150 COMPREHENSIVE NLP QUESTIONS AND ANSWERS

Comprehensive NLP Questions and Answers with Practical Python Code Examples

1. **Question:** What is Natural Language Processing (NLP)?
   Answer: NLP is a field of AI that enables computers to understand, interpret, and generate human language, both written and spoken, to facilitate meaningful interaction between humans and machines.

2. **Question:** What are common applications of NLP?
   Answer: Applications include text classification, sentiment analysis, machine translation, speech recognition, chatbots, summarization, and named entity recognition.

3. **Question:** What is tokenization in NLP?
   Answer: Tokenization is the process of splitting text into smaller units like words or sentences, which are easier for computers to analyze.

4. **Question:** How do you tokenize a sentence using Python's NLTK?
   Answer: Using NLTK's word_tokenize function:

   ```python
   from nltk.tokenize import word_tokenize
   tokens = word_tokenize("NLP is fun!")
   print(tokens)
   ```

5. **Question:** What are stopwords, and why are they removed?
   Answer: Stopwords are common filler words (e.g., "the", "is") that carry little semantic meaning. Removing them improves processing efficiency and sometimes model accuracy.

6. **Question:** What is lemmatization?
   Answer: Lemmatization reduces words to their base or dictionary form, helping to normalize text for better analysis (e.g., "running" becomes "run").

7. **Question:** Explain Bag of Words (BoW) model in NLP.
   Answer: BoW represents text by counting the frequency of words regardless of order, used for text classification and information retrieval.

8. **Question:** What is TF-IDF?
   Answer: Term Frequency-Inverse Document Frequency weights words based on how common they are in a document versus across many documents, highlighting important terms.

9. **Question:** What is Named Entity Recognition (NER)?
   Answer: NER identifies and classifies named entities like persons, locations, and organizations within text.

10. **Question:** How do you perform sentiment analysis using TextBlob?
    Answer:

    python

    ```
    from textblob import TextBlob
    text = "I love NLP!"
    blob = TextBlob(text)
    print(blob.sentiment.polarity)
    ```

    Sentiment polarity ranges from -1 (negative) to 1 (positive).

1) **Question:** What is the difference between stemming and lemmatization in NLP?
   Answer: Stemming cuts words to their root form often crudely (e.g., "running" to "run"), while lemmatization uses vocabulary and grammar to get the base form (e.g., "better" to "good").

2) **Question:** How can you perform stemming using NLTK's PorterStemmer?
   Answer:

   ```
   from nltk.stem import PorterStemmer
   ps = PorterStemmer()
   print(ps.stem("running"))
   ```

13. **Question:** What is part-of-speech (POS) tagging?
    Answer: POS tagging assigns grammatical categories (noun, verb, adjective, etc.) to each word in a sentence, aiding syntactic analysis.

14. **Question:** How to do POS tagging in NLTK?
    Answer:

    ```
    import nltk
    nltk.download('averaged_perceptron_tagger')
    sentence = "NLP helps computers understand language"
    tokens = nltk.word_tokenize(sentence)
    pos_tags = nltk.pos_tag(tokens)
    ```

```
print(pos_tags)
```

15. **Question:** What are word embeddings, and why are they needed?
    Answer: Word embeddings are dense vector representations of words capturing semantic meaning, allowing models to understand relationships beyond simple counts.

16. **Question:** Name two popular pre-trained word embedding models.
    Answer: Word2Vec and GloVe are popular embeddings trained on large corpora encoding semantic similarity in vector space.

17. **Question:** Describe the Bag of Words model's limitation.
    Answer: BoW ignores word order and context, which can cause loss of meaning, for example in distinguishing "not good" from "good".

18. **Question:** What is n-gram modeling in NLP?
    Answer: N-grams are sequences of 'n' words used to capture context and word order; bigrams are 2-word sequences, trigrams are 3-word sequences.

19. **Question:** How do you generate bigrams from a sentence using NLTK?
    Answer:

    ```
    from nltk import bigrams

    sentence = "NLP is interesting"

    tokens = nltk.word_tokenize(sentence)

    print(list(bigrams(tokens)))
    ```

20. **Question:** What is sentiment analysis used for?
    Answer: Sentiment analysis detects positive, negative, or neutral opinions in text, useful in social media monitoring, product reviews, and customer feedback.

21. **Question:** What is Named Entity Recognition (NER) and why is it important?
    Answer: NER identifies and classifies entities like names, locations, organizations in text, enabling automated information extraction for applications like search and knowledge graphs.

