# MT Interim Report: Neural Word Alignemnt


Bailey Parker
Johns Hopkins University
bailey@jhu.edu

Vivian Tsai
Johns Hopkins University
viv@jhu.edu

William Watson
Johns Hopkins University
billwatson@jhu.edu



**Abstract**


# 1 Introduction

# 2 Data Procurement and Processing

## 2.1 Symbol Tokenization

## 2.2 Number Tokenization

## 2.3 Proper Noun Tokenization

## 2.4 Lemmatization Techniques

## 2.5 POS Tagging

# 3 Model Components

## 3.1 Preliminary Notation

We begin by stating several operations frequently used in our discussion.

### 3.1.1 Hadamard Product

To perform element-wise multiplication of two matricies $A$ and $B$, of equivalent dimensions, we use the hadamard product, defined as the $\circ$.

$$(A \circ B)_{ij} = A_{ij} \cdot B_{ij} \tag{1}$$

### 3.1.2 Sigmoid Function

The sigmoid function $\sigma$ is applied element wise and defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

### 3.1.3 Hyperbolic Tangent Function

The hyperbolic tangent function tanh is defined as follows:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3}$$

and is applied element-wise.

### 3.1.4 Variables

We define our source sequence as $s$, target sequence as $t$, and refer to the $i$-th source token as $s_i$. We refer to the $j$-th target token as $t_j$. We refer to a matrix $\psi$ whose rows represent values related to source tokens $s_i$ and whose columns refer to values realted to target tokens $t_j$. Hence, $\psi$ is an $|s| \times |t|$ sized matrix.

## 3.2 Softmax and Log Softmax

The softmax function transforms a vector of values into a probability distirbution. Appliying the softmax function to an n-dimensional input tensor rescales it so that the elements of the n-dimensional output tensor lie in the range (0,1) and sum to 1.

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4}$$

For a matrix $\psi$, the rows correspond to source words $s_i$ and the columns correspond to target words $t_j$. In addition, when we softmax with respect to the targets, i.e. $\sigma_t(\psi)$, the softmax is applied per column. If applied per row, we denote this as $\sigma_s(\psi)$.

$$\sigma_t(\psi)_{ij} = \frac{e^{\psi_{ij}}}{\sum_k e^{\psi_{kj}}} \tag{5}$$

$$\sigma_s(\psi)_{ij} = \frac{e^{\psi_{ij}}}{\sum_k e^{\psi_{ik}}} \tag{6}$$

However, it is sometimes better to work in log-space, and use the log softmax operator for numerical stability.

$$\log \sigma_t(\psi)_{ij} = \log \frac{e^{\psi_{ij}}}{\sum_k e^{\psi_{kj}}} \tag{7}$$

$$\log \sigma_s(\psi)_{ij} = \log \frac{e^{\psi_{ij}}}{\sum_k e^{\psi_{ik}}} \tag{8}$$

### 3.3 Word Embeddings

Word Embedding layers allow for a simple lookup table that stores embeddings of a fixed dictionary and size. More specifically, these layers are often used to store word embeddings and retrieve them using indices. The input to the embedding layer is a list of indices, and the output is the corresponding word embeddings. This allows words to be represented numerically to a set embedding dimension size, and thus can be passed onto later layers.

Word Embeddings can be formulated as a weight matrix $W_e$, where the vector representation of word $w_i$ is the $i$-th row of the matrix, and can be represented as $W_e[w_i]$.

$$W_e = \begin{bmatrix} \longleftarrow w_1 \longrightarrow \\ \vdots \\ \longleftarrow w_i \longrightarrow \\ \vdots \\ \longleftarrow w_n \longrightarrow \end{bmatrix} \tag{9}$$

### 3.4 Gated Recurrent Units (GRU)

Gated Recurrent Units are a more complex formulation to recurrent layers to process sequences, and improve the vanishing gradient problem found in vanilla RNN layers while using less parameters than a Long Short-Term Memory (LSTM) layer[1]. A GRU makes use of 3 gates: $r_t$ reset gate; $z_t$ update gate; $n_t$ new gate. These 3 gates allow for a blending of the hidden state with new input, and are computed as follows for each input $x_t$ in a sequence $\mathbf{x}$:

$$\begin{aligned} r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \\ z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \\ n_t &= \tanh(W_{in}x_t + b_{in} + r_t \circ (W_{hn}h_{t-1} + b_{hn})) \\ h_t &= (1 - z_t) \circ n_t + z_t \circ h_{t-1} \end{aligned} \tag{10}$$

where $x_t$ is input at time $t$, $h_{t-1}$ is the hidden state of the previous layer at time $t-1$ or the initial hidden state at time 0, $h_t$ is the new hidden state at time $t$, and $\sigma$ is the sigmoid function.

### 3.5 Bidirectional GRU

### 3.6 Alignment Prior

### 3.7 Batch Matrix Multiplication (BMM)

## 4 Model Descriptions

### 4.1 Dot Aligner

### 4.2 Bidirectional GRU Aligner

### 4.3 Extensions

## 5 Loss Function

### 5.1 Supervised Loss

### 5.2 Unsupervised Alignment Loss

## 6 Ground Truth Alignment Results?

## 7 Current Status

## 8 Future Work

## References

[1] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.