An Introductory Survey on Attention Mechanisms in NLP Problems

Dichao Hu
College of Computing, Georgia Institute of Technology
801 Atlantic Dr NW
Atlanta, Georgia 30332

Abstract

First derived from human intuition, later adapted to machine translation for automatic token alignment, attention mechanism, a simple method that can be used for encoding sequence data based on the importance score each element is assigned, has been widely applied to and attained significant improvement in various tasks in natural language processing, including sentiment classification, text summarization, question answering, dependency parsing, etc. In this paper, we survey through recent works and conduct an introductory summary of the attention mechanism in different NLP problems, aiming to provide our readers with basic knowledge on this widely used method, discuss its different variants for different tasks, explore its association with other techniques in machine learning, and examine methods for evaluating its performance.

1 Introduction

We introduce our main topic via a concrete example of neural machine translation. Traditional methods (Kalchbrenner and Blunsom 2013; Cho et al. 2014) are formulated by an encoder-decoder architecture, both of which are recurrent neural networks. An input sequence of source tokens is first fed into the encoder, of which the last hidden representation is extracted and used to initialize the hidden state of the decoder, and then target tokens are generated one after another. Despite achieving higher performance compared to purely statistical methods, the RNN-based architecture suffers from two serious drawbacks. First, RNN is forgetful, meaning that old information gets washed out after being propagated over multiple time steps. Second, there is no explicit word alignment during decoding and therefore focus is scattered across the entire sequence. Aiming to resolve the issues above, attention mechanism was first introduced into neural machine translation (Bahdanau, Cho, and Bengio 2014). They maintain the same RNN encoder, for each step j during decoding they compute an attention score α_{ji} for hidden representation $\boldsymbol{h_i^{in}}$ of each input token to obtain a context vector $\boldsymbol{c_j}$ (see Figure 1):

$$e_{ji} = a(\boldsymbol{h_i^{in}}, \boldsymbol{h_j^{out}}) \tag{1}$$

$$\alpha_{ji} = \frac{e_{ji}}{\sum_i e_{ji}} \tag{2}$$

$$\boldsymbol{c_j} = \sum_i \alpha_{ji} \boldsymbol{h_i^{in}} \tag{3}$$

Here $\boldsymbol{c_j}$, a weighted average of elements in the input sequence, is the encoded sentence representation with respect to the current element $\boldsymbol{h_j^{out}}$, and $a(\boldsymbol{h_i^{in}}, \boldsymbol{h_j^{out}})$ is the alignment function that measures similarity between two tokens and will be discussed in detail later. Then $\boldsymbol{c_j}$ is combined with the current hidden state $\boldsymbol{h_j}$ and the last target token $\boldsymbol{y_{j-1}}$ to generate the current token $\boldsymbol{y_j}$:

$$\boldsymbol{y_j} = f_y(\boldsymbol{h_j^{out}}, \boldsymbol{y_{j-1}}, \boldsymbol{c_j}) \tag{4}$$

$$\boldsymbol{h_{j+1}^{out}} = f_h(\boldsymbol{h_j^{out}}, \boldsymbol{y_j}) \tag{5}$$

Here f_y and f_h stands for the output layer and hidden layer in recurrent networks. This procedure is repeated for each token $\boldsymbol{y_j}$ until the end of the output sequence. By introducing this additional encoding step, problems mentioned earlier can be tackled: the bad memory of RNN is no longer an issue, since the computation of attention score is performed on each element in the input sequence and therefore computation of the encoded representation $\boldsymbol{c_j}$ is unaffected by the sequence length; on the other hand, soft alignment across the input sequence can be achieved since each element is either highlighted or down-weighted based on its attention score and focus is paid only to the important parts in the sequence, discarding useless or irrelevant parts. As a result, the attention-based translation model achieved great success in machine translation and then attention mechanism got widely applied to other NLP tasks. Moreover, different variants of the basic form of attention have been proposed to handle more complex tasks. As an overview of the following sections, we will:

1. Explain the basic form of attention in detail. (Formulation)
2. Discuss different variants based on the special task they are dealing with. (Variation)
3. Explore how attention is associated with other concepts or techniques in machine learning, such as pre-training and ensemble. (Application)