22. **Question:** How to perform NER using SpaCy in Python?
    Answer:

    ```
    import spacy

    nlp = spacy.load("en_core_web_sm")

    doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
    ```

```
for ent in doc.ents:
    print(ent.text, ent.label_)
```

23. **Question:** What is language modeling in NLP?
    Answer: Language modeling predicts the next word or sequence in text, crucial for tasks like text generation and speech recognition.

24. **Question:** Explain the difference between statistical and neural language models.
    Answer: Statistical models use probabilities from word counts; neural models use neural networks learning distributed word representations for better context handling.

25. **Question:** What role does the corpus play in NLP?
    Answer: A corpus is a large, structured set of text data used to train or evaluate NLP models.

26. **Question:** What is text classification? Give examples.
    Answer: Text classification assigns predefined categories to text data, such as spam detection, topic labeling, or sentiment tagging.

27. **Question:** How can you vectorize text data for machine learning models?
    Answer: Common methods include Bag of Words, TF-IDF, and word embeddings like Word2Vec or GloVe.

28. **Question:** What is the significance of stopword removal?
    Answer: Removing stopwords reduces noise, speeds up processing, and sometimes improves model performance by focusing on meaningful words.

29. **Question:** What is the main purpose of POS tagging?
    Answer: It helps understand sentence structure, enabling better parsing, information extraction, and semantic analysis.

30. **Question:** What is the difference between supervised and unsupervised NLP?
    Answer: Supervised NLP uses labeled data for tasks like classification; unsupervised NLP finds patterns or clusters without labels, e.g., topic modeling.

31. **Question:** What is sentiment polarity in sentiment analysis?
    Answer: Sentiment polarity quantifies the positivity or negativity of text, usually ranging from -1 (negative) to +1 (positive).

32. **Question:** How does a Bag of Words model represent text data?
    Answer: It treats text as an unordered collection of words, counting frequencies without considering grammar or order.

33. **Question:** What is the purpose of word embeddings like Word2Vec?
Answer: They convert words into dense vectors capturing semantic meaning and relationships between words based on context.

34. **Question:** What is one-hot encoding in NLP?
Answer: One-hot encoding represents each word as a binary vector with a 1 at the word's index and 0 elsewhere, useful for categorical data representation.

35. **Question:** What library in Python is commonly used for building and training NLP deep learning models?
Answer: TensorFlow and PyTorch are popular for deep learning-based NLP models.

36. **Question:** Define language translation in NLP.
Answer: Language translation automatically converts text or speech from one language to another using NLP techniques.

37. **Question:** What is a confusion matrix, and how is it relevant to text classification?
Answer: It is a table showing actual vs predicted labels, helping evaluate classifier performance on text data.

38. **Question:** Explain the term 'stopword list' in NLP.
Answer: It's a predefined set of words considered irrelevant for analysis and commonly removed during preprocessing.

39. **Question:** What is the key challenge of ambiguity in NLP?
Answer: Words or sentences can have multiple meanings depending on context, making interpretation challenging.

40. **Question:** Can you name two common tokenization techniques besides word tokenization?
Answer: Subword tokenization (e.g., Byte Pair Encoding) and sentence tokenization.

41. **Question:** What is word tokenization and why is it important? Show how to tokenize a sentence using NLTK.
Answer: Word tokenization splits text into individual words or tokens needed for further analysis. It's fundamental for almost every NLP task.

```
from nltk.tokenize import word_tokenize

sentence = "Tokenization splits text into words."

tokens = word_tokenize(sentence)

print(tokens)
```

Output will be a list of words: ['Tokenization', 'splits', 'text', 'into', 'words', '.'].

42. **Question:** Explain sentence tokenization and provide an example using NLTK.
Answer: Sentence tokenization splits paragraphs into sentences, useful for tasks like summarization or translation.

```
from nltk.tokenize import sent_tokenize

paragraph = "NLP is exciting. It uses computers to understand text."

sentences = sent_tokenize(paragraph)

print(sentences)
```

This outputs: ['NLP is exciting.', 'It uses computers to understand text.'].

43. **Question:** What are n-grams? Demonstrate bigrams extraction with code.
Answer: N-grams are sequences of 'n' words that help capture context. Bigrams are two consecutive words.

```
from nltk import bigrams

tokens = ['natural', 'language', 'processing', 'is', 'fun']

bigrams_list = list(bigrams(tokens))

print(bigrams_list)
```

Output: [('natural', 'language'), ('language', 'processing'), ('processing', 'is'), ('is', 'fun')].

