# Sentence Generation by Sequence to Sequence Model in Language Acquisition Environment

**Zahra Shekarchi F.**
Department of Computer Science
University of Toronto
Ontario, Canada
zahra@cs.toronto.edu

## Abstract

How children acquire language is still an open problem and has been studied for many years. Learning to generate sentences related to an event or a representation can be seen as a translation problem in which the semantic features in the scene should be translated to words in a meaningful utterance. This problem can also be compared with image captioning problem, in which instead of generating sequences (captions) for images, we want to generate sequences for observed semantic features in a scene. Recurrent Neural Networks have been successful in both the translation and image captioning problems, more specifically Sequence to Sequence (seq2seq) model is an effective solution for translation problem. Hence, in this work, we want to examine how a seq2seq model can learn to make sentences based on semantic features of a scene. One of the successful models in language acquisition is based on incremental and probabilistic learning mechanism, similar to IBM I. Our approach for translating semantic features to utterances, by seq2seq learning, could beat this model.

## 1 Introduction

How do people learn to communicate and how they comprehend language as a tool to do so? How do they link the structure of a sentence to its meaning? How do they correspond unknown semantic features to heard words? This is a problem which has been studied by people from linguistics, psychology, and computer science. The problem of how we make sentences for an observed event or scene can be interpreted as a translation problem from a source language to the destination language. The source language consists of some semantic features which are extracted from a scene, and the destination language includes meaningful sentences corresponding to the scene (figure 1). Another way to look at sentence generation problem is to compare it with popular image captioning problem, in which a model makes meaningful and related captions for given images. The difference is that in image captioning, we deal with image processing first, but in our problem the scene is represented in text (as a set of semantic features) instead of image segments.

We want to implement a model which can behave like children, learning a language, and learn to make sentences by perceiving some semantic features in a scene. There are different approaches to do so. One of these approaches is cross-situational learning which suggest that people perceive the regularities that occur in different situations and use this evidence to understand meanings and structures [1, 2]. This is explained in section 1.1. Another approach is based on neural networks which will be presented in section 1.2.

In section 2, we will show what our model is and how we make sentences in different models. For the data we have, there is a need to examine seq2seq model effectiveness. We will speak about experiments which have been done on the models and their results in section 3. Finally, in section 4 we would discuss about the results and some future works can be done to investigate into the proposed model.

### 1.1 Probabilistic Word Learning Model

The word learning model proposed by Fazly et al. (FAS) [1] is an incremental probabilistic learning model which makes a lexicon learned by processing pairs of utterances and related semantic features (figure 1). In each pair, it is not clear which semantic feature is associated to which word in the utterance.

Utterance: Joe is happily eating an apple
Scene: {joe, quickly, et, a, big, red, apple, hand}

*Figure 1 A sample utterance-scene pair*

Word meaning in this model is defined as a probabilistic association between a word form and a concept. These associations are learned based on a probabilistic learning, incrementally. The model keeps a meaning representation for each word as a probability distribution over all semantic features seen so far. The initial distributions are considered to be uniform, and by processing input pairs of utterance-scene the model updates these distributions and associations between every word and every semantic feature. Figure 2 shows alignments between words and semantic features in the utterance-scene pair of figure 1. The thickness of each line between a word and a semantic feature shows the strength of the alignment.

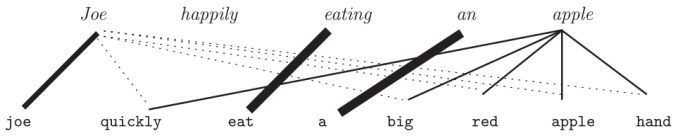


*Figure 2 Sample alignments between words and semantic featres in the corresponding scene representation; dashed lines represent very weak alignments. Image is from [1]*

#### 1.1.1 Input representation

The input to the model contains a sequence of utterance-scene pairs that correspond to a scene representation perceived by the child, and the utterance that describes it and heard by the child. Each utterance is represented as a set of words and the corresponding scene as a set of semantic features, like:

1. $U^{(t)}$: Joe is quickly rolling a ball
   $S^{(t)}$: {joe, happy, roll, a, red, ball, hand, mommy, talk}

$U^{(t)}$ shows the current utterance and $S^{(t)}$ shows the current scene.

