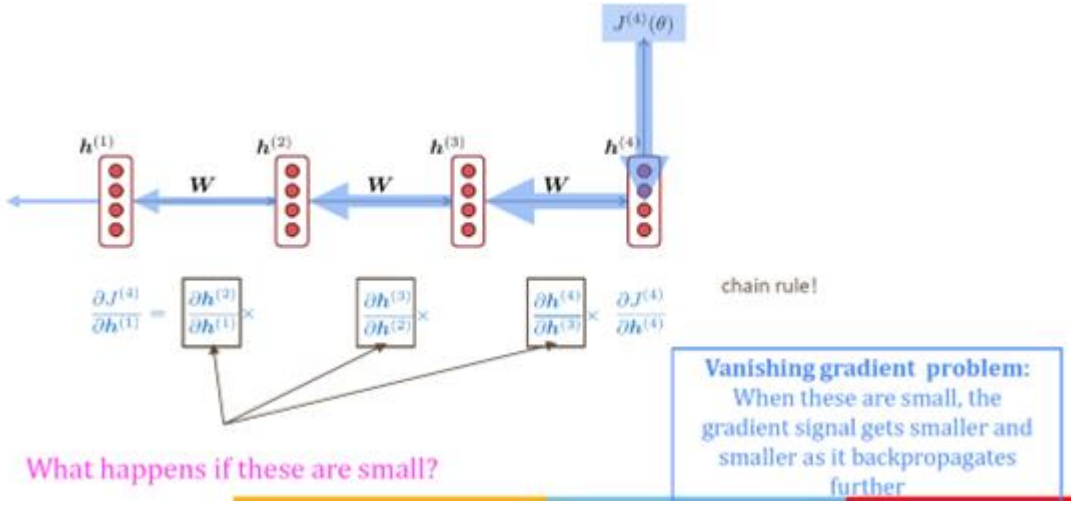
- Domain Knowledge Modeling (Ontology Creation)
- Data Integration into the Knowledge Graph
- Semantic Search
- Disambiguation using context

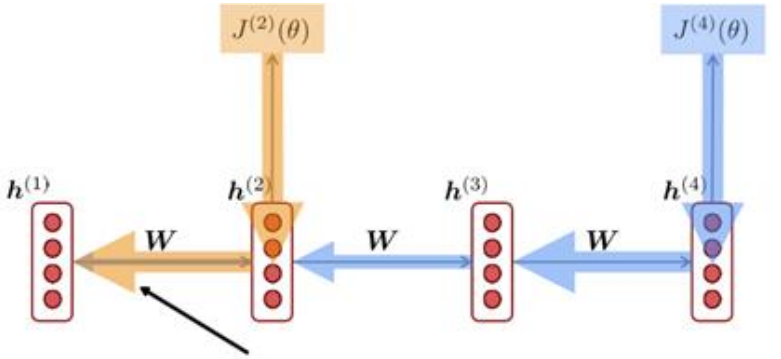
Question 6.

a. Explain the Vanishing Gradients problem and how does it impact the RNNs and the Encoder-Decoder architecture? Support your answer with relevant diagrams. (1+1 marks)

Solution: Vanishing Gradients is a problem faced during training RNNs which arises from the need to backpropagate the error signal back through time. The hidden layer at time t contributes to the loss at the next time step since it takes part in that calculation. As a result, during the backward pass of training, the hidden layers are subject to repeated multiplications, as determined by the length of the sequence. A frequent result of this process is that the gradients are eventually driven to zero, a situation called the vanishing gradients problem.

Diagramatically:





Gradient signal from far away is lost because it's much smaller than gradient signal from close-by.

So, model weights are basically updated only with respect to near effects, not long-term effects.

- **LM task:** *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her* _____

- To learn from this training example, the RNN-LM needs to model the dependency between *"tickets"* on the 7th step and the target word *"tickets"* at the end.

- But if the gradient is small, the model can't learn this dependency
  - So, the model is unable to predict similar long-distance dependencies at test time

- In practice a simple RNN will only condition ~7 tokens back [vague rule-of-thumb]

b. Given the sentence: "Dream Big" and information below, demonstrate the mathematical application of a single attention head to derive Self Attention. Support your answer with diagrams and relevant mathematical equations, detailing all steps. Apply score normalization, masking and softmax.

Embeddings:

	Token
Dream	[1 1]
Big	[1 0]

Weight Matrix for Query W^q

[0 1

1 0]

Weight Matrix for Keys W^k

[1 0

0 1]

Weight Matrix for Values W^v

[1 2

2 1]

- Exp(1.41) = 4.1
- Compute all fractions only to Second Decimal place (X.XX)

Solution:

(Marking scheme in brackets against each step)

1. Input Embedding:

	Token
Dream	[1 1]
Big	[1 0]

2. Compute Q, K & V (1.5 marks)

Q

$$\begin{matrix} 1 \\ 0 \end{matrix} \quad \begin{matrix} 1 \\ 1 \end{matrix}$$

K

$$\begin{matrix} 1 \\ 1 \end{matrix} \quad \begin{matrix} 1 \\ 0 \end{matrix}$$

V

$$\begin{matrix} 3 \\ 1 \end{matrix} \quad \begin{matrix} 3 \\ 2 \end{matrix}$$

3. Compute QK^T (0.5 marks)

2	1

1	0

4. Apply Normalization with d_k=2 (0.5 marks)

1.41	0.71
0.71	0.00

5. Apply Masking (0.5 marks)

1.41	0.71
0.71	0.00

6. Apply Softmax (0.5 marks)

formula for softmax

1.00	0.00
0.67	0.33

7. Multiply above Weights with V (Final Answer) (0.5 marks)

Self-Attention =

3.00	3.00
2.34	2.67

Question 7. Text Summarization

For the generic text summarization, given below 5 documents in the collection as per the chronological order of mentions, answer the following questions sequentially. Here are the documents:

Document 1 (d1): “There lived a shepherd boy.”

