Advanced Topics in Natural Language Processing: RNNs, LSTMs, GRUs and Word Embeddings

Minor in AI - IIT ROPAR

25th April, 2025

Contents

1 Introduction to Advanced NLP Concepts

Natural Language Processing (NLP) has evolved significantly with the introduction of sophisticated architectures for handling sequential data. This document extends our previous discussion by exploring Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) units, Gated Recurrent Units (GRUs), and advanced word embedding techniques. These frameworks collectively form the foundation for modern language processing capabilities.

2 Loss Minimization Revisited

Loss minimization serves as a unifying framework across various machine learning tasks. In the context of NLP, different modalities of input data (text, speech, etc.) can be generalized through this approach.

2.1 Mathematical Formulation

For a given task, we define:

- Input space $\mathcal{X}$ (e.g., text documents)
- Output space $\mathcal{Y}$ (e.g., sentiment classes)
- Model parameters θ
- Loss function $\mathcal{L}(\theta)$

The objective is to minimize:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta) = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

Algorithm 1 Stochastic Gradient Descent for NLP Models

```
Initialize parameters θ randomly
Set learning rate η
for epoch = 1 to num_epochs do
    for batch b in training data do
        Compute forward pass to get predictions
        Calculate loss ℒ(θ)
        Compute gradients ∇_θ ℒ(θ)
        Update parameters: θ ← θ − η∇_θ ℒ(θ)
    end for
end for
```

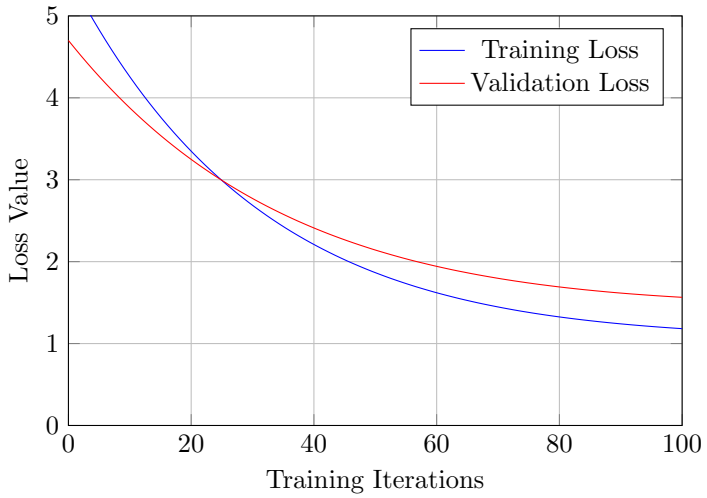

Figure 1: Loss Minimization Progress in a Typical NLP Task

2.2 Gradient-Based Optimization

Most NLP models utilize gradient-based methods for optimization:

3 Auto-Regressive Models

Auto-regressive models constitute a class of models that predict future values based on past observations. They are particularly relevant in language modeling, where the probability of the next word depends on preceding words.

3.1 Formal Definition

An auto-regressive model of order p is defined as:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon_t$$

where:

- x_t is the value at time t
- c is a constant
- ϕ_i are the model parameters
- ε_t is white noise

3.2 Language Modeling Application

In language modeling, the auto-regressive property manifests as:

$$P(w_t|w_1, w_2, \ldots, w_{t-1}) = f_\theta(w_1, w_2, \ldots, w_{t-1})$$

This conditional probability is the foundation of text generation systems.

