Descriptive:

UNIT-3

* How does lexical syntax help determine part-of-speech tags, and can you provide examples?
* How does the Expectation-Maximization algorithm improve part-of-speech tagging models?
* What is the role of the forward algorithm in Hidden Markov Models for sequence labelling?

UNIT-4

* Explain the concept of dependency parsing in deep learning. (6 marks)
* Explain how semi-supervised learning techniques can improve model performance in deep learning tasks. (6 marks)
* A. Define multi-task learning and its application in deep learning. (6 marks)
* B. Discuss the role of activation functions in deep learning models and their impact on model performance. (6 marks)
* A. Define multi-task learning and its application in deep learning. (6 marks)
* B. Discuss the role of activation functions in deep learning models and their impact on model performance. (6 marks)

UNIT – 5

* Explain how Named Entity Recognition (NER) works in Information Extraction using sequence labelling techniques. (6 marks)
* Discuss the principles behind phrase-based translation and its advantages over other translation approaches. (6 marks)
* A. Describe the basic issues encountered in Machine Translation and how they impact translation quality. (6 marks)

B. Discuss the process of relation extraction in Information Extraction tasks and its significance in extracting structured information from unstructured text. (6 marks)

* A. Explain the role of statistical learning methods in improving the performance of Machine Translation systems. (6 marks)
* B. Define word alignment in the context of Machine Translation and its importance in aligning source and target language words. (6 marks)

Open Book

**UNIT – 3**

* Explain How GloVe generates word embeddings by optimizing word vectors based on co-occurrence probabilities, efficiently capturing semantic relationships within the vocabulary?
* Word2vec learns word embeddings by predicting a word's context from neighbouring words, updating embeddings to reflect semantic relationships in the vocabulary.

**UNIT- 4**

* Discuss about Activation functions in NLP deep learning enable non-linearity, with ReLU offering simplicity and faster training despite potential vanishing gradient issues in deeper networks?
* Gradient descent checks validate gradient accuracy in NLP deep learning, enhancing parameter optimization stability, but they incur computational overhead and may overlook subtle errors.

**UNIT- 5**

* Explain the role of statistical learning in Machine Translation and what is statistical machine translation (SMT) and its key components. Illustrate with examples and diagrams.?
* Explain a step-by-step explanation of Named Entity Recognition (NER) in Information Extraction with examples?

1. How does lexical syntax help determine part-of-speech tags, and can you provide examples?

Ans:

Lexical syntax plays a crucial role in determining part-of-speech (POS) tags, which are labels assigned to words in a sentence based on their grammatical categories and functions. Here's how lexical syntax influences POS tagging with examples:

Word Structure: The structure of a word often provides clues about its part of speech. For instance:

Nouns typically have singular and plural forms (e.g., "cat" and "cats").

Verbs often have various forms indicating tense, aspect, and mood (e.g., "run," "ran," "running").

Adjectives might have comparative and superlative forms (e.g., "big," "bigger," "biggest").

Word Endings: Certain word endings can indicate the part of speech. For example:

Words ending in "-ly" are often adverbs (e.g., "quickly," "happily").

Words ending in "-ed" or "-ing" are typically verb forms (e.g., "walked," "walking").

Prefixes and Suffixes: Affixes added to words can also signal their part of speech:

Prefixes like "un-" often indicate negation (e.g., "unhappy," "unthinkable").

Suffixes like "-able" suggest adjectives (e.g., "comfortable," "adaptable").

Position in Sentence: The position of a word within a sentence can provide contextual clues for POS tagging:

Words occurring before articles ("a," "an," "the") are likely nouns (e.g., "apple," "cat").

Words following auxiliary verbs are often verb forms (e.g., "has eaten," "will run").

Contextual Information: Sometimes, the context of a word within a sentence or a larger text can help determine its part of speech:

In the phrase "bright blue sky," "blue" is an adjective describing "sky."

In the sentence "She runs quickly," "quickly" functions as an adverb modifying the verb "runs."

By analyzing these lexical features and considering the surrounding context, natural language processing systems can accurately assign part-of-speech tags to words in text data, enabling various downstream applications such as syntactic parsing, information extraction, and machine translation.

