**Sequence to Sequence Project (Residency Day 2)**

Abuchi Godswill Okeke

School of Computer and Information Sciences, University of the Cumberlands

MSAI532 - M51 Natural Language Processing

Dr. Toni Farley

Feb 01, 2025

**1. Explain Long Short-Term Memory (LSTM) Architecture**

Long Short-Term Memory (LSTM) networks are presented as an improvement over recurrent neural networks (RNNs) by solving the vanishing gradient problem that prevents the model from capturing long-distance dependencies. The three gates; the forget gate that removes unwanted information, the input gate that adds in new information where it is required, and the output gate that controls the information that is passed on to the next time step- help LSTMs to achieve this (Staudemeyer & Morris, 2019). These mechanisms make LSTMs powerful architectures for sequential data tasks like speech recognition, machine translation, and time series prediction (Sutskever, Vinyals, & Le, 2014).

**Example with a real number in Machine Translation**

An example of this is an LSTM model with four layers of 1,000 hidden units that were trained on 12 million English-French sentence pairs. Given the input sentence "I am an MSAI532 student", the model processes it in reverse order as "student MSAI532 an am I", updating the memory cell state Cₜ and the hidden state hₜ at each time step.

* Step 1: Input "student"; the forget gate retains 60% of past memory.
* Step 2: Input "MSAI532"; input gate adds 80% new information.
* Step 3: Input "an"; cell state updates to 0.79.
* Step 4: Input "am", final hₜ = 0.46.
* Step 5: Input "I"; the final encoder state is passed to the decoder.

The decoder generates "Je suis un étudiant MSAI532" using this context vector. This LSTM model achieved a BLEU score of 34.8, outperforming statistical machine translation models (Sutskever et al., 2014).

**2. Why Does LSTM Read the Input Sequence One Time Step at a Time to Obtain a Large Fixed-Dimensional Vector Representation?**

LSTMs process input sequences one time step at a time to incrementally build a fixed-dimensional vector representation that encodes the entire sequence. This approach guarantees that every input word (or token) can contribute to the learner's context, which preserves short and long-term dependencies. In contrast to the traditional feedforward networks that fix input lengths, LSTMs increase their memory size by dynamically changing their hidden states.

This step-by-step processing is especially efficient for sequence-to-sequence tasks such as machine translation. For example, while translating the English sentence “I am a University of the Cumberlands student” to French, the model uses “student Cumberlands the of University a am I,” allowing the process to reverse order to increase the efficiency of translation. The final hidden state of the encoder is used as a 1,000-dimension vector, which encapsulates the meaning of the sentence, and the decoder utilizes this vector to develop the translated sequence (Sutskever et al. 2014).

LSTMs place a strong emphasis on the maintenance of the specific grammatical relationships and semantic dependencies that are significant with context. So, in contrast to most systems, this system allows LSTMs to generalize well, even on longer sequences, which is very important in the areas of natural language processing, speech recognition, and time-series forecasting.

**Example: Translating "I am a University of the Cumberlands student"**

LSTM processes the reversed input: "student Cumberlands the of University a am I"

*Step-by-Step Updates*

Step 1: "student"

* Forget gate: 70% memory retained
* Input gate: 80% new info added
* Cell state: 0.48

Hidden state: 0.32

* Step 2: "Cumberlands"
* Forget: 75%, Input: 85%
* Cell: 0.717, Hidden: 0.46

Step 3-7: Remaining Words Processed Similarly

Final hidden state: 1,000-dimensional vector storing the sentence’s meaning.

Final Output (French Translation)

Decoder generates:

"Je suis un étudiant de l'Université de Cumberlands."

**3. Explain the Model Used in the Paper with Some Examples with Numbers**

The model proposed by Sutskever et al.. (2014) consists of an encoder-decoder architecture using deep LSTM networks. The encoder LSTM reads the input stepwise, synthesizing it into a fixed-dimensional vector that captures short-term and long-term dependencies. The output sequence is generated by the decoder LSTM in a token-by-token manner, using the context in the encoded vector. Each word is generated with reference to the previously generated word, resulting in the predicted sequence being informative and contextual. This technique makes it possible to accurately obtain translations and texts of structured content for differing sequence lengths (Sutskever et al., 2014).