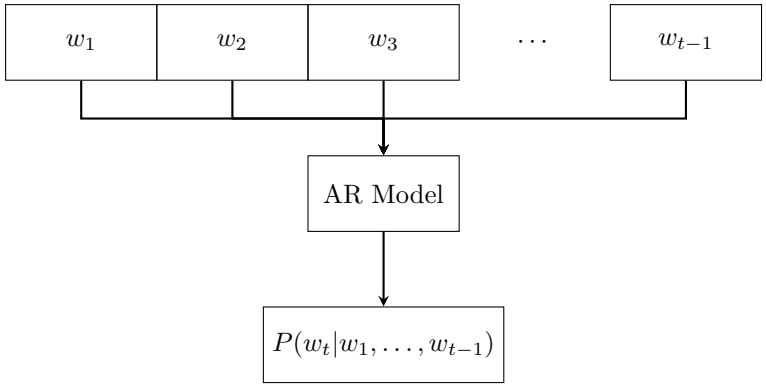

Figure 2: Auto-regressive Language Model Structure

4 Word Embedding Techniques

Word embeddings represent words as dense vectors in a continuous vector space, capturing semantic relationships between words.

4.1 Limitations of One-Hot Encoding

One-hot encoding represents each word as a sparse vector with a single '1' at the position corresponding to the word's index:

Limitations include:

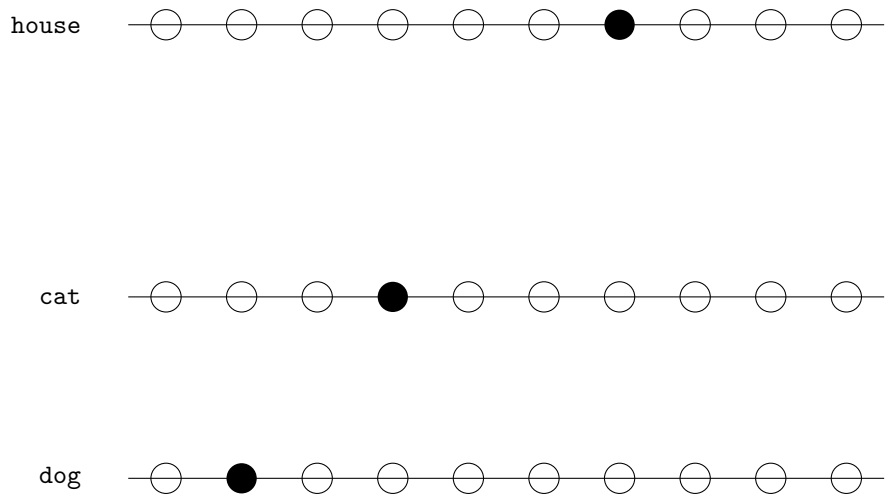

Figure 3: One-Hot Encoding Representation

- Curse of dimensionality (vocabulary size determines vector dimension)
- No semantic relationship capture
- Sparsity leading to computational inefficiency

4.2 Dense Vector Representations

Dense embeddings represent words in a lower-dimensional space:

4.3 Word2Vec

Word2Vec learns word embeddings through two primary architectures:

4.3.1 Continuous Bag of Words (CBOW)

CBOW predicts a target word from its surrounding context words:

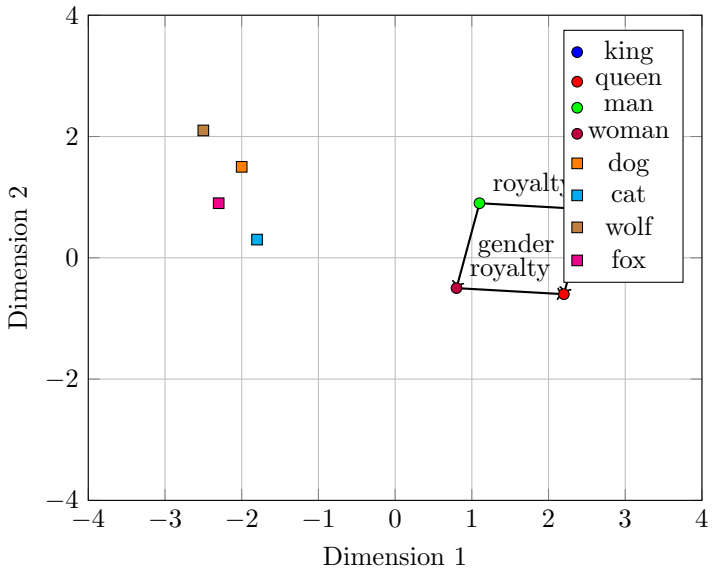

Figure 4: 2D Visualization of Word Embeddings

4.3.2 Skip-gram

Skip-gram predicts context words from a target word:

4.4 GloVe (Global Vectors)

GloVe combines global matrix factorization with local context window methods:

$$J = \sum_{i,j=1}^{|V|} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \tag{1}$$

where:

- X_{ij} is the co-occurrence frequency of words i and j
- $f(X_{ij})$ is a weighting function
- w_i and $\tilde{w}_j$ are word vectors
- b_i and $\tilde{b}_j$ are bias terms

5 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are specialized neural networks designed to process sequential data by maintaining internal memory.

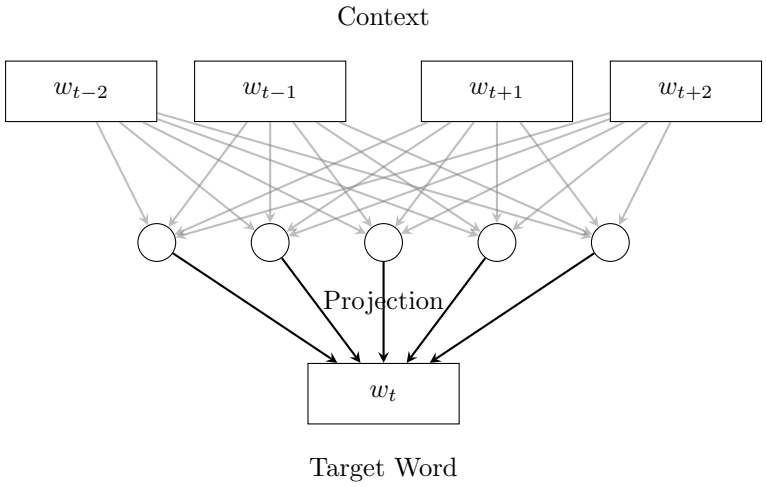

Figure 5: CBOW Architecture

5.1 Architecture

The basic RNN updates its hidden state at each time step:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{2}$$

And generates an output:

$$y_t = W_{hy}h_t + b_y \tag{3}$$

where:

- h_t is the hidden state at time t
- x_t is the input at time t
- y_t is the output at time t
- W_{hh}, W_{xh}, W_{hy} are weight matrices
- b_h, b_y are bias vectors
- σ is an activation function (typically tanh or ReLU)

5.2 Vanishing/Exploding Gradient Problem

The recursive nature of RNNs leads to multiplication of gradients during back-propagation through time, causing either:

- Vanishing gradients: When gradients become extremely small, early time steps don't receive meaningful updates