#### 1.1.2 Calculating the alignment probabilities and word meanings

The likelihood of aligning a semantic feature (m) in the scene with a word (w) in the utterance is proportional to the meaning probability of the given feature for the word (p(m|w)).

$$a(w|m, \mathrm{U}^{(t)}, \mathrm{S}^{(t)}) = \frac{p^{(t-1)}(m|w)}{\sum_{w' \in \mathrm{U}^{(t)} \cup \{d\}} p^{(t-1)}(m|w')} \quad (1)$$

In equation (1), for smoothing, d is a dummy word that is added to the utterance. $P^{(t)}(m|w)$ is calculated in equation (3). Then we add the current alignment probabilities for word w and the semantic features m in $S^{(t)}$ to the accumulated evidence 'assoc' which is updated incrementally (with initial value of zero):

$$\mathrm{assoc}^{(t)}(w, m) = \mathrm{assoc}^{(t-1)}(w, m) + a(w|m, \mathrm{U}^{(t)}, \mathrm{S}^{(t)}) \quad (2)$$

In (3) the summation in the denominator is over all symbols encountered so far (to time t). β is an upper bound on the expected number of semantic features, and λ is a smoothing factor.

$$p^{(t)}(m|w) = \frac{\text{assoc}^{(t)}(m,w) + \lambda}{\sum_{m' \in \mathcal{M}} \text{assoc}^{(t)}(m',w) + \beta \times \lambda} \quad (3)$$

### 1.2 Recurrent Neural Network

Recurrent neural networks (RNNs) are a superset of feed-forward neural networks that capture time dynamics, and pass information across time steps. Unlike feed-forward neural networks, recurrent networks can process examples one at a time, retaining a state, or memory, that reflects an arbitrarily long context window. Recurrent neural networks have stemmed from both cognitive modeling and supervised learning. Neural networks are well-suited for machine perception tasks; however, despite their power, feed-forward neural networks rely on the assumption of independence among the data points; after each example is processed, the state of the network is forgotten. One of the cases this assumption is not true, is the words sequenced in a sentence in a natural language processing application. Thus, the idea of retaining a memory inside neurons in RNNs seems helpful now.
For a RNN, at time t, nodes receiving input along recurrent edges receive input activation from the current data point $x^{(t)}$ and from hidden nodes $c^{(t-1)}$ in the network's previous state. The output $h^{(t)}$ is computed given the hidden state $c^{(t)}$ at time step t. A simple recurrent neural network is shown in figure 3 (left side). The dynamics of this RNN can be seen by unfolding the network shown in figure 3 (right side). In this figure, we can see a deep neural network with one layer per time step and shared weights across time steps. So, the unfolded network can get trained across some time steps using back-propagation.

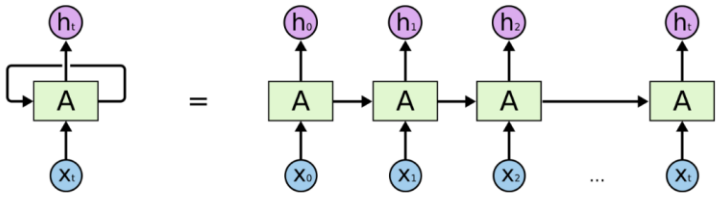


*Figure 3 An unrolled recurrent neural network – image from [9]*

Now we want to introduce two recently proposed recurrent units (LSTM and GRU) for sequence modeling.
The Long Short-Term Memory (LSTM) unit was proposed by Hochreiter and Schmidhuber [6]. Since then, a number of modifications have been made to the original design. We will follow the implementation of LSTM as used in Graves [7].
A single LSTM cell is depicted in figure 4. The computations related to different parts of the cell is given here:

$$i_t = \sigma\,(W_{xi}\,x_t + W_{hi}\,h_{t-1} + W_{ci}\,c_{t-1} + b_i),$$
$$f_t = \sigma\,(W_{xf}\,x_t + W_{hf}\,h_{t-1} + W_{cf}\,c_{t-1} + b_f),$$
$$c_t = f_t\,c_{t-1} + i_t \tanh\,(W_{xc}\,x_t + W_{hc}\,h_{t-1} + b_c),$$
$$o_t = \sigma\,(W_{xo}\,x_t + W_{ho}\,h_{t-1} + W_{co}\,h_{t-1} + b_o),$$
$$h_t = o_t \tanh(c_t).$$