Document 2 (d2): “He cried out loudly, Wolf Wolf "

Document 3 (d3): "Once lived a shepherd boy.”

Document 4 (d4): “Shepherd cried wolf”

Document 5 (d5): "Shepherd cried loudly”

a) Preprocess the training document to remove the below set of tokens. Note: Ignore word cases but retain all the words. Stop words & punctuations: {the, a, he, there, was, for, is, '.', ','}

[1 Mark]

Solution:

Document 1 (d1): “lived shepherd boy”

Document 2 (d2): “cried out loudly Wolf Wolf "

Document 3 (d3): "Once lived shepherd boy”

Document 4 (d4): “Shepherd cried wolf”

Document 5 (d5): "Shepherd cried loudly”

b) Compute the saliency score of every sentence using the sum of weights (normalized by size of pre-processed sentence) of all the words in the above preprocessed sentences. Use only the new simplified formula to compute word’s weightage.

Weight (word Wi) = TF(i,j) * N/DF(i)

Solution:

N = Number of document given in the collection 5

TF(i,j) = Frequency of word i in document j

DF(i) = Number of documents containing word i

Word	TF(d1)	TF(d2)	TF(d3)	TF(d4)	TF(d5)	DF
lived	1	0	1	0	0	2
shepherd	1	0	1	1	1	4
boy	1	0	1	0	0	2
cried	0	1	0	1	1	3
out	0	1	0	0	0	1
loudly	0	1	0	0	1	2
Wolf	0	2	0	1	0	2
Once	0	0	1	0	0	1

Weight (word Wi) = TF(i,j) * N/DF(i)

Document1 : lived shepherd boy

lived = 1* 5/2 = 2.5

shepherd = 1* 5/4 = 1.25

boy = 1* 5/2 = 2.5

Total weight = 2.5+1.25+2.5 = 6.25

Saliency Score (normalized by 4 words): 6.25 / 3 = 2.08

Document 2 (d2): “cried out loudly Wolf Wolf

Cried = 1*5/3 = 1.66

out = 1*5/1 = 5

loudly = 1*5/2 = 2.5

Wolf = 2*5/2 = 5

Total weight = 1.66+5+2.5+5 = 14.16

Saliency Score (normalized by 5 words): 14.16 / 5= 2.832

Document 3 (d3): "once lived shepherd boy"

Once = 1 * 5/1 = 5

lived = 1 * 5/2 = 2.5

shepherd = 1 * 5/4 = 1.25

boy = 1 * 5/2 = 2.5

Total weight: 5+2.5+1.25+2.5=11.25

Saliency Score (normalized by 4 words): 11.25 / 4= 2.81

Document 4 (d4): "shepherd cried wolf"

shepherd: 1 * 5/ 4 =1.25

cried: 1*5/3 =1.67

wolf: 1*5/2 =2.5

Total weight: 1.25+1.67+2.5=5.42

Saliency Score (normalized by 3 words): 5.42/3=1.806

Document 5 (d5): "shepherd cried loudly"

shepherd: 1*5/4 =1.25

cried: 1*5/3 =1.67

loudly: 1*5/2 =2.5

Total weight: 1.25+1.67+2.5=5.42

Saliency Score (normalized by 3 words): 5.42 /3 =1.806

c) Extract summary in reverse chronological order with the top 2 most informative documents from part b)’s result.

Based on the saliency scores, the top 2 documents in reverse chronological order are:

Document 2 (d2): Saliency Score = 2.83

Document 3 (d3): Saliency Score = 2.81

The extracted summary would be from d2 and d3 in reverse order:

"Once lived a shepherd boy."

"He cried out loudly, Wolf Wolf."

d) Use ROUGE-1 to evaluate the results of the above system summary obtained in part b) w.r.t the below reference summary.

“Shepherd cried loudly, wolf”

Unigrams in System Summary:

once, lived, shepherd, boy, cried, out, loudly, wolf, wolf

Unigrams in Reference Summary:

shepherd, cried, loudly, wolf

ROUGE-1 Precision: 4/ 9 = 0.44
ROUGE-1 Recall: 4 /4 = 1
ROUGE-1 F1-Score: = 0.61

ROUGE-1 Precision:

$$\text{Precision} = \frac{\text{Number of overlapping unigrams}}{\text{Total unigrams in system summary}} =$$

ROUGE-1 Recall:

$$\text{Recall} = \frac{\text{Number of overlapping unigrams}}{\text{Total unigrams in reference summary}} =$$

ROUGE-1 F1-Score:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$