Ans : Dependency Representation: Dependency parsing represents sentences as directed graphs, where words are nodes and labeled edges represent syntactic relationships. For example, consider the sentence "The cat chased the mouse." The dependency tree for this sentence might look like this:

chased (root)

/ \

Cat mouse

/ \ / \

The the the the

Here, "chased" is the root word, and it has two dependencies: "cat" and "mouse."

1. Neural Network Architecture: Deep learning models for dependency parsing utilize neural network architectures to capture word dependencies. Graph-based neural networks or recursive neural networks are commonly used. These architectures process the input sentence to predict dependencies.
2. Feature Extraction: Features extracted from the input sentence include word embeddings, part-of-speech tags, and syntactic features. For example, the word "chased" might have a word embedding vector, a part-of-speech tag indicating it is a verb, and syntactic features representing its relationship with other words in the sentence.
3. Training and Optimization: The model is trained using annotated data, adjusting its parameters to minimize the difference between predicted and actual dependencies. This process involves optimizing a loss function, typically using techniques like stochastic gradient descent. The loss function measures the disparity between predicted and actual dependencies.
4. Inference: Once trained, the model predicts dependencies for new sentences. For example, given the input sentence "The dog chased the ball," the model predicts the dependencies between words. The predicted dependencies can be represented as a dependency tree similar to the example provided earlier.
5. Evaluation: Dependency parsing models are evaluated based on their accuracy in predicting dependencies. Common metrics include labeled attachment score (LAS) and unlabeled attachment score (UAS). LAS measures the percentage of correctly predicted dependencies with correct labels, while UAS measures the percentage of correctly predicted dependencies without considering labels.
6. Overall, dependency parsing in deep learning involves utilizing neural networks to analyze the grammatical structure of sentences, with each word's dependencies represented as labeled edges in a directed graph. This approach enables accurate syntactic analysis of text data, facilitating various natural language processing tasks.

===========================================================================

1. Using the example sentence "The students eagerly attended the lecture on natural language processing," explain how the Word2vec algorithm processes the input to generate word embeddings. Provide a detailed step-by-step explanation of the algorithm's operation, discussing each stage and its significance in learning distributed representations of words

The Word2vec algorithm is instrumental in generating word embeddings, facilitating numerous natural language processing (NLP) tasks. Let's delve into how Word2vec processes the example sentence "The students eagerly attended the lecture on natural language processing" to generate word embeddings:

Data Preparation:

Tokenize the sentence into individual words: ["The", "students", "eagerly", "attended", "the", "lecture", "on", "natural", "language", "processing"].

Create a vocabulary of unique words: {"The", "students", "eagerly", "attended", "the", "lecture", "on", "natural", "language", "processing"}.

Word Representation Initialization:

Initialize word vectors for each word in the vocabulary. These vectors can be initialized randomly or with pre-trained embeddings.

Context-Target Pair Generation:

Define a context window for each word. Assuming a window size of 2:

For the word "students": (context: ["The", "eagerly"], target: "students").

For the word "eagerly": (context: ["students", "attended", "the"], target: "eagerly").

For the word "attended": (context: ["eagerly", "the", "lecture"], target: "attended").

And so on for each word in the sentence.

Neural Network Architecture:

Choose between the Continuous Bag of Words (CBOW) or Skip-gram architecture. Let's consider the Skip-gram architecture.

Skip-gram predicts context words given the target word.

Model Training:

Train the neural network using the generated context-target pairs.

Update the word vectors through backpropagation and gradient descent optimization to minimize the loss function.

Embedding Learning:

During training, adjust the word vectors to capture the semantic relationships between words based on their contexts.

Word Embedding Generation:

Once training is complete, the learned word vectors serve as word embeddings.

These embeddings encode semantic similarities between words, facilitating various NLP tasks such as sentiment analysis, named entity recognition, and text summarization.

Through this process, Word2vec learns distributed representations of words that capture semantic relationships based on their contextual usage in the training data, thereby enhancing the performance of NLP applications.