Here, σ is the logistic sigmoid function. *i*, *f*, *o*, and *c* are the *input gate*, *forget gate*, *output gate*, *cell*, and *cell input* activation vectors (of the same size of the hidden vector *h*). $W_{hi}$ weight matrix is the hidden-input gate matrix, $W_{xo}$ is the input-output gate matrix, and so on. The weight matrices from the cell to gate vectors (for example $W_{ci}$) are diagonal, so element m in each gate vector only receives input from element m of the cell vector. For optimization, we have used gradient descent optimizer built in Tensorflow [11].
LSTM unit can detect an important feature from an input sequence at early stages, and carry the information over the long distance, maintaining long-distance dependencies. Meanwhile, they can ensure that the gradient can pass across many time steps without vanishing or exploding [4].

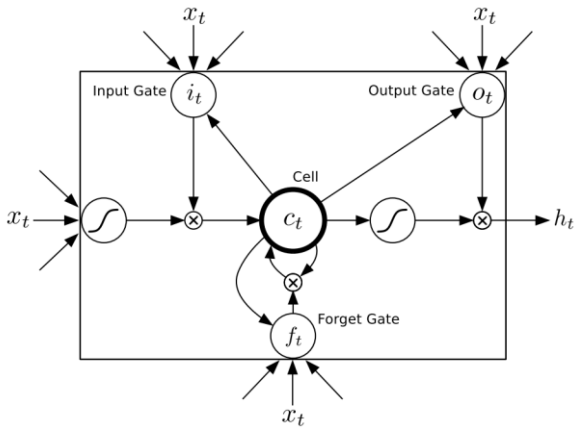


*Figure 4 Long Short-Term Memory cell – image from [7]*

A Gated Recurrent Unit (GRU) was introduced in 2014, to make each recurrent unit capture dependencies of different time scales. Similar to the LSTM, the GRU has gates that retain the flow of information inside the unit, without having a memory cell separately [5]. Thus, these units have less parameters to be learned. In [5], it is found that GRUs are comparable to LSTM units.

### 1.2.1 Sequence to Sequence Model

Despite the flexibility and power of Deep Neural Networks (DNNs), they can only be used for problems that input and output encoded with vectors of fixed dimension. However, this condition is not true in many applications; such as machine translation. In Sutskever et al. [3], they solved general sequence to sequence problem by LSTM architecture [6]. The idea is to use one LSTM for reading input sequence, one step at a time, to obtain large fixed dimensional vector representation, and use another LSTM to extract the output sequence from the vector. An application of their model for neural machine translation is depicted in figure 5. In figure 5, the system first read the source sentence using an encoder (the blue network on left) to build a 'thought' vector, a sequence of numbers that represents the sentence meaning. There is a decoder which processes the sentence vector to generate a translation (the red network on right). This 'encoder-decoder' architecture can capture distant dependencies in languages. For example, gender agreements or syntax structures.

For different problems, different architectures can be applied. The number of layers, number of neurons, directionality (unidirectional or bidirectional), and cell types which can be LSTM, GRU, etc. are existing options for making an appropriate architecture.

As we can see in figure 5, we have both source sentence and target sentence given to the model. However, for test phase, we want to generate a meaningful fluent translation. So what can we do for decoder input? One idea to solve this, is shown in figure 6.

The source sentence encoding is the same as training. As soon as the decoder receives "<s>" the decoding is started. Then, one step at a time, in the decoder, we choose the most likely word (the word associated with the maximum predicted value) as a translated word and then feed it back as input to the next time step. As we choose the most likely word in each time step, we call it 'greedy' decoding. There are other ways to do this; like sampling or beam-search.

## 2 The model

In this section we want to use models mentioned in section 1 for the task of translating a set of semantic features to a meaningful sentence. For example, for the given set of semantic features like *{joe, happy, roll, a, red, ball, hand, mommy, talk}*, we expect to get a sentence like '*Joe is quickly rolling a ball*'.

