Machine Translation: Summer Term 2021

SMT – NMT Intuitions

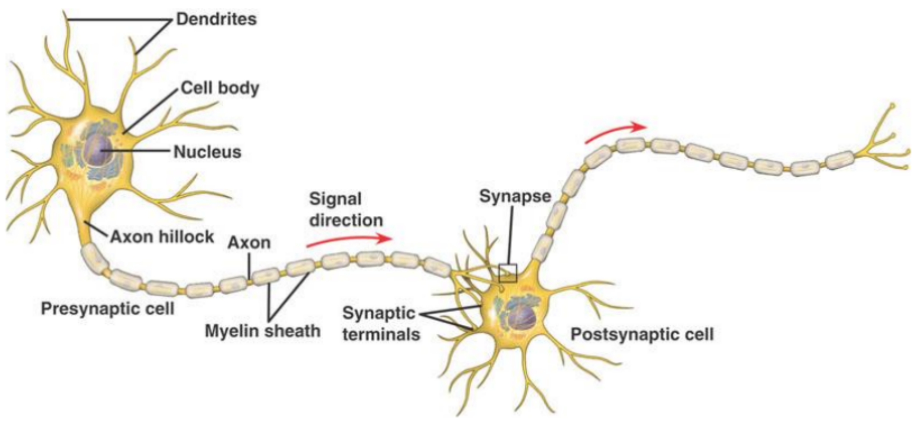
Sangeet Sagar (7009050)
sasa00001@stud.uni-saarland.de

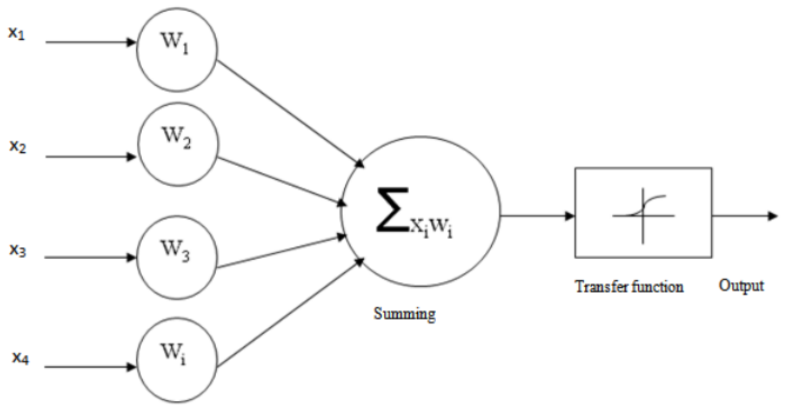
July 10, 2021

1. **What is cool and what is not so cool about PB-SMT?**
   Cool
   - It's one of the most successful LTs to date like ASR, search engines, etc.
   - Brought machine translation into daily use.
   - It's language agnostic where input data is required.
   - It works for many language pairs.

   Not so cool
   - 5-15 components are estimated individually.
   - Lots of independence assumptions are made.
   - Lots of heuristics and human insights and engineering in the components.
   - Not optimized against overall loss function.

2. **In your own words, please describe how a biological neuron cell works and how an artificial neuron (AN) is loosely modelled on a biological neuron:**





A neuron (also known as a nerve cell) is an electrically excitable cell that takes up, processes, and transmits information through electrical and chemical signals. A typical neuron is divided into three parts: the cell body, the dendrites, and the axon.
All neurons are electrically excitable. The electrical impulse mostly arrives on the dendrites, gets processed into the cell body to then move along the axon. On its all length an axon functions merely as an electric cable, simply transmitting the signal. Once the electrical reaches the end of the axon, at the synapses, things get a little more complex.
The key to neural function is the synaptic signaling process, which is partly electrical and partly chemical. Once the electrical signal reaches the synapse, a special molecule called a neurotransmitter is released by the neuron. This neurotransmitter will then stimulate the second neuron, triggering a new wave of electrical impulse, repeating the mechanism described above. [Source: HOW DOES A NEURON WORK?]

Artificial neuron also known as perceptron is the basic unit of the neural network. In simple terms, it is a mathematical function based on a model of biological neurons. It can also be seen as a simple logic gate with binary outputs. They are sometimes also called perceptrons. Each artificial neuron has the following main functions:

- Takes inputs from the input layer
- Weighs them separately and sums them up
- Pass this sum through a nonlinear function to produce output.

[Source: What Is The Relation Between Artificial And Biological Neuron?]

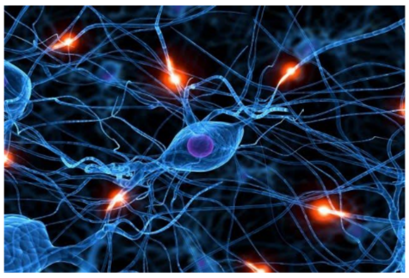
3. **What are the parameters of the artificial neuron in (2) above?**
Parameters of artificial neural networks are the ones that are trainable. These are the number of layers, neurons, or nodes in these layers, the weight associated with these nodes, and bias.