**Example: English-to-French Translation**

**For the input sentence: - “I am taking MSAI-532 residency class,”**

**Step 1: Encoding the Input Sentence**

The model reverses the input sentence before processing:

Original: "I am taking MSAI-532 residency class."

Reversed: "class residency MSAI-532 taking am I."

Each word is transformed into a word embedding vector that is fed into the encoder LSTM. In simple terms, this matrix maps each word to a dense vector, so if our vocabulary has words and our chosen embedding size is (say, 1,000), then each word is first encoded as a one-hot vector and then transformed into a continuous vector via a matrix using the operation:

This means that words carrying similar meanings end up with similar embeddings, which allows the model to capture subtle semantic and syntactic relationships in a far more streamlined way. These embedding vectors serve as the rich, learned representations that the LSTM uses as it processes the input.

As the encoder LSTM processes these word embeddings one at a time, it updates its hidden state using a few simple but powerful gating mechanisms: the input gate, the forget gate, and the output gate. At each time step , the LSTM combines the current word’s information with the context gathered so far, updating its internal cell state ​ and producing a new hidden state . This is done via equations such as:

Where ​ is the output gate and ⊙ denotes element-wise multiplication. This process allows the LSTM to “decide” what new information to add and what old information to discard as it reads the reversed sentence (Sutskever et al., 2014).

**Example of hidden state values at each step:**

* h₁: (0.3, 0.5, …) → "class"
* h₂: (0.6, 0.2, …) → "residency"
* h₃: (0.7, 0.4, …) → "MSAI-532"
* h₄: (0.5, 0.8, …) → "taking"
* h₅: (0.2, 0.9, …) → "am"
* h₆ (Final State): (0.9, 0.7, …) → "I"

The last hidden state (h₆) becomes the fixed-size vector representation (e.g., 1,000-dimensional) summarizing the sentence.

**Step 2: Decoding the Translation**

The decoder LSTM starts with this fixed vector and then generates the translation one word at a time.

It uses previously generated words as context while predicting the next word.

**Example output sequence generation:**

Input to decoder: Fixed vector (from encoder).

* Output: "Je" (90% confidence).
* Next step: Use "Je" + fixed vector.
* Output: "prends" (95% confidence).
* Next step: Use "Je prends" + fixed vector.
* Output: "le" (92% confidence).
* Next step: Use "Je prends le" + fixed vector.
* Output: "cours" (98% confidence).
* Next step: Use "Je prends le cours" + fixed vector.
* Output: "de résidence" (96% confidence).
* Next step: Uses "Je prends le cours de résidence" + fixed vector.
* Output: "MSAI-532" (100% confidence).

Final step: Generate the End-of-Sequence (EOS) token, indicating completion, and the final translation: "Je prends le cours de résidence MSAI-532" is generated.

***For the size and configuration of the model:***

The encoder and decoder LSTMs possessed four layers, with 1 thousand hidden units at each layer. The model was trained on 12 million English-French sentence pairs, and the size of the vocabulary for English was 160,00 and for French was 80,000. This method permits the model to manage varying lengths of input sequences while maintaining the internal and external context relations of the words (Sutskever et al., 2014).

**4. Explain the Experiments in the Paper: Input, Processing Procedure, and Output/Results**

**Input Data**

The experimental training dataset used in the paper consisted of a subset of 12 million parallel sentences consisting of 348M French words and 304M English words from the WMT’14 English-to-French dataset (Sutskever et al., 2014). This large-scale dataset provided extensive language pairs for effective learning. To manage computational efficiency and ensure model robustness, the vocabulary size was limited to 160,000 words for English and 80,000 words for French. Any words that were not part of the defined vocabulary were replaced with a special “UNK” token, which helped the model handle out-of-vocabulary (OOV) words during inference and training (Sutskever, Vinyals, & Le, 2014).