44. **Question:** How can lemmatization improve text normalization?
Answer: Unlike stemming, lemmatization returns dictionary forms which can be more accurate for language understanding.

```
import spacy

nlp = spacy.load('en_core_web_sm')

doc = nlp("running ran runs")

lemmas = [token.lemma_ for token in doc]

print(lemmas)
```

45. **Question:** Define TF-IDF and show how to calculate it in sklearn.
Answer: TF-IDF scores how important a word is in a document relative to a corpus.

```
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = ['I love NLP', 'NLP is exciting']

vectorizer = TfidfVectorizer()
```

```
tfidf_matrix = vectorizer.fit_transform(corpus)

print(vectorizer.get_feature_names_out())

print(tfidf_matrix.toarray())
```

46. **Question:** What is part-of-speech tagging? Provide code example with NLTK.
    Answer: POS tagging labels words as nouns, verbs, adjectives, etc.

```
import nltk

nltk.download('averaged_perceptron_tagger')

sentence = "NLP helps computers understand language"

tokens = nltk.word_tokenize(sentence)

pos_tags = nltk.pos_tag(tokens)

print(pos_tags)
```

47. **Question:** Explain Named Entity Recognition (NER) and extract entities using SpaCy.
    Answer: NER detects names, places, dates, etc., from text aiding in information extraction.

```
import spacy

nlp = spacy.load("en_core_web_sm")

doc = nlp("Google was founded in Mountain View in 1998")

for ent in doc.ents:

   print(ent.text, ent.label_)
```

48. **Question:** Describe the Bag of Words model's pros and cons.
    Answer: BoW is simple and effective for many tasks but ignores word order and semantics, which can limit understanding.

49. **Question:** What is word embedding? Show how to load pre-trained Word2Vec in Gensim.
    Answer: Word embeddings represent words as dense vectors capturing semantic similarity.

```
from gensim.models import KeyedVectors

model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)

print(model['computer'])
```

50. **Question:** How is sentiment analysis done using TextBlob? Include example.
Answer: Sentiment analysis classifies text as positive/negative. TextBlob returns polarity scores.

```
from textblob import TextBlob

text = "I love learning NLP!"

blob = TextBlob(text)

print(blob.sentiment.polarity)
```

51. **Question:** What is language modeling, and why is it central to many NLP applications?
Answer: Language modeling predicts the likelihood of a sequence of words, helping machines generate text, autocomplete input, or understand intent. Modern applications include chatbots, automatic captioning, and translation tools.

52. **Question:** How do you use NLTK to generate bigrams from a list of tokens? Why are bigrams useful?
Answer: Bigrams model relationships between consecutive words, capturing local context that single words miss—used in tasks like text classification and spell-checking.

python

```
from nltk import bigrams

tokens = ["Natural", "Language", "Processing", "rocks"]

print(list(bigrams(tokens)))

Output: [('Natural', 'Language'), ('Language', 'Processing'), ('Processing', 'rocks')]
```

53. **Question:** Explain document similarity. How can you compute similarity between texts using TF-IDF vectors in sklearn?
Answer: Document similarity quantifies how closely texts are related, used in clustering, search engines, and plagiarism detection.

python

```
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

docs = ["NLP is fun", "I enjoy learning NLP"]

vec = TfidfVectorizer()

tfidf = vec.fit_transform(docs)
```

```
sim = cosine_similarity(tfidf[0:1], tfidf)

print(sim)
```

Cosine similarity gives a score between 0 and 1 indicating textual relatedness.

54. **Question:** What are the main challenges in text preprocessing for NLP?
Answer: Issues include inconsistent capitalization, misspellings, slang, emojis, and handling non-text data like numbers or special symbols. Proper cleaning is crucial for effective NLP modeling and interpreting real-world data.

55. **Question:** Discuss term frequency (TF) and inverse document frequency (IDF).
Answer: TF counts how often a word appears in a document; IDF scores how rare it is across all documents. Their product, TF-IDF, increases the weight of important words while reducing the impact of common words, aiding documents' differentiation.

56. **Question:** How can SpaCy be used for dependency parsing? Why is this useful?
Answer: Dependency parsing in SpaCy identifies grammatical relations among words, important for extracting subject-verb-object triples and understanding meaning.

```
python

import spacy

nlp = spacy.load("en_core_web_sm")

doc = nlp("The cat sat on the mat.")

for token in doc:

   print(token.text, token.dep_, token.head.text)
```

57. **Question:** What is topic modeling? Name one algorithm used for it.
Answer: Topic modeling uncovers hidden thematic structures in a text collection. Latent Dirichlet Allocation (LDA) is a popular unsupervised algorithm that assigns topic probabilities to documents.