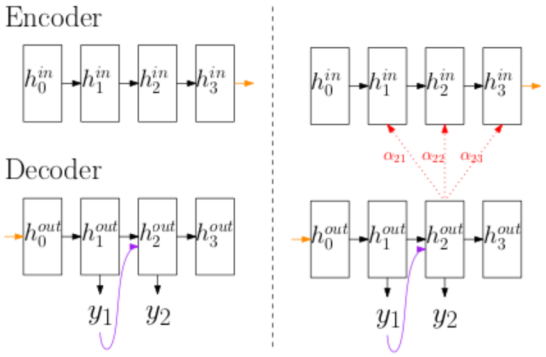

Figure 1: A comparison between the traditional encoder-decoder architecture (left) and the attention-based architecture (right). During decoding, an additional attention score α_{ji} is computed for each source token $\boldsymbol{h_i^{in}}$ with respect to the target token $\boldsymbol{h_j^{out}}$, and the scores are then used to obtain the contextual encoding $\boldsymbol{c_j}$.

4. Examine methods for evaluating the performance of attention. (Evaluation)

2 Formulation

Previously, we discussed the application of attention in neural machine translation. To formally generalize it into the basic form of attention, we define $V = \{\boldsymbol{v_i}\} \in \mathbb{R}^{n \times d_v}$ as a sequence of vector elements and rewrite the previous steps as:

$$
\begin{aligned}
e_i &= a(\boldsymbol{u}, \boldsymbol{v_i}) && \text{(compute attention scores)} \\
\alpha_i &= \frac{e_i}{\sum_i e_i} && \text{(normalize)} \\
c &= \sum_i \alpha_i \boldsymbol{v_i} && \text{(encode)}
\end{aligned}
$$

In the first step, $\boldsymbol{u} \in \mathbb{R}^{d_u}$ is a task-specific pattern vector to match with each element in the sequence $\{\boldsymbol{v_i}\}$ based on the alignment function $a(\boldsymbol{u}, \boldsymbol{v})$ that outputs a scalar score $e_i \in \mathbb{R}$ to indicate quality of match. In most cases we assume $d_u = d_v = d$. Common choices (Luong, Pham, and Manning 2015) are given by:
Multiplicative:

$$\boldsymbol{u^T v} \quad \text{(dot)} \tag{6}$$

$$\boldsymbol{u^T} W \boldsymbol{v} \quad \text{(general)} \tag{7}$$

Additive:

$$\boldsymbol{w_2^T} tanh(W_1[\boldsymbol{u}; \boldsymbol{v}]) \tag{8}$$

MLP:

$$\sigma(\boldsymbol{w_2^T} tanh(W_1[\boldsymbol{u}; \boldsymbol{v}] + \boldsymbol{b_1}) + b_2) \tag{9}$$

all of which measures pairwise similarity in different representations. The final score $\alpha_i \in \mathbb{R}$ is the normalized weight for each element $\boldsymbol{v_i}$ and then used to encode the entire sequence into a context vector $\boldsymbol{c} \in \mathbb{R}^{d_u}$, which is later incorporated into a downstream task as an additional contextual feature. Intuitively, $\boldsymbol{v_i}$ that closely matches the pattern $\boldsymbol{u}$ receives a large weight and therefore dominates the final encoding $\boldsymbol{c}$. In machine translation, the attention score can be naturally interpreted as the alignment measure of a target token to each source token. Ideally, when generating a target token $\boldsymbol{y_j}$, we expect the gold alignment source token(s) to receive high attention score and therefore the encoded representation $\boldsymbol{c_j}$ can provide closely relevant information for decoding.
To avoid confusion, we explain our use of notation and abbreviation before our further discussion. We use lower-case symbols for scalars, lower-case boldface symbols for vectors, upper-case symbols for matrices, and upper-case boldface symbols for tensors. When we refer to attention scores, e stands for original scores and α stands for normalized scores. W, b are by default weights and biases to be learned during training, and $\boldsymbol{v_i}$ is by default a general element that can refer to either one of the following depend on the scenario: an embedded token within a sentence, a hidden representation of the token, an embedded sentence within a document, etc. For simplicity, we occasionally use $softmax$ to represent the normalization step that involves exponentiation, summation and division.

