**Neural Machine Translation 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. Describe RNN Encoder-Decoder from section 2.1: use some numbers to exemplify.**

The Recurrent Neural Network (RNN) Encoder-Decoder framework is a network that encodes an input sentence into a fixed-length vector , which is then used to generate a translated output sentence (Bahdanau et al., 2015). This is a two-step process for translating sentences. First, the **encoder** reads the whole sentence in one language and turns it into a summary of a single fixed-length vector code. Then, the **decoder** takes that code and generates the translation in the target language, word by word.

For example, imagine translating **“I love MSAI532 Class”** from English to French.

**Encoding Phase:**

Since the sentence “I love MSAI532 Class” has four (4) words.

Each word is converted into a hidden state.

Then, the final hidden state stores the sentence’s meaning.

*Example hidden states:*

h₁: (0.3, 0.5, …) → “I”

h₂: (0.6, 0.2, …) → “love”

h₄ (Final State): (0.8, 0.4, …) → “Class” (summary of sentence)

**Decoding Phase:**

Once the encoder has processed the input sentence and generated a final hidden state (h₄), the decoder takes over to generate the translation. The decoder works one word at a time, using the previously generated word and the context from the encoder's hidden state to predict the next word.

*Decoding steps (probability assignment):*

Step 1: "J’" (90%)

Step 2: "aime" (95%)

Step 3: "MSAI532" (100%)

Step 4: "Classe" (92%)

Final Output, the completed French translation is: "J’adore la classe MSAI532."

One limitation of this is that if the sentence is too long, squeezing everything into one fixed-length summary (vector) becomes difficult. The longer the sentence, the harder it is to capture all the details, making translations less accurate.

**2. Describe the Decoder general description from section 3.1: use some numbers to exemplify**

The decoder generates a target sentence by predicting the probability of each target word given the previous words and a context vector. Each word is generated using the formula:

where is the decoder's hidden state at a time , and ​ is a context vector specific to each target word (Bahdanau et al., 2015, p. 3).

Based on this, for instance, we want to translate the word “**I am an executive program student at the University of the Cumberlands.**” Since the input sentence has **12 words** (so ), the model assigns different attention scores (probability of each target word given the previous words and a context vector) to different words at each step instead of relying on just one overall summary. When generating the first word of the translation, it might assign 50% attention to word 1, 30% to word 2, and the rest spread out across other words. For the second word of the translation, the focus shifts, maybe 60% on word 3 and 40% on word 4. This continues for every word in the translation until completion.

**Example: Translating Sentence with Attention Weights**

Input Sentence (English, 12 words, Tₓ = 12): "I am an executive program student at the University of the Cumberlands."

**Step 1**: Predicting the First Word ("Je")

The model distributes attention across the input words:

"I" (50%), "am" (30%), other words (spread across remaining 20%)

The decoder predicts: "Je" (French for "I").

**Step 2**: Predicting the Second Word ("suis")

Focus shifts:

"am" (60%), "an" (40%), rest = 0%.

The decoder generates: "suis" (French for "am").

**Step 3:** Predicting "un" (for "an")

Focus on:

"an" (60%), "executive" (40%).

It predicts: "un" (French for "an").

**Step 4-6:** Translating "executive program student"

The model gradually shifts focus:

"executive" (50%), "program" (40%), "student" (10%) → "étudiant en programme exécutif"

**Step 7-12**: Translating "at the University of the Cumberlands"

The attention shifts again:

"University" (60%), "Cumberlands" (40%) → "à l’Université des Cumberlands."

**Final Translation (French)**

"Je suis un étudiant en programme exécutif à l’Université des Cumberlands."

This is an improvement of the previous models because the baseline models had trouble when sentences got long because they tried to squeeze all the information into a single vector dimension. This approach **lets the model focus on the most relevant words at each step**, leading to **better and more accurate translations**, especially for longer sentences.