58. **Question:** How do word embeddings like GloVe differ from Bag of Words?
Answer: Bag of Words represents words with sparse, high-dimensional vectors based on counts, lacking semantics. GloVe embeds words into low-dimensional dense vectors encoding meaning and context relationships learned from global co-occurrence statistics.

59. **Question:** What is attention mechanism in modern NLP models?
Answer: Attention allows models (like Transformers) to focus on relevant parts of the

input sequence when processing or generating output, improving performance in tasks such as translation and summarization.

60. **Question:** Give an example of text classification using scikit-learn's logistic regression on simple data.
Answer: Text classification predicts labels for text; logistic regression can be used with features like TF-IDF vectors.

```python
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression

text = ["spam offer", "important update", "buy now", "meeting agenda"]

labels = [1, 0, 1, 0]

vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(text)

model = LogisticRegression()

model.fit(X, labels)

print(model.predict(vectorizer.transform(["urgent buy offer"])))
```

61. **Question:** What is the difference between supervised and unsupervised learning in NLP?
Answer: Supervised learning uses labeled data (e.g., sentiment-labeled reviews) to train models, while unsupervised learning finds patterns in unlabeled data, like clustering similar documents or topic modeling.

62. **Question:** How can you use NLTK to perform POS tagging on a sentence?
Answer: NLTK offers a built-in tagger with pretrained models:

```python
import nltk

nltk.download('averaged_perceptron_tagger')

sentence = "I am learning NLP"

tokens = nltk.word_tokenize(sentence)

pos_tags = nltk.pos_tag(tokens)

print(pos_tags)
```

Output example: [('I', 'PRP'), ('am', 'VBP'), ('learning', 'VBG'), ('NLP', 'NNP')]

63. **Question:** Describe the role of Transformers in NLP.
    Answer: Transformers use self-attention mechanisms to process input sequences in parallel, excelling in tasks like translation, summarization, and language generation.

64. **Question:** What is BERT, and why is it influential in NLP?
    Answer: BERT (Bidirectional Encoder Representations from Transformers) captures context from both past and future words, enabling state-of-the-art understanding in many NLP tasks.

65. **Question:** How do you perform sentiment analysis using VADER in NLTK?
    Answer: VADER is a lexicon and rule-based sentiment analyzer, optimized for social media texts.

    python

    ```
    from nltk.sentiment.vader import SentimentIntensityAnalyzer

    sid = SentimentIntensityAnalyzer()

    text = "This is an awesome NLP tutorial!"

    print(sid.polarity_scores(text))
    ```

66. **Question:** What is text summarization? Describe extractive vs abstractive summarization.
    Answer: Text summarization condenses long texts. Extractive selects important sentences; abstractive generates new sentences preserving meaning.

67. **Question:** How can you perform text vectorization using Keras Tokenizer?
    Answer: Keras Tokenizer converts text to sequences of integers, ready for deep learning.

    python

    ```
    from keras.preprocessing.text import Tokenizer

    texts = ["Deep learning is fun", "NLP with deep learning"]

    tokenizer = Tokenizer(num_words=100)

    tokenizer.fit_on_texts(texts)

    sequences = tokenizer.texts_to_sequences(texts)

    print(sequences)
    ```

68. **Question:** What are stopwords, and should they always be removed?
Answer: Stopwords are common words usually filtered out to reduce noise. However, in some tasks like sentiment analysis or phrase extraction, retaining stopwords might improve results.

69. **Question:** Explain word sense disambiguation.
Answer: It’s the process of resolving which meaning of a word is used in context, critical for accurate understanding and translation.

70. **Question:** How do you handle out-of-vocabulary (OOV) words in word embedding models?
Answer: Techniques include using special OOV tokens, subword embeddings (e.g., FastText), or retraining embeddings to include new words.

71. **Question:** What is the Transformer architecture, and what problem does it solve in NLP?
Answer: Transformers use self-attention mechanisms that process entire sequences simultaneously instead of sequentially, solving issues of long-range dependencies and parallelizing training efficiently.

72. **Question:** How do you load a pre-trained BERT model and tokenizer using Hugging Face Transformers?
Answer:

```
python

from transformers import BertTokenizer, BertModel

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

model = BertModel.from_pretrained('bert-base-uncased')

inputs = tokenizer("Hello, NLP!", return_tensors="pt")

outputs = model(**inputs)

print(outputs.last_hidden_state.shape)
```

73. **Question:** Explain sequence-to-sequence (seq2seq) models and their application in NLP.
Answer: Seq2seq models convert input sequences to output sequences, used in machine translation, summarization, and chatbots. They consist of an encoder to process input and a decoder to generate output.