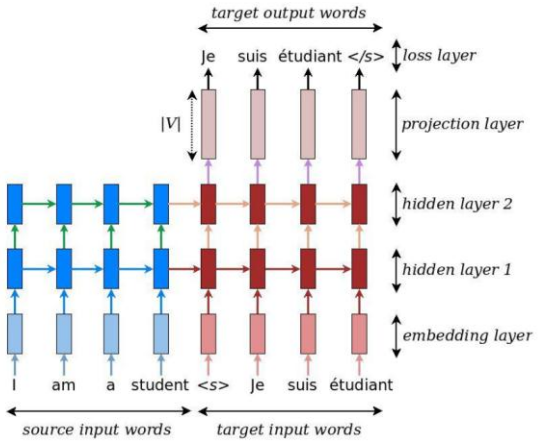


*Figure 5 A deep recurrent architecture for translating a source sentence "I am a student" into a target sentence "Je suis étudiant". "<s>" marks the start of the decoding process while "</s>" tells the decoder to stop. Image from [10]*

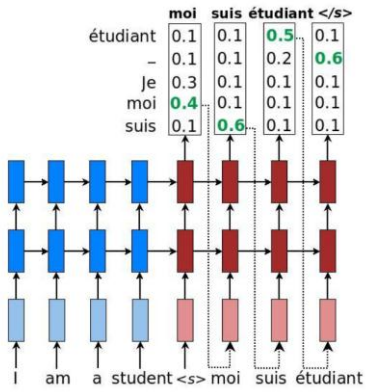


*Figure 6 Greedy decoding – example of how a trained model produces a translation for a source sentence "Je suis étudiant" using greedy search. Image from [10]*

## 2.1 Translate by Probabilistic Learning Model (FAS)

After learning alignment probabilities between words and semantic features, we have a lexicon which keeps all these probabilities. For translating a given set of semantic features like {*joe, happy, roll, a, red, ball, hand, mommy, talk*}, for each semantic feature, we use the most probable word; a greedy choice which has been used in many translation tasks. For instance, for feature 'joe' the model learned 'Joe' has the highest alignment probability score among other words.

## 2.2 Non-Sequence to Sequence

For training seq2seq model, the input is a sequence of semantic-features–utterance pairs. The set of semantic features (which does not keep order) is given to encoder input and the utterance words (which have orders) are given to the decoder input. At test time, for each test input we give the set of semantic features to the encoder input and get the predicted sentence from the decoder output, in the way explained in section 1.2.1.

## 2.3 Sequence to Sequence

In this section, we want to see the behavior of the seq2seq model in face of ordered semantic features instead of not-ordered ones. For training seq2seq in this case, the input is a sequence of semantic-features – utterance pairs. The difference between this case and the one in previous section is that the semantic features have order (similar to the words in the utterance). This order in semantic features can be interpreted as the order in which the child pays attention to the concepts and features in the

scene. The ordered semantic features are given to the encoder input and the corresponding utterance words are given to the decoder input. For testing the model in this case, for each test input, we give the ordered semantic features to the encoder input and expect the predicted utterance from the decoder output.

# 3 Experiments

In this section, we want to speak about the different experiments we have done to examine the behaviors of the models introduced in section 2. Also, we will talk more about our evaluation and the data we use in our experiments and details about its quality and quantity.

## 3.1 Dataset Details

The corpus from CHILDES [12] contains 120000 utterance-scene pairs in form represented in figure 1. The vocabulary size (for words in utterances) is 3837 and the number of semantic features in scenes are 6227. For final comparison we use 96000 pairs for training and 23000 pairs for testing the models. For first 20000 pairs in the corpus, the number of words and semantic features are 1776 and 3446 (respectively). For the first 40000 pairs, the number of words and semantic features are 2368 and 4298 (respectively). The semantic features related to each scene are ordered in the input data. Whenever we want to eliminate this order we have to shuffle these semantic features to represent them as an unordered set.