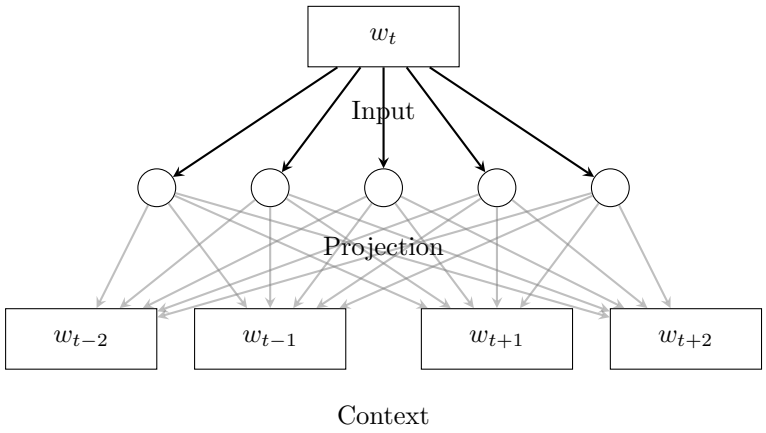

Figure 6: Skip-gram Architecture

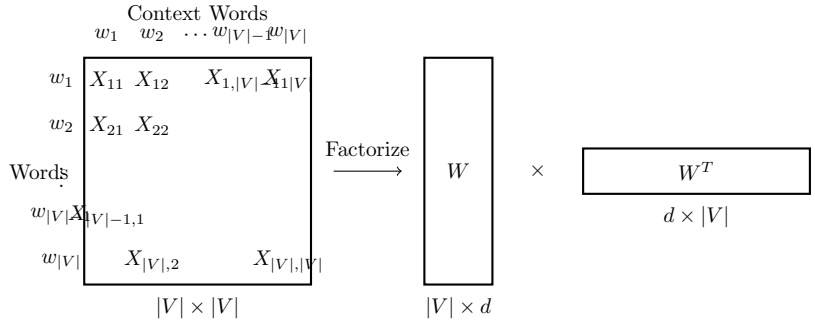

Figure 7: GloVe Co-occurrence Matrix Factorization

- Exploding gradients: When gradients become extremely large, causing unstable training

This limitation motivates the development of LSTM and GRU architectures.

6 Long Short-Term Memory (LSTM) Units

LSTMs extend RNNs with mechanisms to selectively remember or forget information, addressing the vanishing gradient problem.

6.1 Architecture

An LSTM cell contains three gates and a memory cell:

- **Forget gate:** Controls what information to discard.

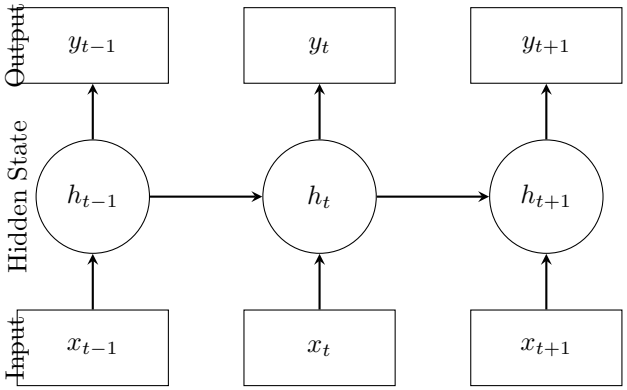

Figure 8: Unfolded Recurrent Neural Network

- **Input gate:** Controls what new information to store.
- **Output gate:** Controls what information to output.
- **Memory cell:** Stores long-term information.

6.2 Mathematical Formulation

The LSTM forward pass is defined by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{6}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{7}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{8}$$
$$h_t = o_t \odot \tanh(C_t) \tag{9}$$

where $\odot$ represents element-wise multiplication.

7 Gated Recurrent Units (GRUs)

GRUs simplify the LSTM architecture while retaining many of its advantages.

7.1 Mathematical Formulation

The GRU forward pass is defined by:

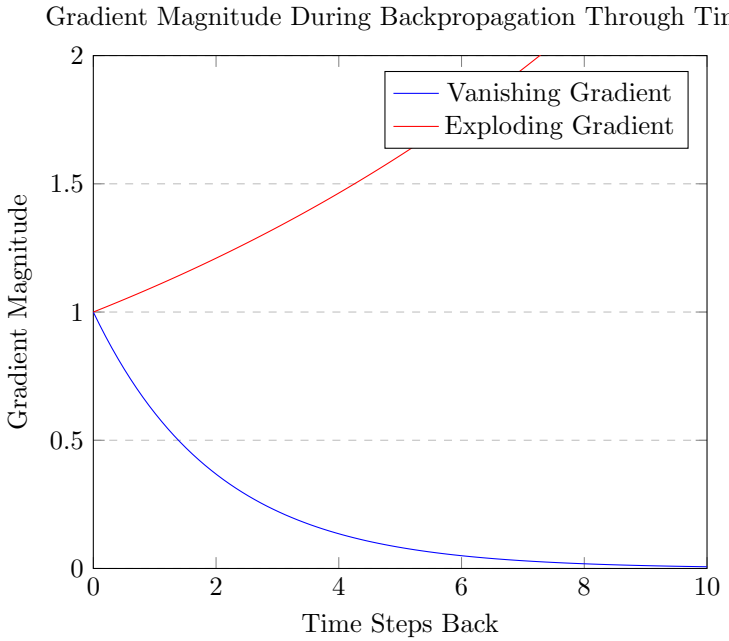

Figure 9: Vanishing and Exploding Gradients in RNNs

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{10}$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{11}$$
$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{12}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{13}$$