74. **Question:** What are attention weights in the Transformer model?
Answer: Attention weights indicate how much focus the model gives to each word when encoding/decoding, allowing dynamic contextual understanding.

75. **Question:** How can you fine-tune a pre-trained Transformer model for text classification?
Answer: Load the pre-trained model, add a classification head, prepare labeled data, and use gradient descent to update weights on your task.

76. **Question:** How do you implement word embeddings using GloVe vectors in Python?
Answer: Load pre-trained GloVe embeddings and map words to vectors:

```
python

import numpy as np

embedding_index = {}

with open('glove.6B.100d.txt', 'r') as f:

    for line in f:

        values = line.split()

        word = values[0]

        vector = np.asarray(values[1:], dtype='float32')

        embedding_index[word] = vector

print(embedding_index['computer'][:5])
```

77. **Question:** Describe transfer learning in NLP and its advantages.
Answer: Transfer learning reuses models trained on large datasets to improve performance on specific, smaller NLP tasks, reducing data and compute needs.

78. **Question:** What is the difference between extractive and abstractive summarization? Give a code example of extractive summarization using Python’s Gensim.
Answer: Extractive picks key sentences; abstractive generates new sentences. For extractive summarization:

```
python

from gensim.summarization import summarize

text = "Long article text here..."

print(summarize(text, word_count=50))
```

79. **Question:** Define text generation and an example use case.
Answer: Text generation predicts likely next words to create new text, used in chatbots, story writing, and auto-completion.

80. **Question:** How does tokenization affect handling out-of-vocabulary words in NLP models?
Answer: Subword tokenization splits unknown words into known smaller units, improving coverage and reducing OOV effects.

81. **Question:** What is the role of positional encoding in Transformer models?
Answer: Since Transformers do not process sequences sequentially, positional encoding injects information about word order into input embeddings, allowing models to understand position and sequence.

82. **Question:** How do attention mechanisms improve machine translation?
Answer: Attention helps the model focus on relevant source words when generating each target word, improving translation quality by capturing alignment between languages.

83. **Question:** Describe transfer learning with pre-trained language models like BERT and GPT.
Answer: These models are trained on massive corpora and can be fine-tuned on specific tasks, requiring less labeled data while achieving high accuracy.

84. **Question:** Demonstrate how to use Hugging Face's pipeline for sentiment analysis.
Answer:

python

```
from transformers import pipeline

classifier = pipeline('sentiment-analysis')

result = classifier("I love learning NLP with examples!")

print(result)
```

85. **Question:** What is zero-shot learning in NLP?
Answer: Zero-shot learning enables models to predict tasks they were never explicitly trained on by leveraging learned knowledge, like classifying unseen categories using prompt-based approaches.

86. **Question:** What challenges exist in multi-lingual NLP?
Answer: Issues include diverse languages/scripts, scarce labeled data, differing grammar, and cultural nuances affecting interpretation.

87. **Question:** What is the difference between extractive and abstractive question answering?
Answer: Extractive answers are copied from the source text; abstractive answers are generated, often more natural and concise, but harder to produce.

88. **Question:** How do you implement text preprocessing steps like lowercasing, punctuation removal, and stopword removal in Python?
Answer:

```
python

import string

from nltk.corpus import stopwords

text = "Hello, world! This is a test."

text = text.lower()

text = text.translate(str.maketrans('', '', string.punctuation))

words = text.split()

stop_words = set(stopwords.words('english'))

filtered = [w for w in words if w not in stop_words]

print(filtered)
```

89. **Question:** Explain the use of beam search in sequence generation tasks.
Answer: Beam search keeps multiple hypotheses at each decoding step, selecting the most likely sequences overall instead of greedy word-by-word selection, improving output quality.

90. **Question:** What is contextual word embedding? How is it different from static embedding?
Answer: Contextual embeddings (e.g., from BERT) produce word vectors dependent on surrounding text, capturing word sense dynamically; static embeddings assign a fixed vector regardless of context.

91. **Question:** What is text normalization in NLP, and which common steps does it include?
Answer: Text normalization prepares raw text for analysis by converting to lowercase, removing punctuation, expanding contractions, and eliminating noise like extra whitespace or special characters.

92. **Question:** How can you remove punctuation from a text string in Python?
Answer:

```
python

import string

text = "Hello! How are you?"

clean_text = text.translate(str.maketrans('', '', string.punctuation))

print(clean_text)  # Output: Hello How are you
```

93. **Question:** Explain the difference between unigram, bigram, and trigram models.
Answer: Unigram models look at single words independently, bigrams consider pairs sequentially, and trigrams capture triplets, improving context capture but increasing complexity.