## 3.2 Evaluation Metrics

Bilingual Evaluation understudy (BLEU) is a measure for evaluating the quality of translated text from one language to another one [8]. The idea behind their measure is "the closer a machine translation is to a professional human translation, the better it is."
The general idea of this measure is to use combined precisions of common n-grams between a reference sentence and a candidate sentence. The factor of n-gram is a continuous sequence of n words of a sequence of text. A simple example of these precisions is shown in table 1 for following reference sentence and candidate sentences:

*Sample 1:Reference: Joe is happily eating an apple*
*Candidate 1: Joe is eat fruit happy happy*
*Candidate 2: an apple Joe is happily eating*
*2-gram 4-gram*

We can combine these precisions with different weights and add some penalty factors for too short sentences. Papineni et al. [8] claimed that their measure is correlated to human judgment if we compute precision for n-grams of n = 1 to 4.

*Table 1 example of 1-4-gram precisions for reference and candidates in sample 1*

| Metric | Candidate 1 | candidate 2 |
|---|---|---|
| Precision 1-gram | 2/6 | 6/6 |
| Precision 2-gram | 1/5 | 4/5 |
| Precision 3-gram | 0/4 | 2/4 |
| Precision 4-gram | 0/3 | 1/3 |

## 3.3 Experimental Results

### 3.1.1 sentence generation by FAS model on CHILDES data

In section 2.1 we mentioned how we translate semantic features to utterances by using a lexicon learned in FAS model. There is a point to say that the only thing we learn in FAS is a lexicon which keeps the alignment probabilities between words and semantic features. To generate sequences as utterances, we need to feed sequential data for the translation. For example, if the raw data is like this:

*Sample 2:SENTENCE: you are not watching that*

*SEM FEATURES: you, pronoun, 2nd_person; stative, be; unary, not, negative, conjunction; watch, search, perceive; that, singular, determiner;*

We will keep the order in semantic features and use the greedy way for selecting the appropriate word to put in the candidate utterance or sentence. The BLEU score for 23000 test pairs, after training over 96000 pairs is **0.26**.

### 3.1.2 Developing Sequence to Sequence Model on CHILDES data

In this section, we report the result of different experiments have been done on the seq2seq model described in section 2.2 and 2.3 (for finding the best architecture).
In table 2, we can see that 1 layer of 1024 neurons is the best fit for our problem. In table 3, we are showing the BLEU score for different values of learning rate. So, for next experiments, learning rate = 0.4 is used. From table 4, the same result can be seen for larger input data. The positive effect of number of iterations can be seen in table 5. Hence for next experiments, number of iterations = 1500 is used. In table 6, different neural architectures have been examined and we can see that increasing the number of layers cannot be helpful for our purpose.

*Table 2 BLEU score for 20000 train pairs, 2000 test pairs, learning rate=0.3, and number of iterations=100*

| Number of layers | Number of cells in each layer | Average BLEU score |
|---|---|---|
| 1 | 512 | 0.21 |
| **1** | **1024** | **0.23** |
| 2 | 512 | 0.19 |

*Table 3 BLEU score for 20000 train pairs, 2000 test pairs, 1 layer of 1024 neurons, and number of iterations=200*

| Learning rate | Average BLEU score |
|---|---|
| 0.1 | 0.22 |
| 0.2 | 0.25 |
| 0.3 | 0.29 |
| **0.4** | **0.34** |
| 0.5 | 0.27 |

*Table 4 BLEU score for 32000 train pairs and 4000 test pairs, 1 layer of 1024 neurons, and number of iterations=500*

| Learning rate | Average BLEU score |
|---|---|
| 0.3 | 0.39 |
| 0.4 | 0.40 |

*Table 5 BLEU score for 96000 train pairs and 23000 test pairs, 1 layer of 1024 neurons, and learning rate=0.4*

| Number of iterations | Average BLEU score |
|---|---|
| 500 | 0.35 |
| **1500** | **0.44** |

*Table 6 BLEU score for 96000 train pairs and 23000 test pairs, learning rate=0.4, and number of iterations=1500*

| Number of layers | Number of cells in each layer | Average BLEU score |
|---|---|---|
| **1** | **1024** | **0.44** |
| 2 | 512 | 0.40 |
| 2 | 256 | 0.39 |

