An Introduction to Neural Machine Translation

Prof. John D. Kelleher
@johndkelleher

ADAPT Centre for Digital Content Technology
Dublin Institute of Technology, Ireland

June 25, 2018
Outline

The Neural Machine Translation Revolution

Neural Networks 101

Word Embeddings

Language Models

Neural Language Models

Neural Machine Translation

Beyond NMT: Image Annotation
Microsoft Translator launching Neural Network based translations for all its speech languages

Microsoft Translator is now powering all speech translation through state-of-the-art neural networks.

All speech translation apps that use this service, such as Skype Translator and the Microsoft Translator app for mobile devices, are now using neural network technology. Furthermore, the technology is available to all developers and end-users who want to use the Microsoft Translator speech API to integrate the technology into their favorite apps and services.

In addition to the nine languages supported by the Microsoft Translator speech API, namely Arabic, Chinese Mandarin, English, French, German, Italian, Brazilian Portuguese, Russian and Spanish, neural networks also power Japanese and Korean text translations. These eleven languages together represent more than 80% of the translations performed daily by Microsoft Translator.

Neural network technology has been used for the last few years in many artificial intelligence scenarios, such as speech and image processing. Many of these capabilities are available through Microsoft Cognitive services. Neural networks are making in-roads into the machine translation industry, providing major advances in translation quality over the existing industry-standard Statistical Machine Translation (SMT) technology. Because of how the technology functions, neural networks better capture the context of full sentences before translating them, providing much higher quality and more human-sounding output.
Higher quality neural translations for a bunch more languages

Last November, people from Brazil to Turkey to Japan discovered that Google Translate for their language was suddenly more accurate and easier to understand. That's because we introduced neural machine translation—using deep neural networks to translate entire sentences, rather than just phrases—for eight languages overall. Over the next couple of weeks, these improvements are coming to Google Translate in many more languages, starting right now with Hindi, Russian and Vietnamese.

Neural translation is a lot better than our previous technology, because we translate whole sentences at a time, instead of pieces of a sentence. (Of course...

Image from https://www.blog.google/products/translate
Facebook finishes its move to neural machine translation

Posted Aug 3, 2017 by John Mannes (@JohnMannes)

Facebook announced this morning that it had completed its move to neural machine translation —
Linguee’s Founder Launches DeepL in Attempt to Challenge Google Translate

by Florian Faes on August 30, 2017

Barely two years after bursting into the translation tech scene, neural machine translation (NMT) is everything the MT community is talking about. Microsoft, Google, Facebook, and other large technology companies have all transitioned to NMT, as did the European Patent Office and the World Intellectual Property Organization. Even end-buyers are starting to build their own systems based on open-source models.
Neural Networks 101
What is a function?

A function maps a set of inputs (numbers) to an output (number)\(^1\)

\[ \text{sum}(2, 5, 4) \rightarrow 11 \]

\(^1\) This introduction to neural network and machine translation is based on: Kelleher (2016)
What is a **weightedSum** function?

\[
\text{weightedSum}([x_1, x_2, \ldots, x_m], [w_1, w_2, \ldots, w_m])
\]

- **Input Numbers**
- **Weights**

\[
= (x_1 \times w_1) + (x_2 \times w_2) + \cdots + (x_m \times w_m)
\]

**Example**:

\[
\text{weightedSum}([3, 9], [-3, 1])
\]

\[
= (3 \times -3) + (9 \times 1)
\]

\[
= -9 + 9
\]

\[
= 0
\]
What is an activation function?

An activation function takes the output of our weightedSum function and applies another mapping to it.
What is an **ACTIVATION** function?

\[
\text{logistic}(z) = \frac{1}{1 + e^{-z}}
\]

\[
\text{tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}
\]

\[
\text{rectifier}(z) = \max(0, z)
\]
What is an activation function?

\[
\text{ACTIVATION} = \ \text{LOGISTIC}(\text{WEIGHTEDSUM}((x_1, x_2, \ldots, x_m), (w_1, w_2, \ldots, w_m)))
\]

\[
\text{LOGISTIC}(\text{WEIGHTEDSUM}([3, 9], [-3, 1]))
\]

\[
= \text{LOGISTIC}((3 \times -3) + (9 \times 1))
\]

\[
= \text{LOGISTIC}(-9 + 9)
\]

\[
= \text{LOGISTIC}(0)
\]

\[
= 0.5
\]
What is a **Neuron**?

The simple list of operations that we have just described defines the fundamental building block of a neural network: the **Neuron**.

\[
\text{Neuron} = \text{activation} \left( \text{weightedSum} \left( \left[ x_1, x_2, \ldots, x_m \right], \left[ w_1, w_2, \ldots, w_m \right] \right) \right)
\]

- **Neuron** = activation(\text{weightedSum}([x_1, x_2, \ldots, x_m], [w_1, w_2, \ldots, w_m]))
- 
  - Input Numbers
  - Weights
What is a **Neuron**?

\[ \sum \varphi \]

\[ x_0 \]

\[ x_1 \]

\[ x_2 \]

\[ x_3 \]

\[ \ldots \]

\[ x_m \]

\[ w_0 \]

\[ w_1 \]

\[ w_2 \]

\[ w_3 \]

\[ w_m \]

Activation

Image generated using code from Martin Thoma

https://github.com/MartinThoma
What is a **Neural Network**?