94. **Question:** What is a stop word list, and can you give an example of removing stopwords using NLTK?
Answer: A stop word list contains common words that are often removed to reduce noise. Example:

```
python

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

sentence = "This is a simple example"

words = word_tokenize(sentence)

filtered_words = [w for w in words if w.lower() not in stop_words]

print(filtered_words)  # Output: ['simple', 'example']
```

95. **Question:** How does the cosine similarity metric work for comparing text data?
Answer: Cosine similarity measures the cosine of the angle between two vectors (e.g., TF-IDF vectors), quantifying text similarity from 0 (no similarity) to 1 (identical).

96. **Question:** What are the benefits of using pretrained NLP models versus training from scratch?
Answer: Pretrained models provide better accuracy with less data and training time, leverage large datasets for transfer learning, and reduce compute resources required.

97. **Question:** How do transformers differ from RNN or LSTM models?
Answer: Transformers use self-attention and parallel processing, handling long-range dependencies better and training faster, while RNNs/LSTMs process sequentially with potential vanishing gradient problems.

98. **Question:** What is an embedding layer in neural networks for NLP?
Answer: An embedding layer maps words or tokens into dense vectors learned during training, capturing semantic meanings for downstream processing.

99. **Question:** Demonstrate creating an embedding layer in Keras.
Answer:

```
python

from keras.models import Sequential

from keras.layers import Embedding

model = Sequential()

model.add(Embedding(input_dim=1000, output_dim=64, input_length=10))

model.summary()
```

100. **Question:** How can attention visualize which words a model focuses on during prediction?
Answer: Attention weights can be extracted and visualized as heatmaps to interpret and debug model behavior by showing word importance for specific outputs.

101. **Question:** What is the purpose of the embedding dimension in word embeddings, and how is it chosen?
Answer: The embedding dimension defines the size of the vector representing each word. Higher dimensions can capture more semantic nuances but risk overfitting and require more computational power. Common sizes are 50, 100, 200, or 300 dimensions. The choice depends on dataset size and task complexity.

102. **Question:** How can you implement a basic sentiment analysis pipeline using Scikit-learn?
Answer: Steps include text preprocessing, vectorization (e.g., TF-IDF), model training with labeled sentiment data, and evaluation.

```
python

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression
```

```
texts = ["I love this product", "This is bad"]
labels = [1, 0]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)
model = LogisticRegression()
model.fit(X, labels)
print(model.predict(vectorizer.transform(["I hate this"])))
```

103. **Question:** What is the effect of using subword tokenization techniques like Byte Pair Encoding (BPE)?
Answer: BPE splits rare or unknown words into smaller, frequent subwords, reducing out-of-vocabulary cases and improving handling of morphologically rich languages.

104. **Question:** How do Named Entity Recognition (NER) models distinguish between different entity types?
Answer: NER models learn patterns and contexts of words to classify entities into predefined categories like Person, Location, Organization, Date, etc., often using sequence labeling techniques like Conditional Random Fields or transformers.

105. **Question:** Describe how attention matrices can be interpreted for explainability in transformer models.
Answer: Each attention head outputs weights indicating word-to-word influence. Visualizing these matrices reveals model focus areas and reasoning paths of predictions, helping debug and trust model outputs.

106. **Question:** What is a language model perplexity?
Answer: Perplexity measures how well a probability model predicts a sample; lower perplexity indicates better predictive performance in NLP language models.

107. **Question:** How to handle class imbalance in text classification tasks?
Answer: Techniques include using weighted loss functions, oversampling minority classes, undersampling majority classes, or generating synthetic data via methods like SMOTE.

108. **Question:** What are the challenges with sarcasm detection in sentiment analysis?
Answer: Sarcasm flips literal sentiment and relies on context and tone, making it difficult for models to detect with standard lexical methods, necessitating deeper contextual and multimodal analysis.

109. **Question:** Explain how encoder-decoder architecture works in sequence-to-sequence NLP tasks.
Answer: The encoder processes input sequence into a context vector; the decoder uses this to generate output sequence, often augmented with attention mechanisms for better alignment.

110. **Question:** Show an example of using Hugging Face's pipeline for question answering.
Answer:

python

```
from transformers import pipeline

qa_pipeline = pipeline("question-answering")

context = "The Eiffel Tower is located in Paris."

question = "Where is the Eiffel Tower located?"

result = qa_pipeline(question=question, context=context)

print(result)
```

111. **Question:** What is the role of Beam Search in sequence prediction tasks?
Answer: Beam Search maintains multiple candidate sequences at each step, balancing exploration and exploitation to find more probable sequences, improving output quality in tasks like translation.