3 Variation

Previously, we discussed basic form of the attention mechanism. Because of its simplicity and interpretability, it is widely used in various NLP tasks. Nevertheless, such attention mechanism is in general not powerful enough for more complicated tasks. Here is a simple example (Sukhbaatar et al. 2015):

Sam walks into the kitchen. (1)
Sam picks up an apple. (2)
Sam walks into the bedroom. (3)
Sam drops the apple. (4)
Q: Where is the apple?
A: Bedroom

The underlying difficulty is that there is no direct relationship between Q and (3), and therefore we need to design a more sophisticated mechanism to guide attention to the correct location using latent clues within the context (temporal reasoning in this case). As different variants of attention have been proposed in recent years, we summarize them into several most representative categories (see Table 1): basic attention, multi-dimensional attention, hierarchical attention, self-attention, memory-based attention and task-specific attention. From left to right, the corresponding task increases in complexity, and the mechanism becomes more task specific. We will discuss each category in detail in the following sections.

3.1 Multi-dimensional Attention

The basic form of attention mechanism computes a scalar score α_i for each term in a sequence $V = \{\boldsymbol{v_i}\}$. This could be named as 1D attention or vector attention because the

Type of Attention	Purpose in brief
Basic Attention	Extracting important elements from a sequence
Multi-dimensional Attention	Capturing multiple types of interaction between terms
Hierarchical Attention	Extracting globally and locally important information
Self-Attention	Capturing deep contextual information within a sentence
Memory-based Attention	Discovering latent dependencies in sophisticated NLP tasks
Task-specific Attention	Capturing important information specified by the task

Table 1: An overview of each type of attention and its corresponding purpose

1D Attention	2D Attention
$\boldsymbol{u^T W v}$ $(W \in \mathbb{R}^{d\times d})$	$\boldsymbol{u^T W v}$ $(\boldsymbol{W} \in \mathbb{R}^{k\times d\times d})$
$\boldsymbol{w_2^T} tanh(W_1[\boldsymbol{u};\boldsymbol{v}])$ $(\boldsymbol{w_2} \in \mathbb{R}^e)$	$W_2^T tanh(W_1[\boldsymbol{u};\boldsymbol{v}])$ $(W_2 \in \mathbb{R}^{e\times k})$
$\sigma(\boldsymbol{w_2^T} tanh(W_1[\boldsymbol{u};\boldsymbol{v}] + \boldsymbol{b_1}) + b_2)$	$\sigma(W_2^T tanh(W_1[\boldsymbol{u};\boldsymbol{v}] + \boldsymbol{b_1}) + \boldsymbol{b_2})$

Table 2: Some examples of extending from 1D attention to 2D, assuming $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d, W_1 \in \mathbb{R}^{e\times 2d}$

concatenated output scores $\boldsymbol{\alpha} = \{\alpha_i\}$ is a vector $\boldsymbol{\alpha} \in R^n$. The motivation for multi-dimensional attention is simply to capture multiple interaction between terms in different representation space, which can be easily constructed by directly stacking together multiple single dimensional representations.

As an example of 2D attention in aspect and opinion terms extraction (Wang et al. 2017), given a sequence of hidden representation of tokens $V = \{\boldsymbol{v_i}\}$, an aspect prototype vector $\boldsymbol{u}$, and a 3D tensor $\boldsymbol{W} = \{W_k\} \in \mathbb{R}^{K\times d\times d}$ where each slice $W_k \in \mathbb{R}^{d\times d}$ is a matrix that captures one type of composition between each token and the prototype, the vector attention score $\boldsymbol{e_i}$ (not normalized) for each element $\boldsymbol{v_i}$ is given by:

$$\boldsymbol{e_i} = tanh(\boldsymbol{u^T W v_i}) = concat(tanh(\boldsymbol{u^T} W_k \boldsymbol{v_i})) \quad (10)$$