![Diagram of a neural network with layers labeled as Input Layer, Hidden Layer 1, Hidden Layer 2, Hidden Layer 3, and Output Layer 4.}]
Training a **Neural Network**

- We train a neural network by iteratively updating the weights.
- We start by randomly assigning weights to each edge.
- We then show the network examples of inputs and expected outputs and update the weights using **Backpropogation** so that the network outputs match the expected outputs.
- We keep updating the weights until the network is working the way we want.
Word Embeddings
Word Embeddings

- Language is sequential and has lots of words.
“a word is characterized by the company it keeps”

— Firth, 1957
Word Embeddings

1. Train a network to predict the word that is missing from the middle of an n-gram (or predict the n-gram from the word)

2. Use the trained network weights to represent the word in vector space.
Word Embeddings

Each word is represented by a vector of numbers that positions the word in a multi-dimensional space, e.g.:

\[
\begin{align*}
\text{king} &= \langle 55, -10, 176, 27 \rangle \\
\text{man} &= \langle 10, 79, 150, 83 \rangle \\
\text{woman} &= \langle 15, 74, 159, 106 \rangle \\
\text{queen} &= \langle 60, -15, 185, 50 \rangle
\end{align*}
\]
Word Embeddings

\[ \text{vec}(\text{King}) - \text{vec}(\text{Man}) + \text{vec}(\text{Woman}) \approx \text{vec}(\text{Queen})^2 \]

\(^2\) Linguistic Regularities in Continuous Space Word Representations Mikolov et al. (2013)
Language Models
Language Models

- Language is **sequential** and has lots of words.
A language model can compute:

1. the probability of an upcoming symbol:

   \[ P(w_n|w_1, \ldots, w_{n-1}) \]

2. the probability for a sequence of symbols\(^3\)

   \[ P(w_1, \ldots, w_n) \]

\(^3\) We can go from 1. to 2. using the Chain Rule of Probability \[ P(w_1, w_2, w_3) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \]
Language models are useful for machine translation because they help with:

1. word ordering

\[ P(Yes \ I \ can \ help \ you) > P(Help \ you \ I \ can \ yes) \]

2. word choice

\[ P(Feel \ the \ Force) > P(Eat \ the \ Force) \]

\(^4\) Unless its Yoda that speaking
Neural Language Models
Recurrent Neural Networks

A particular type of neural network that is useful for processing sequential data (such as, language) is a **Recurrent Neural Network**.
Recurrent Neural Networks

Using an RNN we process our sequential data one input at a time.

In an RNN the outputs of some of the neurons for one input are feed back into the network as part the next input.
Simple Feed-Forward Network
Recurrent Neural Networks
Recurrent Neural Networks
Recurrent Neural Networks
Recurrent Neural Networks
Recurrent Neural Networks
Recurrent Neural Networks
Recurrent Neural Networks
\[ h_t = \phi \left( (W_{hh} \cdot h_{t-1}) + (W_{xh} \cdot x_t) \right) \]

\[ y_t = \phi \left( W_{hy} \cdot h_t \right) \]

**Figure:** Recurrent Neural Network
Recurrent Neural Networks

Output:

$y_1$  $y_2$  $y_3$  $y_t$  $y_{t+1}$

$h_1$  $h_2$  $h_3$  $h_t$  $h_{t+1}$

Input:

$x_1$  $x_2$  $x_3$  $x_t$  $x_{t+1}$

Figure: RNN Unrolled Through Time
Hallucinating Text

Output:

\[ *\text{Word}_2 \quad *\text{Word}_3 \quad *\text{Word}_4 \quad \cdots \quad *\text{Word}_{t+1} \]

Input:

\[ \text{Word}_1 \]

\[ h_1 \quad h_2 \quad h_3 \quad \cdots \quad h_t \]
Hallucinating Shakespeare

PANDARUS: Alas, I think he shall be come approached and the day When little srain would be attain’d into being never fed, And who is but a chain and subjects of his death, I should not sleep.

Second Senator: They are away this miseries, produced upon my soul, Breaking and strongly should be buried, when I perish The earth and thoughts of many states.

DUKE VINCENTIO: Well, your wit is in the care of side and that.

From: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Neural Machine Translation
Neural Machine Translation

1. RNN Encoders
2. RNN Language Models
Encoders

Figure: Using an RNN to Generate an Encoding of a Word Sequence
Language Models

Output:

\*Word_2 \*Word_3 \*Word_4 \*Word_{t+1}

\[ h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow \cdots \rightarrow h_t \]

Input:

Word_1 Word_2 Word_3 Word_t

0/0

Figure: RNN Language Model Unrolled Through Time
Figure: Using an RNN Language Model to Generate (Hallucinate) a Word Sequence
**Encoder-Decoder Architecture**

**Figure:** Sequence to Sequence Translation using an Encoder-Decoder Architecture
Neural Machine Translation

Figure: Example Translation using an Encoder-Decoder Architecture
Beyond NMT: Image Annotation
Image Annotation

1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

14x14 Feature Map

Image from Image from Show, Attend and Tell: Neural Image Caption Generation with Visual Attention Xu et al. (2015)
Thank you for your attention

john.d.kelleher@dit.ie
@johndkelleher
www.machinelearningbook.com
https://ie.linkedin.com/in/johndkelleher

Acknowledgements: The ADAPT Centre is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References I