**Processing Procedure**

***Preprocessing***

The preprocessing phase involved decoding, rescoring, and reversing the source sentences, significantly improving translation quality. Beam search decoding maintained a beam of B partial hypotheses, selecting the most probable translation at each step. Also, LSTM rescored the 1000-best lists from an SMT system by averaging log probabilities, enhancing accuracy.

Reversing source sentences was another preprocessing method used; this reduced the test perplexity from 5.8 to 4.7, and BLEU scores increased from 25.9 to 30.6. This method introduced short-term dependencies, improving memory utilization and making long-sequence learning more effective (Sutskever et al., 2014).

***Model Training***

The model training was fairly easy; four layers of deep LSTMs were used, including 1000 cells at each layer and 1000-dimensional word embeddings with 160,000 input and 80,000 output vocabularies, and unknown words were replaced with “UNK” tokens. For example, using the Sentence: "I am taking an Artificial Intelligence class." So, if "Artificial" is missing from the vocabulary, then it is replaced with <UNK>, and the processed sentence is “I am taking <UNK> Intelligence class” (Sutskever et al., 2014).

Also, the training was done in such a way as to maximize the probability of the correct translation given an input sequence. To achieve this, Stochastic Gradient Descent (SGD) was implemented as the primary optimization method. Additionally, the training was done using a parallelized 8-GPU machine beam, with each layer running on different GPUs and communicating its activations to the next GPU once computed (Sutskever et al., 2014).

**Results and Performance**

***BLEU Score (Bilingual Evaluation Understudy Score)***

The model’s effectiveness was evaluated using the BLEU metric, which measures the quality of machine-generated translations. A BLEU score of 34.8 was achieved by the LSTM-based model, significantly outperforming the phrase-based statistical machine translation (SMT) baseline with a score of 33.3. However, when the LSTM model was utilized to rerank SMT-generated translations, the BLEU score improved further to 36.5, showing the superior performance of the sequence-to-sequence learning approach (Sutskever et al., 2014).

Also, unlike previous neural machine translation models, which often struggled with longer input sequences, the deep LSTM model successfully maintained context and dependencies across long sentences.

**5. Final Thoughts and Conclusion**

Sutskever et al. (2014) present a breakthrough in sequence-to-sequence learning. The paper demonstrated that deep LSTMs can successfully model complex sequence mappings, outperforming traditional phrase-based statistical machine translation (SMT) systems.

One of the most impactful findings I got from this paper was the effectiveness of reversing input sequences. This simple yet powerful technique significantly improved the model’s ability to capture short-term dependencies, making it easier to learn word alignments between source and target languages. Furthermore, the model performed exceptionally well on long sentences, a challenge that had previously hindered RNN-based architectures. By reducing the “time lag” between corresponding words in the source and target, reversing the input not only speeds up the training process but also leads to more accurate translations, especially for tougher, longer sentences. This key finding is significant because it offers a straightforward and practical fix for a common challenge in sequence learning, giving us valuable clues on how to better align and manage the timing of words in neural networks.

Despite its success, the model had some limitations; for example, in handling out-of-vocabulary words, the translation accuracy was potentially reduced because the use of a fixed vocabulary meant that words outside the training dataset had to be replaced with a “UNK” token.

In addition, training a model of this scale takes a lot of computing power, which can be a major challenge for researchers with limited hardware. Also, sticking to a fixed vocabulary makes it hard to quickly adapt to new terms or specialized language. These issues clearly show that we need to develop more flexible and resource-efficient methods.

In summary, the paper provided a good foundation for modern neural machine translation (NMT) systems. The proposed LSTM-based approach demonstrated a potential outcome of learning a sequence end-to-end, paving the way for further advancements in deep learning for natural language processing.

**References**

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding Long Short-Term Memory Recurrent Neural Networks – A tutorial-like introduction. arXiv preprint arXiv:1909.09586.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks*. In *Advances in Neural Information Processing Systems* (pp. 3104–3112).