Then the 2D attention representation E for the entire sequence is obtained by concatenation: $E = \{\boldsymbol{e_i}\}$. As a concrete example, consider the sentence "*Fish burger is the best dish*", here *Fish burger* is an aspect term and therefore will receive high attention score with respect to the aspect prototype $\boldsymbol{u}$, and the learned $\boldsymbol{W}$ should be able to highlight both *Fish* and *burger* after they are projected to different spaces. A drawback of this multi-dimension approach is that a strongly indicative element can capture multiple types of attention at the same time and therefore reduces its representation power. To compensate for this, we can impose a regularization penalty in Frobenius norm on the 2D attention matrix A (Du et al. 2018; Lin et al. 2017):

$$||AA^T - I||_F^2 \quad (11)$$

to constrain each attention column to focus on different parts in the sequence.

3.2 Hierarchical Attention

Let us begin our discussion on hierarchical attention by looking at an example in document classification. Given a short document (Yang et al. 2016): *How do I get rid of all the old* ***web searches*** *I have on my* ***web browser****? I want to clean up my* ***web browser*** *go to tools → options Then click "delete history" and "clean up temporary internet files."*, the task is to classify the document into one of several categories, which in this case is *Computer and Internet*. Intuitively, we can identify words that provide clues for classification, since the term ***web browser*** can appear frequently in computer-related documents. Moreover, typical sentences can be potentially informative for classification as well, as non-professionals using a computer software for the first time tend to seek instruction on *how to get rid of ...* If we think of the nested structure of textual data: $character \in word \in sentence \in document$, a hierarchical attention can be constructed accordingly, either bottom-up (i.e, word-level to sentence-level) or top-down (word-level to character-level) to identify clues or extract important information both globally and locally.

Bottom-up construction has been used in document classification (Yang et al. 2016). Two BiGRUs are applied to generate a word-level and a sentence-level contextual representation, respectively. Then a pair of hierarchical attention layers are applied to obtain a word-level and sentence-level encoding:

$$\boldsymbol{h_i^{(t)}} = BiGRU(\boldsymbol{v_i^{(t)}}) \quad (12)$$

$$\boldsymbol{v_i} = \sum_t softmax(\boldsymbol{u_w^T h_i^{(t)}}) \cdot \boldsymbol{h_i^{(t)}} \quad (13)$$

$$\boldsymbol{h_i} = BiGRU(\boldsymbol{v_i}) \quad (14)$$

$$\boldsymbol{c} = \sum_i softmax(\boldsymbol{u_s^T h_i}) \cdot \boldsymbol{h_i} \quad (15)$$

$softmax$ is equivalent to the normlization step we previously discussed. $\boldsymbol{h_i^{(t)}}$ and $\boldsymbol{h_i}$ stand for hidden representation for words and sentences. $\boldsymbol{u_w^T}$ and $\boldsymbol{u_s^T}$ are word-level and sentence-level pattern vectors to be learned during training. The final sentence-level representation $\boldsymbol{c}$ is then fed into a logistic regression layer to predict the category.

Another type of hierarchical attention takes a top-down approach, an example of which is for grammatical error correction (Ji et al. 2017). Consider a corrupted sentence: *I have no enough previleges*. The idea is to first bulid an encoder-decoder architecture similar to the one for machine translation, then apply a word-level attention for global grammar and fluency error correction (*I have no enough → I don't have enough*), and optionally a character-level attention for local spelling error correction *(previleges → privileges)*. Top-down techniques are also used in album summarization (Yu, Bansal, and Berg 2017), where a photo-level attention is used to select appropriate photos from an album and a word-level attention is integrated with a sequence model for text generation.