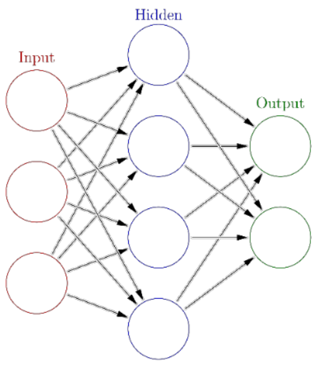
4. **Mathematically, what is the function computed by the AN in (2) above?**

$$a_1 = \sigma(\sum_{i=1}^{n} x_i \cdot w_i) \tag{1}$$

where σ is the activation function.

5. **In terms of a high level metaphor, how does learning happen in biological neural networks and in artificial neural networks (ANNs)?**

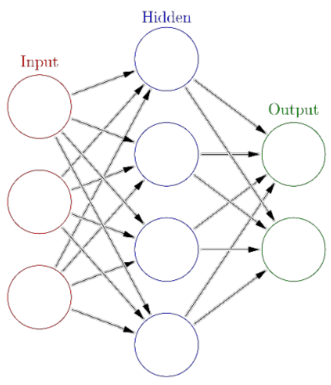




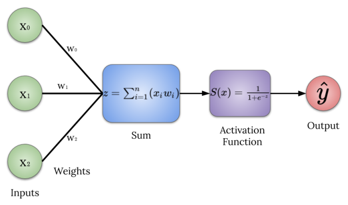
Each neuron learns by moving along the gradient of improvement as defined by synaptic learning rules. At the network and system level, this results in behavior optimization over time. [Source: How Does An Individual Neuron Learn?]
The ANNs adapt and modify their architecture in order to learn. They update weights of connections based on input and desired output. Based on the difference between the actual value and the value that

was outputted by the network, an error value is computed and sent back through the system. For each layer of the network, the error value is analyzed and used to adjust the threshold and weights for the next input. In this way, the error keeps becoming marginally lesser each run as the network learns how to analyze values. This procedure is known as backpropagation and is applied continuously through a network until the error value is kept at a minimum. [Source: How Do Artificial Neural Networks Learn?]

6. **What is function composition? In your own words, how does the ANN below learn the composition $f(g(x))$ of two functions f and g? In the picture below, where is f, and where is g?**

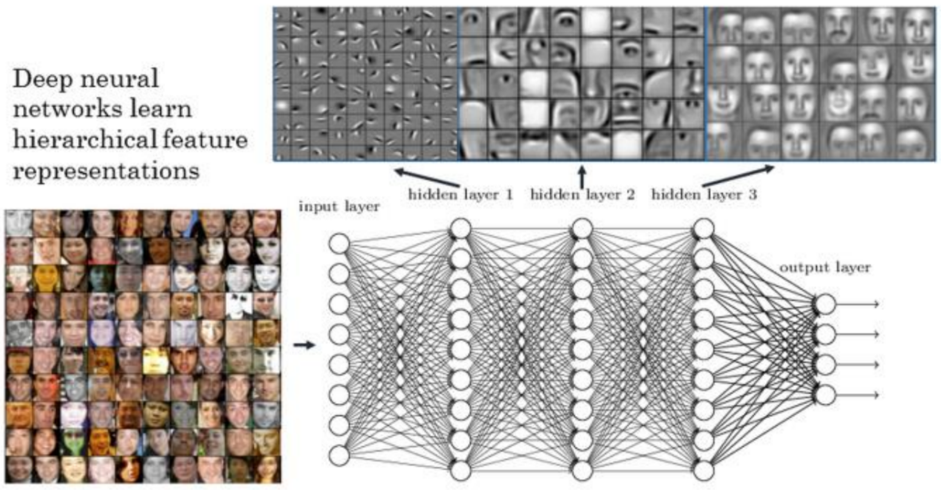


Function composition is the combination of two function to form a new function. One simply takes the output of the first function and uses it as the input to the second function. [Source: Math Insight]



Here f is the activation function and $g(x)$ represents the affine function or the weighted sum of inputs and the weights. One could simply write this function in vector form $g(x) = W^T \cdot x + b$, where W is the weight matrix, x is the input vector and b is the bias.
In fitting a neural network, backpropagation computes the gradient of this composition function $f(g(x))$ with respect to each of the weights of the network by the chain rule, computing the gradient one layer at a time. The gradient shows how much the parameters like weights and biases need to change in order to minimize the cost function. This is done by iterating backward from the last layer and repeatedly adjusting the weights of the connections in the network.