*Table 7 BLEU score for 96000 train pairs and 23000 test pairs, 1 layer of 1024 neurons, learning rate=0.4, and number of iterations=1500*

| Cell type | Encoder input type | Average BLEU score |
|---|---|---|
| LSTM | Sequence | 0.37 |
| **LSTM** | **Non-sequence** | **0.30** |
| GRU | Sequence | 0.44 |
| GRU | Non-sequence | 0.28 |

In table 7, we are examining two points. First is the type of neuron cells we can use in seq2seq model, and second the type of input data to the encoder module. As FAS model does not consider sequence in the input data, it is not fair to compare it with seq2seq model (experiments explained so far). So we shuffle input data given to the encoder and rerun the seq2seq learning (actually non-seq2seq). At this point, we can say that the non-seq2seq model (table 7, 2nd row) can beat the baseline 0.26 BLEU score of FAS translation model, which translate semantic features to meaningful utterances. The second point related to cell type, notices that using LSTM on not sequenced input data increases the BLEU score, but for sequenced data, it is much better to use GRU cells.

## 4 Discussion

There are some points to be mentioned now. First is that seq2seq and non-seq2seq could generate sentences of higher BLEU score, in comparison with the FAS learning model, though their model is a successful cognitive one. The measure we used here is BLEU score, however there are other translation quality measurements which should be considered. Because, in BLEU score, some points are missed; like synonyms. It means BLEU penalizes the output sentence if it is "this is beautiful" instead of "this is pretty", so we need a measure to handle this issue. Another point to be mentioned is that, FAS learning model is much faster than seq2seq/non-seq2seq model (FAS model takes about 2 hours to be trained and some minutes to be tested, but seq2seq /non-seq2seq takes 10-12 hours to be trained and around 2 hours to be tested, on the same data size). Hence, if time matters, maybe FAS model is a better choice. It is also important that LSTM based network needs more time to be trained compared to GRU based one. Finally, as non-seq2seq model could outperform FAS model, and since FAS model could be successful in presenting cognitive behaviors, we can investigate on non-seq2seq to find out whether cognitive behaviors of children, during language acquisition, can be observed or not. Furthermore, we can try to make a lexicon by using this model; we can give one semantic feature each time to the encoder input and predict related words from decoder output.

### References


[1] Afsaneh Fazly, Afra Alishahi, Suzanne Stevenson, "A Probabilistic Computational Model of CrossSituational Word Learning," Cognitive Science 34 (6), 1017-1063.

[2] Aida Nematzadeh, Barend Beekhuizen, Shanshan Huang, Suzanne Stevenson, "Calculating Probabilities Simplifies Word Learning," Cognitive Science Feb. 2017, pp. 853-858.

[3] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, "Sequence to Sequence Learning with Neural Networks," NIPS'14 Proceeding of the 27th International Conference on Neural Information Processing Systems – Volume 2, Pages 3104-3112.

[4] Zachary C. Lipton, John Berkowitz, Charles Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," Computing Research Repository (CoRR), June 5th, 2015.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," NIPS 2014 Workshop on Deep Learning, December 2014.

[6] Sepp Hochreiter, Jürgen Schmidhuber, "Long Short-Term Memory," Neural Comput. 1997 Nov 15; 9(8), pp. 1735-80.

[7] Alex Graves, "Generating Sequences With Recurret Neural Networks," arXiv preprint arXiv:1308.0850, 4 Aug 2013.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," Association for Computational Linguistics (ACL), July 2002, pp. 311-318.

[9] Christopher Olah, "Understanding LSTM Networks," August 27, 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed: 20- Dec- 2017].

[10] Thang Luong, Eugene Brevdo, Rui Zhao, "Neural Machine Translation (seq2seq) Tutorial," Dec. 2017. [Online]. Available: https://github.com/tensorflow/nmt [Accessed: 20- Dec- 2017].

[11] "TensorFlow", open-source software library. [Online]. Available: https://www.tensorflow.org/ [Accessed: 20- Dec- 2017].

[12] MacWinney, B. (2000). "The CHILDES Project: Tools for Analyzing talk," volume 2: The Database (3rd Ed.).