3.3 Self Attention

Let us revisit the steps for constructing the basic form of attention. Given a sequence of elements $V = \{\boldsymbol{v_i}\}$ and a pattern vector $\boldsymbol{u}$, for each element $\boldsymbol{v_i}$ we can compute the attention score $\alpha_i = a(\boldsymbol{u}, \boldsymbol{v_i})$. This can also be termed as external attention, since attention is computed by matching an external pattern $\boldsymbol{u}$ with each element $\boldsymbol{v_i}$, and each score e_i indicates quality of match. On the contrary, in self-attention, the external pattern $\boldsymbol{u}$ is replaced by parts of the sequence itself, and therefore is also termed as internal attention. To illustrate this with an example: *Volleyball match is*

in progress between ladies, here *match* is the sentence head on which all other tokens depend, and ideally we want to use self-attention to capture such intrinsic dependency automatically. Alternatively, we can interpret self-attention as matching with each element $\boldsymbol{v_i}$ an internal pattern $\boldsymbol{v'}$ within V:

$$e_i = a(\boldsymbol{v'}, \boldsymbol{v_i}) \tag{16}$$

A typical choice for $\boldsymbol{v'}$ is simply another element $\boldsymbol{v_j}$, so as to compute a pairwise attention score, but in order to fully capture complex interaction between terms within a sequence, we can further extend this to compute attention between every pair of terms within a sequence, i.e., to set $\boldsymbol{v'}$ as each element $\boldsymbol{v_j}$ in a sequence and compute a score for each pair of terms. Therefore we modify the previous equations to:

$$e_{ij} = a(\boldsymbol{v_i}, \boldsymbol{v_j}) \tag{17}$$

$$\alpha_{ij} = softmax(e_{ij}) \tag{18}$$

aiming to capture complex interaction and dependency between terms within a sequence. Then the choice of the alignment function a is literally the same as the basic attention, such as a single layer neural network:

$$\alpha_{ij} = softmax(tanh(\boldsymbol{w^T}[\boldsymbol{v_i}; \boldsymbol{v_j}] + b)) \tag{19}$$

In this way, each token maintains a distributed relation representation with respect to all other tokens and the complex pairwise relationship can be easily interpreted from its assigned scores. And the model can further be enriched with multi-dimensional attention (Shen et al. 2017) as we mentioned earlier.

Another motivation for self-attention is related to the word embedding. To be specific, we want to utilize self-attention models to learn complex contextual token representation in a self-adaptive manner. We can illustrate this point by an example of word sense disambiguation:

I arrived at the bank after crossing the street.

I arrived at the bank after crossing the river.

The word *bank* has different meanings under different contexts, and we want our model to learn contextual token embeddings that can capture semantic information from their surrounding contexts. Transformer (Vaswani et al. 2017) is an exemplar novel attention-based architecture for machine translation. It is a hybrid neural network with sequential blocks of feed forward layers and self-attention layers. Similar to the previous self-attention mode, the novel self-attentive encoding can be expressed as:

$$A = softmax[\frac{(VW_1)(VW_2)^T}{\sqrt{d_{out}}}] \tag{20}$$

$$C = A^T(VW_3) \tag{21}$$

Here $V = \{\boldsymbol{v_i}\} \in \mathbb{R}^{n \times d_{in}}$ represents an input sequence and $W_1, W_2, W_3 \in \mathbb{R}^{d_{in} \times d_{out}}$ are matrices to be learned for transforming V to its query, key and value representation. $C = \{\boldsymbol{c_i}\} \in \mathbb{R}^{n \times d_{out}}$ therefore forms a sequence of self-attentive token encoding. We can expect each input token to learn a deep context-aware configuration via adjusting its relation with its surroundings during end-to-end training. We should also be aware that the architecture excludes all recurrent and convolution layers, as computation within a self-attention layer is parallel (therefore outweighs RNN) and parameter-efficient (compared with CNN). Various techniques have been proposed to further enhance its representation power. Positional encoding (Vaswani et al. 2017) is introduced to provide the model with additional positional information of each token, an example of which can be constructed as follows:

$$PE(pos, 2i) = sin(pos/10000^{2i/d}) \tag{22}$$

$$PE(pos, 2i+1) = cos(pos/10000^{2i/d}) \tag{23}$$

and later incorporated into the sentence encoding as additional features. To avoid receving attention from undesired direction, a directional mask M (Shen et al. 2017), a triangular matrix with -∞ entries on the disabled position and 0s otherwise, is added to the score representation before normalizing:

$$\alpha_{ij} = softmax(e_{ij} + M_{ij}) \tag{24}$$

where

$$M_{ij} = \begin{cases} 0, & i > j \\ -\infty, & \text{otherwise} \end{cases} \tag{25}$$

for backward disabling. This can be useful while training a left-to-right language model since the future context should not provide any clue for generating the next token. Other techniques include relative position encoding (Shaw, Uszkoreit, and Vaswani 2018) that aims for incorporating pairwise distance information into the contextual token representation $\boldsymbol{c_i}$ as following:

$$\boldsymbol{c_i} = \sum_j \alpha_{ij}(\boldsymbol{v_j} + \boldsymbol{w_{ij}}) \tag{26}$$

where each weight $\boldsymbol{w_{ij}}$ corresponds to a directed edge from vertex $\boldsymbol{v_i}$ to $\boldsymbol{v_j}$, if V is considered as a fully connected graph. These weights are initialized and learned during training.

Besides the pair-wise score computation, Lin et al. proposed an alternative method to obtain self-attention scores based on a fully connected neural network:

$$A = softmax(W_2 tanh(W_1(V^T)) \tag{27}$$

where each row in A represents a single type of attention. Then the entire sequence encoding C can be obtained by:

$$C = AV \tag{28}$$

In this case, the attention that an element receives is determined by its relevance to all elements in the entire sequence via full connection. Such a technique can be used for obtaining a fix-length encoding from a variable-length sequence since the dimension of C is independent of the input sequence length.

3.4 Memory-based Attention

To introduce a new type of attention, we first reconstruct the old attention in an alternative way. Given a list of key value pairs $\{(\boldsymbol{k_i}, \boldsymbol{v_i})\}$ stored in memory and a query vector $\boldsymbol{q}$, we redefine the three steps as:

1. $e_i = a(\boldsymbol{q}, \boldsymbol{k_i})$ (address memory)

2. $\alpha_i = \frac{exp(e_i)}{\sum_i exp(e_i)}$ (normalize)

3. $\boldsymbol{c} = \sum_i \alpha_i \boldsymbol{v_i}$ (read contents)

Here we re-interpret computing attention score as soft memory addressing using query $\boldsymbol{q}$, and encoding as reading contents from memory based on attention scores $\{\alpha_i\}$, which constitutes the very basic form of memory-based attention. In fact, in quite a few literatures, "memory" is simply a synonym for the input sequence. Also note that if every pair of $\boldsymbol{k_i}$ and $\boldsymbol{v_i}$ are equal, this reduces to the basic attention mechanism. However, the alternative (memory-based) attention mechanism can become much more powerful as we incorporate additional functionalities to enable reusability and increase flexibility, both of which we will later discuss in detail.

Reusability A fundamental difficulty in some question answering tasks is that the answer is indirectly related to the question and therefore can not be easily solved via basic attention techniques (demonstrated at the beginning of this section). However, this can be achieved if we can simulate a temporal reasoning procedure by making iterative memory updates (also called multi-hop) to navigate attention to the correct location of the answer step-by-step (Sukhbaatar et al. 2015), an outline of which is illustrated in Figure 2. Intuitively, in each iteration, the query is updated with new contents, and updated query is used for retrieving relevant contents. A pseudo run on an early example is given in Figure 3, where the query is initialized as the original question and is later updated by simply summing up the current query and content (Sukhbaatar et al. 2015):

$$\boldsymbol{q}^{(t+1)} = \boldsymbol{q}^{(t)} + \boldsymbol{c}^{(t)} \tag{29}$$

More sophisticated update methods include constructing a recurrent network across query and content of multiple time steps (Kumar et al. 2016), or inducing the output based on both content and location information (Graves, Wayne, and Danihelka 2014). Results show that when complex temporal reasoning tasks are given (similar to Figure 3), the memory-based attention model can successfully locate the answer after several hops.