7. **In your own words please describe what is "feature and representation learning" by an ANN and how is this different from feature engineering in classical supervised machine learning:**

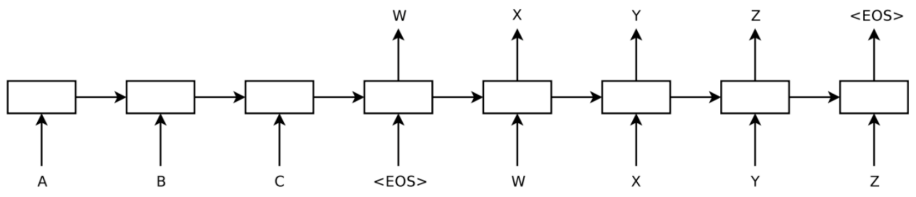


Feature and representation learning is a set of techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data. [Source: Wikipedia]
In the given scenario, all the colored images become the input to the network. The first hidden layer that is completely connected to the input layer learns features that are based on these inputs. The next hidden layer learns features based on the previously hidden layer like the edge of the nose, lips, or eyebrow. Similarly, the next hidden layer learns feature representations based on the last hidden layer.
In feature learning we let out system decide what are the best features for the desired output, however, in feature engineering, new information is created that was not there previously, often by using domain-specific knowledge or by handcrafting new features.

8. **Please list some major ANN architectures.**
   ANN, RNN, CNN, LSTM

9. **In your own words, please describe how the below is a neural MT model:**

$$p(t|s) = p(y_1 \cdots y_m | \, x_1 \cdots x_n) = \prod_{i=1}^{m} p(y_i | y_1 \cdots y_{i-1}, \, x_1 \cdots x_n)$$



Sutskever et al. 2014

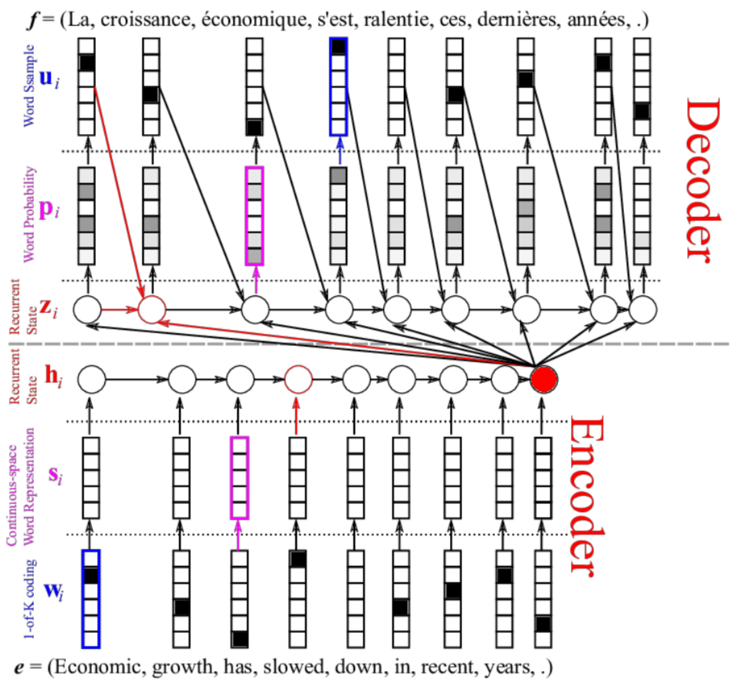
In your description, please refer to sequence to sequence, encoder-decoder models. What licenses the equation above?
This is a neural network with encoder-decoder architecture. It translates tokens $A, B, C, < EOS >$ into a target language and the resulting tokens are $W, X, Y, Z, < EOS >$. The encoder part is where the input is given to the network and the decoder part is where the translations into the target language are observed. The mode is called seq2seq because given a sequence of inputs we get a sequence of outputs. The encoder reads the input sequence and encapsulates the information in internal state vectors or context vectors. The output of the encoders is discarded and only the internal states are preserved. These capture information of all input elements for the decoder to make accurate predictions.
The decoder of the network is an LSTM whose initial states are initialized to the final states of the encoder i.e. the context vector to the encoder's final cell is input to the first cell of the decoder block. Using these initial states, the decoder starts generating the output sequence, and these outputs are also

taken into consideration for future outputs. [Source: Seq2Seq Models]
In the given equation $x_1, x_2, \cdots, x_n$ represents the encoder's input sequence and $y_1, y_2, \cdots, y_n$ represents the decoder's output sequence. When the model comes into action, it conditions the entire input tokens $A, B, C, < EOS >$ or context vector to output the first token W. So this can be formulated as $p_{y_1} = p(y_1|x_1, x_2, x_3, x_4)$. Again, the model conditions the entire context vector as well as all the previous outputs of the decoder to generate X. This can be formulated as $p_{y_2} = p(y_2|y_1, x_1, x_2, x_3, x_4)$. So we see that the model is carrying the full context of the given input as well as context of words that it has already translated. Hence, the final equation of $p(t|s)$ i.e. probability of the translated tokens given the input sequence:

$$p(t|s) = \prod_{i=1}^{m} p_{y_i}$$

10. **What are hot-one encodings?**
One hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1. [Source: What is one hot encoding?]

