Recurrent Neural Networks and Language Modeling

Machine Translation: Advanced Topics

Vincent Vandeghinste

KU Leuven – Brussels

2026

Outline

Recurrent Neural Networks

Why Sequences Matter

For many language tasks, **order matters**.

- "dog bites man" $\neq$ "man bites dog"
- meaning depends on previous words
- sentences have variable length

We therefore need models that:

- process **sequences**
- remember what came before

What is a Recurrent Neural Network (RNN)?

A **Recurrent Neural Network (RNN)** is a neural network designed for sequences.

It processes a sentence **one word at a time**.

At each step it takes:

- the current word (vector)
- a memory of previous words (the **hidden state**)

What Does an RNN Produce?

At each time step, the RNN computes:

- an updated **hidden state**
- optionally an **output** (e.g. next word prediction)

The hidden state functions as a **context memory**.

It summarizes everything the network has seen so far.

RNN Unrolled Over Time

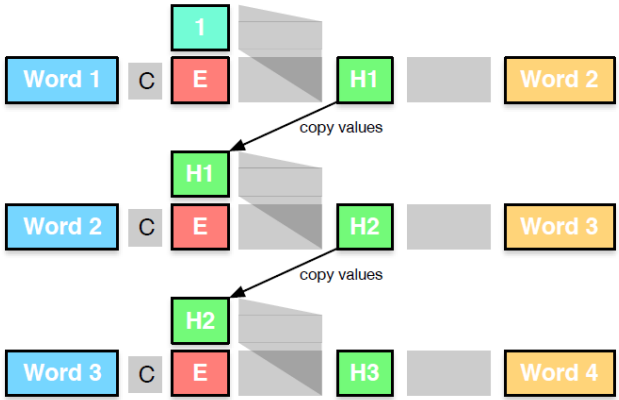

Figure: A simple RNN unrolled over time Koehn 2020

How to Read the Figure

- The same RNN cell is reused at every time step.
- H_1, H_2, H_3 are the hidden states.
- Each hidden state passes information forward.

If the sentence has three words:

$$H_3 = \text{sentence representation}$$

Why RNNs Work for Language

RNNs have two key advantages:

1. **They model word order**
   - hidden state depends on previous words
   - order changes the representation

2. **They handle variable-length sentences**
   - no fixed input size required
   - simply run longer or shorter

Advanced RNN Variants

In practice, simple RNNs are rarely used.

Common variants:

- **LSTM** (Long Short-Term Memory)
- **GRU** (Gated Recurrent Unit)

They improve:

- memory of long-distance dependencies
- training stability

Language Modeling

What is a Language Model?

A **Language Model (LM)** learns how likely word sequences are.

Core question:

If I have seen these words so far,
what word is likely to come next?

Example Predictions

After:

- `she likes` $\rightarrow$ `apples`
- `you are a` $\rightarrow$ `student`
- `<sos>` $\rightarrow$ `i`, `you`, `she`

The model learns these patterns from **data**, not rules.

Language Model as Probability Estimator

A language model assigns a probability to:

$$P(\text{next word} \mid \text{previous words})$$

Example:

- $P(\text{apples} \mid \text{she likes}) = \text{high}$
- $P(\text{teachers} \mid \text{she likes}) = \text{lower}$

It does not "understand" — it detects patterns.

Example: Scoring Sentences

Language Model Intuition

The model compares:

```
she likes apples
she likes teachers
```

If the training data contains many examples of liking apples,
the first sentence receives a higher probability.

Why Language Models Matter for MT

A translation system must produce:

- grammatically correct sentences
- fluent and natural output

Language models help ensure **fluency**

Decoder as Conditional Language Model

In encoder–decoder NMT:

- The **encoder** reads the source sentence.
- The **decoder** predicts target words step by step.

The decoder is a:

conditional language model

It predicts:

$$P(\text{next target word} \mid \text{previous target words, source})$$

Take-Home Message

- RNNs process language sequentially.
- Hidden states act as memory.
- Language models predict the next word.
- NMT decoders are conditional language models.

Understanding language modeling helps us understand how translation systems generate text.

Hands-on

"Google Colab"

You can find the associated Google Colab session HERE.