Flexibility Since keys and values are distinctly represented, we have freedom to incorporate prior knowledge in designing separate key and value embeddings to allow them to better capture relevant information respectively. To be specific, key embeddings can be manually designed to match the question and value embeddings to match the response. In key-value memory network (Miller et al. 2016), a window level representation is proposed such that keys are constructed as windows centered around entity tokens and values are those corresponding entities, aiming for more efficient and accurate matching. Then for the example in Figure 3, entities such as *apple* and *bedroom* are value embeddings, and their surrounding tokens are key embeddings.
More sophisticated architectures include the Dynamic Memory Network (Kumar et al. 2016) where the overall architecture is fine-split into four parts: question module, input module, episodic memory module and answer module,

1. $initialize\ \boldsymbol{q} = question$
2. $e_i = a(\boldsymbol{q}, \phi_k(\boldsymbol{k_i}))$ (address the memory)
3. $\alpha_i = \frac{exp(e_i)}{\sum_i exp(e_i)}$ (normalize)
4. $\boldsymbol{c} = \sum_i \alpha_i \phi_v(\boldsymbol{v_i})$ (retrieve contents)
5. $\boldsymbol{q} = update\_query(\boldsymbol{q}, \boldsymbol{c})$ (update query)
6. $goto$ 2 (multi-hop)

Figure 2: An enhanced version of memory-based attention with multi-hop and alternative key value embeddings

Sam walks into the kitchen. (1)
Sam picks up an apple. (2)
Sam walks into the bedroom. (3)
Sam drops the apple. (4)
Q: Where is the apple?
$\boldsymbol{q^{(0)}}$= Where is the apple?
$\boldsymbol{c^{(0)}}$= Sam drops the apple
$\boldsymbol{q^{(1)}}$= [Where is the apple?; Sam drops the apple]
$\boldsymbol{c^{(1)}}$= Sam walks into the bedroom.
A: bedroom

Figure 3: A pseudo illustration of memory-based attention updates on an early question answering example.

each of which is a complex neural micro-architecture itself. Such modularized design enables piecewise domain knowledge injection, efficient communication among modules, and generalization to a wider range of tasks beyond traditional question answering. A similar architecture is proposed to handle both textual and visual question answering tasks (Xiong, Merity, and Socher 2016), where visual inputs are fed into a deep convolutional network and high-level features are extracted and processed into an input sequence for the attention network. If we further extend memory and query representation to fields beyond question answering, memory-based attention techniques are also used in aspect and opinion term mining (Wang et al. 2017) where query is represented as aspect prototypes, in recommender systems (Zheng et al. 2018) where users become the memory component and items become queries, in topic modelings (Zeng et al. 2018) where latent topic representation extracted from a deep network constitutes the memory, etc.

3.5 Task-specific Attention

In this section we include alternative usage of attention that are intricately designed for a specific task. Although not as generalizable as methods introduced earlier, they are well-motivated and fit properly into their own task, therefore worth mentioning. Tan et al. proposed an alternative for computing attention scores for the task of abstractive document summarization (Tan, Wan, and Xiao 2017). The formulation of new graph-based attention is similar to PageRank algorithm (Page et al. 1999). Given a document $V = \{\boldsymbol{v_i}\}$ where each $\boldsymbol{v_i}$ represents a sentence, the stationary attention

distribution $\boldsymbol{\alpha} = \{\alpha_i\}$ satisfies:

$$\boldsymbol{\alpha} = \lambda W D^{-1} \boldsymbol{\alpha} + (1 - \lambda)\boldsymbol{y} \tag{30}$$

Here W is a square matrix in which each entry W_{ij} encodes a multiplicative composition between $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$, and D is a diagonal normalizing matrix to ensure each column of WD^{-1} sums up to 1. λ and $\boldsymbol{y}$ are the damping factor and an auxiliary uniform distribution, respectively. Then $\boldsymbol{\alpha}$ can be solved analytically as:

$$\boldsymbol{\alpha} = (1 - \lambda)(I - \lambda W D^{-1})^{-1} \boldsymbol{y} \tag{31}$$

The underlying motivation is that a sentence is important in a document if it is heavily linked with many important sentences.