112. **Question:** How can you implement coreference resolution in NLP?
Answer: Coreference resolution links pronouns and entities to their real-world references, which can be implemented using libraries likeSpaCy with neural coref models or AllenNLP's coreference resolver:

python

```
import spacy

import neuralcoref

nlp = spacy.load('en_core_web_sm')

coref = neuralcoref.NeuralCoref(nlp)

nlp.add_pipe(coref)

doc = nlp("My sister has a dog. She loves him.")
```

```
print(doc._.coref_clusters)
```

113. **Question:** Explain the concept of zero-shot and few-shot learning in NLP models.
Answer: Zero-shot learning predicts on unseen categories without training data by leveraging large pre-trained models and context prompts; few-shot learning adapts models with a small number of examples.

114. **Question:** How does transfer learning improve NLP model efficiency?
Answer: It allows models pretrained on large datasets (like BERT, GPT) to be fine-tuned on specific tasks, drastically reducing training time and data needs while improving performance.

115. **Question:** Describe different strategies for data augmentation in NLP.
Answer: Techniques include synonym replacement, back-translation, random insertion/deletion, and contextual word masking to create diverse training examples and improve model robustness.

116. **Question:** What is the significance of the Masked Language Modeling (MLM) objective?
Answer: MLM trains models to predict masked tokens within a sequence, enabling deep contextual understanding, as used in BERT.

117. **Question:** Demonstrate the use of Hugging Face's transformers for text classification with an example.
Answer:

python

```
from transformers import pipeline

classifier = pipeline('text-classification', model='distilbert-base-uncased-finetuned-sst-2-english')

result = classifier("This movie is fantastic!")

print(result)
```

118. **Question:** How can models like GPT be used for story generation?
Answer: GPT models generate coherent text by predicting subsequent tokens given an initial prompt, suitable for story writing, dialogue simulation, or creative content.

119. **Question:** Explain the concept of token importance and how it can be visualized in transformer models.
Answer: Token importance refers to how much each input token contributes to the

model’s prediction, visualized via attention heatmaps or gradient-based attribution methods.

120. **Question:** What are some common evaluation metrics for NLP tasks like translation, summarization, or question answering?
Answer: Metrics include BLEU, ROUGE, METEOR for translation and summarization; F1 score, Exact Match, and EM scores for question answering.

121. **Question:** What is the difference between rule-based and machine learning-based NLP?
Answer: Rule-based uses hand-crafted linguistic rules for processing, while ML-based relies on statistical models learning from data, offering scalability and adaptability.

122. **Question:** How would you explain word embeddings to a non-technical person?
Answer: Word embeddings are like mapping words to coordinates so that similar words are placed close together in a space computers can understand and use.

123. **Question:** What are common methods for text vectorization besides TF-IDF and BoW?
Answer: Word embeddings (Word2Vec, GloVe), subword embeddings (FastText), and transformer-based embeddings.

124. **Question:** How can you handle negation in sentiment analysis?
Answer: By incorporating context windows around negation cues, using dependency parsing or modifying feature extraction to invert sentiment polarity.

125. **Question:** What is a language model's perplexity, and why is it important?
Answer: Perplexity measures how well the model predicts a sample; low perplexity indicates better performance.

126. **Question:** Show Python code to perform POS tagging using SpaCy.
Answer:

```
python

import spacy

nlp = spacy.load('en_core_web_sm')

doc = nlp("I am learning NLP.")

for token in doc:
    print(token.text, token.pos_)
```

127. **Question:** Explain Transformers and self-attention in simple terms.
Answer: Transformers use self-attention to weigh importance of each word relative to others, helping capture context globally.

128. **Question:** How do you fine-tune a transformer for a custom classification task?
Answer: Use pre-trained models with added classification heads, fine-tune on labeled data using libraries like Hugging Face Transformers.

129. **Question:** Describe the role of subword tokenization and methods like Byte-Pair Encoding.
Answer: Subword tokenization breaks words into meaningful pieces, reducing vocabulary size and handling unknown words efficiently.