**3. Describe Encoder from section 3.2: use some numbers to exemplify.**

The encoder in the proposed model by Bahdanau et al. (2015), is a Bidirectional RNN (BiRNN), which consists of a forward and backward RNN. The forward RNN processes the sentence from left to right, producing hidden states , while the backward RNN processes it from right to left, yielding . The final annotation for each word is obtained by concatenating the forward and backward hidden states:

This bidirectional structure allows the model to capture both past and future context for each word (Bahdanau et al., 2015, p. 4).

Based on this, the **encoder** processes a sentence before translating it. Instead of just reading from start to finish, this model uses a **bidirectional approach**, meaning it reads **both forward and backward** to get a complete understanding of the sentence. For example, let’s say we have the sentence: "**The residency class ends on Sunday."**

A traditional approach would read the words from left to right, processing each word one by one. However, the encoder in section 3.2 reads twice:

Forward: It reads the sentence as:

The → residency → class → ends → on → Sunday.

Backward: It reads from the end back to the start:

Sunday → on → ends → class → residency → The.

Since the sentence has six (**6) words**, assuming the model assigns this **hidden focus state** to each word, capturing its meaning in the sentence. That is:

* Forward pass:
  + "The" → h₁ = 0.2
  + "residency" → h₂ = 0.5
  + "class" → h₃ = 0.8
  + "ends" → h₄ = 0.3
  + "on" → h₅ = 0.6
  + "Sunday" → h₆ = 0.7
* Backward pass (processed in reverse):
  + "Sunday" → h₆' = 0.9
  + "on" → h₅' = 0.4
  + "ends" → h₄' = 0.2
  + "class" → h₃' = 0.7
  + "residency" → h₂' = 0.3
  + "The" → h₁' = 0.5

For each word, the final annotation (or encoded representation) is a combination of both forward and backward values. So, for **“residency,”** the encoded value would be:

This means "residency" gets meaning from both its previous words and the upcoming. By reading in two directions and combining the results, the encoder creates a better representation of the sentence, which helps the decoder translate more accurately.

**4. Explain the experiments in the paper (section 4): input, processing procedure, and output/results.**

***Input***

**WMT ‘14 dataset, a large** collection of English-French sentences reduced to pairs containing about **348 million words, was used as the input dataset**. Only the 30,000 most common words in each language were used. Any rare words were replaced with "[UNK]" (unknown word token). Then, the input data is split into a training set, sentences used to teach the model, a validation set used to fine-tune the model, and a test set, 3,003 unseen sentences used to evaluate accuracy (Bahdanau et al., 2015).

***Processing procedure***

Two models were trained: the baseline RNN Encoder-Decoder (RNNencdec) and the proposed RNN with an attention mechanism (RNNsearch) having 1000 hidden units. Both models were trained using mini-batches of 80 sentences at a time and optimized over 5 days. Two versions of these models were trained, one using sentences up to 30 words long and the second version using sentences up to 50 words long. After training, a beam search algorithm was used to generate the best possible translations (Bahdanau et al., 2015).

***output/results***

The output shows that the new model (RNNsearch) performed significantly better than the baseline model (RNN Encoder-Decoder) in translation tasks. Using the BLEU score, which measures how close machine-generated translations are to human translations, the evaluation was conducted on a test set of 3,003 sentences. The results show that RNNsearch was particularly effective for longer sentences, where the baseline model struggled. This tells us that the attention mechanism in RNNsearch helped maintain translation accuracy across different sentence lengths.

**5. Discuss quantitative results from section 5.1.**

The baseline model (RNN Encoder-Decoder) got a BLEU score of 17.82 (out of 100), while the new attention-based model (RNNsearch) got 26.75, a huge 50% improvement from the baseline model. When RNNsearch was trained longer, it reached 28.45, getting close to Moses (a professional translation system scoring 33.30). Additionally, RNNsearch showed no performance drop for long sentences, unlike the baseline model, which degraded significantly when sentence length increased beyond 30 words.

**Reference**

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Proceedings of the International Conference on Learning Representations (ICLR). Retrieved from <https://arxiv.org/abs/1409.0473>