7.2 Comparison with LSTM

GRUs offer several advantages over standard LSTM cells:

Feature	**LSTM**	**GRU**
Number of gates	4 (forget, input, output, cell)	2 (reset, update)
States	2 (cell state, hidden state)	1 (hidden state)
Parameters	More	Fewer
Training speed	Slower	Faster
Performance on small data	Better	Comparable

Table 1: Comparison between LSTM and GRU.

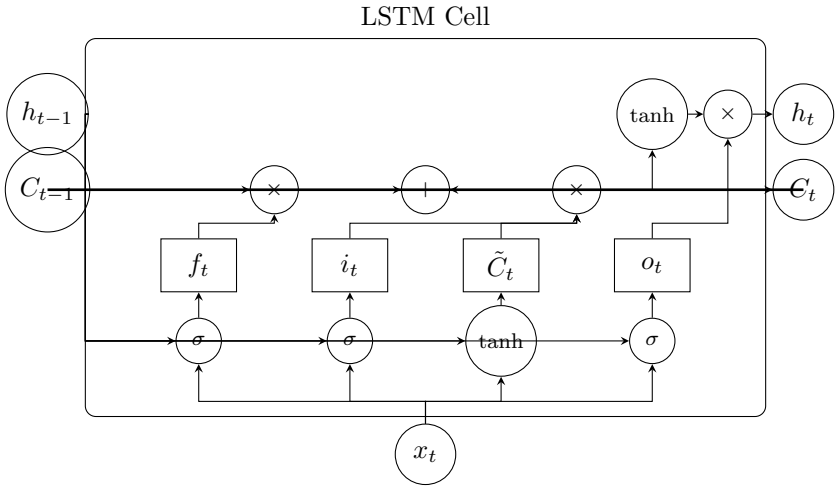

Figure 10: LSTM Cell Architecture

8 Bidirectional RNNs

Bidirectional RNNs process sequences in both forward and backward directions, capturing context from both past and future time steps.

8.1 Architecture

8.2 Mathematical Formulation

The output at each time step is a function of both forward and backward hidden states:

$$\overrightarrow{h}_t = f(x_t, \overrightarrow{h}_{t-1}) \tag{14}$$

$$\overleftarrow{h}_t = f(x_t, \overleftarrow{h}_{t+1}) \tag{15}$$

$$y_t = g(\overrightarrow{h}_t, \overleftarrow{h}_t) \tag{16}$$

where f could be a standard RNN cell, LSTM, or GRU, and g is typically a concatenation followed by a linear projection.

9 Attention Mechanisms

Attention mechanisms allow models to focus on relevant parts of input sequences when making predictions.