130. **Question:** How can you use NLTK to remove stopwords from a text string?
Answer:

python

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


text = "This is a sample sentence."
stop_words = set(stopwords.words('english'))
words = word_tokenize(text)
filtered = [w for w in words if w.lower() not in stop_words]
print(filtered)
```

131. **Question:** What are common NLP evaluation metrics for classification?
Answer: Accuracy, Precision, Recall, F1-score, Confusion Matrix.

132. **Question:** How is tokenization different from parsing?
Answer: Tokenization divides text into tokens; parsing analyzes grammatical structure and relationships among tokens.

133. **Question:** What is coreference resolution and why is it important?
Answer: It links pronouns and references to entities for proper understanding of texts, essential for tasks like summarization and question answering.

134. **Question:** Explain the curse of dimensionality in NLP.
Answer: High-dimensional vector space in methods like BoW leads to sparsity and computational inefficiency, tackled by embeddings.

135. **Question:** How can you create word embeddings using Gensim Word2Vec?
Answer:

```
python

from gensim.models import Word2Vec

sentences = [["I", "love", "NLP"], ["NLP", "is", "fun"]]

model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

print(model.wv['NLP'])
```

136. **Question:** What is the significance of Named Entity Recognition (NER) in NLP applications?
Answer: NER extracts critical entities from text to enable structured understanding for search, question answering, and information extraction.

137. **Question:** How is Natural Language Generation (NLG) used today?
Answer: NLG powers chatbots, content creation, summarization, and report generation by converting data into readable text.

138. **Question:** What is an embedding layer in deep learning NLP architectures?
Answer: The embedding layer translates tokens into dense vectors learned during training, easing pattern learning for models.

139. **Question:** How do you evaluate an NLP model designed for machine translation?
Answer: Using metrics like BLEU, METEOR, ROUGE to compare machine-generated text with reference translations.

140. **Question:** Show example code to classify text using Hugging Face Transformers pipeline.
Answer:

```
python

from transformers import pipeline

classifier = pipeline('text-classification')

result = classifier("The movie was fantastic!")

print(result)
```

142. **Question:** How does a pre-trained language model like GPT generate text? What tasks benefit from this?
Answer: GPT (Generative Pretrained Transformer) uses a transformer decoder trained on large text corpora using autoregressive language modeling. Given a prompt, it predicts the next token step-by-step, generating coherent text continuations. GPT's capabilities power tasks such as story generation, text completion, conversational AI, and code synthesis. Fine-tuning GPT on domain-specific data further adapts it to specialized applications.

143. **Question:** What is the cosine similarity measure? How is it used in NLP?
Answer: Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, providing a similarity score between -1 and 1. In NLP, it's commonly used to compare document or sentence vectors, such as TF-IDF vectors or embeddings, to find how similar two texts are in terms of content. High cosine similarity indicates near-identical meaning or context.

144. **Question:** Explain the process of fine-tuning BERT for a text classification task.
Answer: Fine-tuning BERT involves initializing it with pretrained weights, adding a classification head (usually a dense layer), then training on labeled data for the target task. This lets the model adapt the contextual embeddings learned in pretraining to the specific task with relatively few labeled examples. Training uses small learning rates and typically requires careful regularization to prevent overfitting.

145. **Question:** How can transformers handle input sequences longer than their positional encoding limit?
Answer: Strategies include chunking long texts into segments, using sliding windows, and applying models designed for long sequences like Longformer or Reformer. Additionally, relative positional embeddings or sparse attention mechanisms help scale transformers efficiently to longer inputs.

146. **Question:** Describe the use of Named Entity Recognition (NER) in healthcare NLP.
Answer: NER in healthcare extracts critical medical entities (like diseases, drugs, treatments) from clinical notes, aiding in patient record structuring, decision support, and research. It helps identify relationships and monitor conditions by transforming unstructured text into actionable data.

147. **Question:** What are the advantages and disadvantages of using LSTM networks in NLP?
Answer: LSTMs effectively capture temporal dependencies and work well on

sequential data with long-range dependencies, overcoming vanishing gradients in standard RNNs. However, they process sequentially, making training slower and less parallelizable than transformers. They also may struggle with extremely long contexts and require substantial tuning.

148. **Question:** How does subword embedding (e.g., FastText) improve handling of out-of-vocabulary (OOV) words?
Answer: Subword embeddings learn vectors for character n-grams, allowing representation of unseen words as compositions of known subwords. This reduces OOV issues and better captures morphological variations and rare words, improving model robustness.

149. **Question:** What are some common techniques to mitigate bias in NLP models?
Answer: Mitigation approaches include data balancing, fairness-aware training objectives, debiasing embeddings, transparency in model decisions, and ongoing evaluation on diverse demographic datasets. Approaches like adversarial training and counterfactual data augmentation are used to reduce model biases.

150. **Question:** Show example code for performing sentiment analysis using Hugging Face transformers pipeline.
Answer:

```
from transformers import pipeline

classifier = pipeline('sentiment-analysis')

text = "The movie was absolutely wonderful and captivating."

result = classifier(text)

print(result)
```