11. **In your own words, please describe the following NMT architecture: where are hot-one encodings, the word embeddings, where are the contextualised embeddings, where are the hidden states, where is the context vector linking encoder and decoder**



f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)
Word Ssample $\mathbf{u}_i$
Word Probability $\mathbf{p}_i$
Recurrent State $\mathbf{z}_i$
Decoder
Recurrent State $\mathbf{h}_i$
Continuous-space Word Representation $\mathbf{s}_i$
1-of-K coding $\mathbf{w}_i$
Encoder
e = (Economic, growth, has, slowed, down, in, recent, years, .)

This figure represents an RNN encoder-decoder that encodes a variable-length sequence into a fixed-length vector representation and decodes a fixed-length vector back into a variable-length sequence.
The input to the encoder is a one-hot encoded vector representation of the input sequence of words. These one-hot vectors are projected into a continuous-space representation or what we call static word embeddings. These are dense vectors and are in much lower dimensions as one-hot vectors. They are further summarized by RNN. We apply the recurrent activation function recursively over the input sequence, until the end when the final internal state of the RNN h_T is the summary of the whole input sentence. Now that we have a nice fixed-size representation of a source sentence, the decoder block comes into action. We start by computing the RNN's internal state z_i based on the summary vector h_T of the source sentence, the previous word u_{i-1} and the previous internal state z_{i-1}. With the decoder's internal hidden state z_i ready, we can now score and normalize each target word so as to get proper probabilities for each word. Now we use the probability distribution over the target words to select a

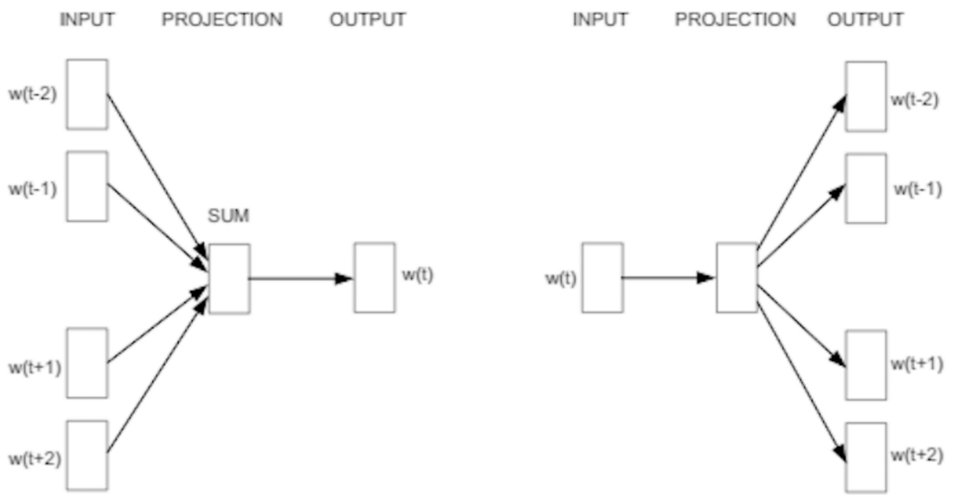
word by sampling the distribution.[Source: Encoder-Decoder Architecture for Machine Translation]
Starting bottom to top, we have

1. one hot encoded vector
2. word embeddings
3. contextualized word embeddings
4. red circle: context vector linking encoder and decoder
5. decoder's hidden state

12. **What are word embeddings?**
Word embeddings are continuous vector representations of source words (in the form of a one-hot encoded vector). It is a learned representation for text where words that have the same meaning have a similar representation.

13. **What is the basic intuition that word embeddings are based on and that they try to capture: think about the distributional hypothesis/distributional semantics.**

14. **Which of the two diagrams below correspond to skip-gram and which to CBOW:**



Left- CBOW (continuous bag of words)
Right- Skip-gram.
In the CBOW model, the distributed representations of context (or surrounding words) are combined to predict the word in the middle. While in the Skip-gram model, the distributed representation of the input word is used to predict the context. [Source: NLP 101: Word2Vec]

15. **What are "contextualised embeddings"? Please refer to the picture in (11) above.**
In contextualized embeddings representation for each token is a function of the entire input sentence. The (11) the contextualized word embeddings are the hidden states h_i. Each word is interpreted in the context of all the previous words.

References