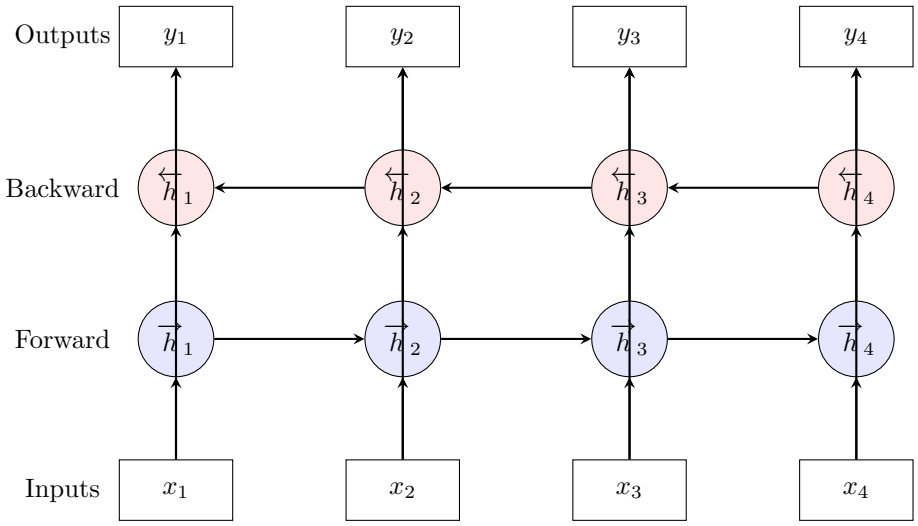

Figure 11: Bidirectional RNN Architecture

9.1 Mathematical Formulation

The attention mechanism computes a weighted sum of encoder hidden states:

$$e_{tj} = f(s_t, h_j) \tag{17}$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \tag{18}$$

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \tag{19}$$

where:

- e_{tj} is the alignment score between decoder state s_t and encoder state h_j
- α_{tj} is the attention weight
- c_t is the context vector

10 Transformer Architecture

Transformers represent a paradigm shift in NLP by replacing recurrent connections with self-attention mechanisms.

10.1 Self-Attention

Self-attention allows each position in a sequence to attend to all positions within the same sequence:

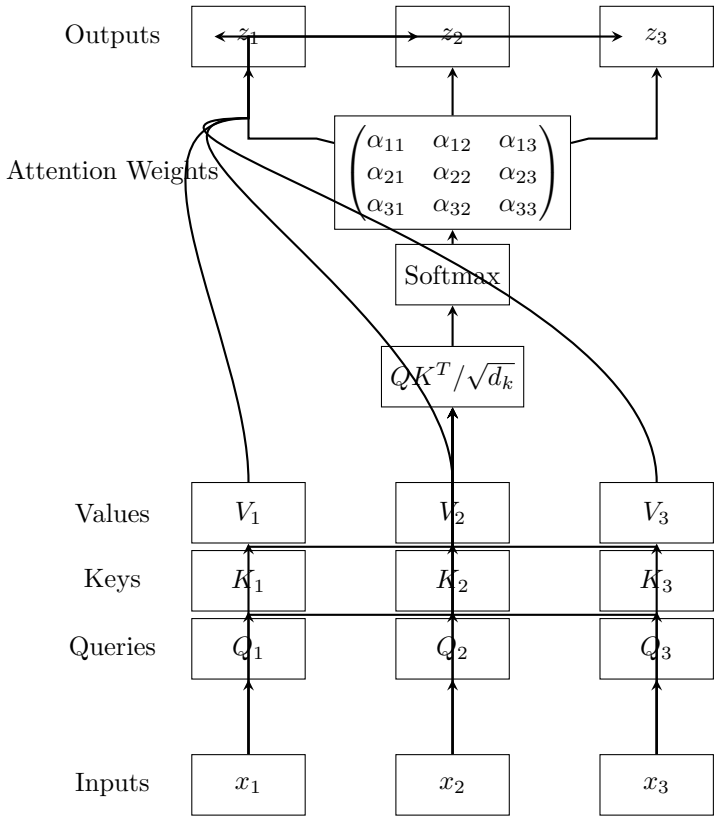

Figure 12: Self-Attention Mechanism

10.2 Multi-Head Attention

Multi-head attention performs multiple attention operations in parallel:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{20}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{21}$$

11 Conclusion

Advanced NLP models have revolutionized how machines understand and generate human language. From RNNs and LSTMs to Transformers and BERT,

the field has seen rapid progress in creating more powerful and efficient architectures. These models power a wide range of applications from machine translation to question answering and text generation.

As research continues, we're seeing improvements in both model performance and efficiency through techniques like parameter-efficient fine-tuning, multi-modal learning, and efficient inference methods. The future of NLP lies in creating models that can handle increasingly complex language tasks while being accessible and efficient enough for widespread deployment.