Comprehensive Natural Language Processing Interview Preparation Guide

Prepared for Deep Learning Interview

June 21, 2025

Contents

1 Introduction

This guide is designed for candidates preparing for a deep learning interview with a focus on Natural Language Processing (NLP). It assumes basic neural network knowledge from Kaggle competitions and provides a detailed overview of NLP concepts, including foundational neural network components (optimizers, callbacks) and advanced architectures like RNNs, LSTMs, GRUs, and Transformers. Each section covers the evolution, mathematical foundations, applications, and when to use each technique.

2 Foundational Neural Network Concepts

Neural Networks form the core of NLP models. This section covers components relevant to NLP tasks.

2.1 Neural Network Architecture

A neural network consists of:

- **Input Layer**: Receives text data (e.g., word embeddings).
- **Hidden Layers**: Process sequences or features.
- **Output Layer**: Produces predictions (e.g., next word, sentiment).

Each neuron computes:

$$z = \sum(w_i \cdot x_i) + b, \quad a = \sigma(z)$$

where w_i are weights, x_i are inputs, b is the bias, and σ is the activation function.

2.2 Activation Functions

- **Tanh** (1980s): $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, used in RNNs/LSTMs.
- **ReLU** (2010): $f(z) = \max(0, z)$, used in feedforward layers.
- **GELU** (2016): $f(z) = z \cdot \Phi(z)$, used in Transformers.

When to Use: Tanh for RNNs; GELU for Transformers.

2.3 Loss Functions

- **Cross-Entropy Loss**: $-\sum y_i \log(\hat{y}_i)$, for classification (e.g., sentiment analysis).
- **Perplexity**: Measures language model quality, lower is better.
- **CTC Loss**: For sequence alignment (e.g., speech-to-text).

When to Use: Cross-Entropy for classification; Perplexity for language modeling.

2.4 Backpropagation

Backpropagation (1986) computes gradients for weight updates, extended to sequences via Backpropagation Through Time (BPTT) in NLP.

2.5 Optimizers

- **SGD with Momentum** (1988): Stable but slow for NLP.
- **RMSProp** (2012): Effective for RNNs due to non-stationary gradients.
- **Adam** (2014): Default for NLP, fast convergence.

- **AdamW** (2017): Improves generalization in Transformers.
- **LAMB** (2019): Optimized for large-batch training in Transformers.

When to Use: Adam for most NLP tasks; RMSProp for RNNs; LAMB for large-scale Transformers.

2.6 Callbacks

- **Early Stopping**: Prevents overfitting in long training.
- **Model Checkpoint**: Saves best model based on perplexity or accuracy.
- **Learning Rate Scheduler**: Adjusts learning rate (e.g., warmup for Transformers).
- **Gradient Clipping**: Prevents exploding gradients in RNNs.

When to Use: Early Stopping and Checkpoint for all models; Gradient Clipping for RNNs.

3 NLP Architectures

NLP focuses on tasks like text classification, translation, and generation. Key architectures:

3.1 Word Embeddings

Pre-neural NLP relied on bag-of-words, but embeddings map words to dense vectors:

- **Word2Vec** (2013): CBOW or Skip-gram, captures semantic similarity.
- **GloVe** (2014): Global word co-occurrence matrix factorization.
- **FastText** (2016): Subword embeddings, handles OOV words.

When to Use: Word2Vec/GloVe for static embeddings; FastText for morphologically rich languages.

3.2 Recurrent Neural Networks (RNNs)

Introduced in 1980s, RNNs process sequences:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b)$$

Issues: Vanishing/exploding gradients, slow due to sequential processing. **When to Use**: Simple tasks with short sequences.

3.3 Long Short-Term Memory (LSTM)

Introduced in 1997, LSTMs use gates to manage long-term dependencies:

- **Forget Gate**: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$.
- **Input Gate**: Decides what to store.
- **Output Gate**: Decides what to output.

Variants: Bidirectional LSTMs, Stacked LSTMs. **When to Use**: Tasks with long dependencies (e.g., machine translation).

3.4 Gated Recurrent Unit (GRU)

Introduced in 2014, GRUs simplify LSTMs with update and reset gates, faster training. **When to Use**: When LSTMs are too slow, similar performance.

3.5 Transformers

Introduced in 2017 ("Attention is All You Need"), Transformers use self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Components:

- **Multi-Head Attention**: Multiple attention mechanisms in parallel.
- **Positional Encoding**: Adds word order information.
- **Feedforward Networks**: Applied per token.
- **Layer Normalization**: Stabilizes training.

Advantages: Parallelizable, captures long-range dependencies. **Evolution**:

- **BERT** (2018): Bidirectional, pre-trained on masked language modeling.
- **GPT** (2018-2023): Autoregressive, for generation.
- **RoBERTa** (2019): Optimized BERT training.
- **T5** (2020): Text-to-text framework, versatile.
- **LLaMA** (2023): Efficient for research, not commercial.

When to Use: Transformers for most NLP tasks due to superior performance.

4 Practical Considerations in NLP

- **Pre-training**: Use pre-trained models (e.g., BERT) for transfer learning.
- **Tokenization**: WordPiece (BERT), BPE (GPT) for subword units.
- **Handling Long Sequences**: Truncate or use Longformer/Sparse Transformers.
- **Evaluation Metrics**:
  - Classification: F1-score, accuracy.
  - Translation: BLEU, ROUGE.
  - Generation: Perplexity, human evaluation.

5 Evolution Timeline

- 1980s: RNNs.
- 1997: LSTMs.
- 2013: Word2Vec.
- 2014: GRUs, GloVe.
- 2016: FastText.
- 2017: Transformers.
- 2018: BERT, GPT.
- 2019: RoBERTa.

- 2020: T5.
- 2023: LLaMA.

6 Interview Tips

- Explain attention as weighted context aggregation.
- Discuss trade-offs (e.g., RNN vs. Transformer for sequence length).
- Relate to Kaggle (e.g., fine-tuned BERT for text classification).
- Be ready to write pseudocode for self-attention or BPTT.

7 Conclusion

This guide provides a comprehensive understanding of NLP concepts for your interview. Focus on practical applications and clear explanations. Good luck!