Kim et al. proposed a structured attention network which integrates attention mechanism with probabilistic graphical model by introducing a sequence of discrete latent variables $\mathbf{Z} = \{\mathbf{z_i}\}$ as soft selectors into the input sequence (Kim et al. 2017). An attention distribution $p(\mathbf{Z} = \boldsymbol{z}|V, \boldsymbol{q})$ is generated from a conditional random field and then used to encode the context vector as an expectation over this probability distribution:

$$\boldsymbol{c} = E_{\boldsymbol{z} \sim p(\boldsymbol{z}|V, \boldsymbol{q})}[f(V, \boldsymbol{z})] \tag{32}$$

where $f(V, \boldsymbol{z})$ is the annotated function that models relationship between latent variables and the given inputs. If $\mathbf{Z}$ is a single random variable and given $f(V, \mathbf{Z} = z) = V_z$ (i.e., selecting the z^{th} element from V), then this is equivalent to soft selection as in the basic attention. Without this restriction, they demonstrate its adaptability to multiple input selection tasks such as syntactic tree selection and subsequence selection under a general case.

In machine translation, a local attention model (Luong, Pham, and Manning 2015) is proposed to handle long sequence translation where computation of global attention (i.e. attending to every element) is expensive. During decoding, a pivot position $p_t \in [0, length(V)]$, which specifies the center of attention, is first predicted by a small neural network, and then Gaussian Smoothing is applied around the center to produce soft alignment.

4 Application

In the previous section, we have showed that attention along with its variants have been widely applied to various NLP tasks. Here we will further explore the connection of attention to other abstract concepts in machine learning. As we have discussed previously, attention can be used for *encoding* a sequence by extracting important terms based on its match with a given pattern; attention can also be used for iterative memory *addressing* and reading given a query. Here we present three more applications of attention: *emsemble*, *gating*, and *pre-training*.

4.1 Attention for Ensemble

If we interpret each element $\boldsymbol{v_i}$ in a sequence as an individual model, and normalized scores α_i as their weighted votes, applying the attention mechanism can then be analogous to model ensemble. This is explored in Kieta et al. where they ensemble a set of word embeddings to construct a meta-embedding with more representative power and flexibility (Kiela, Wang, and Cho 2018). Specifically, attention score $\alpha_i^{[j]}$ for the embedding $\boldsymbol{v_i^{[j]}}$ (the i^{th} embedding for the j^{th} word) is given via self-attention:

$$\alpha_i^{[j]} = softmax(\boldsymbol{w^T v_i^{[j]}} + b) \tag{33}$$

And the meta-embedding $\boldsymbol{v^{[j]}}$ for the j^{th} word is given by:

$$\boldsymbol{v^{[j]}} = \sum_i \alpha_i^{[j]} \boldsymbol{v_i^{[j]}} \tag{34}$$

They demonstrate that certain embeddings are preferred over others depending on characteristics of the word, such as concreteness and frequency. For example, the ImageNet embedding (He et al. 2016) receives larger weights than FastText embeddings (Bojanowski et al. 2016) for concrete words.

4.2 Attention for Gating

Another application of attention is to integrate this mechanisms with memory updates in recurrent network. In traditional GRU (Chung et al. 2014), hidden state updates are given by:

$$\tilde{\boldsymbol{h_i}} = tanh(W\boldsymbol{v_i} + \boldsymbol{r_i} \circ (U\boldsymbol{h_{i-1}}) + \boldsymbol{b^{(h)}}) \tag{35}$$

$$\boldsymbol{h_i} = \boldsymbol{u_i} \circ \tilde{\boldsymbol{h_i}} + (\mathbf{1} - \boldsymbol{u_i}) \circ \boldsymbol{h_{i-1}} \tag{36}$$

where $\boldsymbol{u_i}$ and $\boldsymbol{r_i}$ are update and reset gates learned during training. While in an alternative attention-based GRU (Xiong, Merity, and Socher 2016), $\boldsymbol{u_i}$ is replaced by a scalar attention score α_i received by the i^{th} elelment when updating its hidden state. Then the last update step can be replaced by:

$$\boldsymbol{h_i} = \alpha_i \circ \tilde{\boldsymbol{h_i}} + (1 - \alpha_i) \circ \boldsymbol{h_{i-1}} \tag{37}$$

The attention scores are computed in an external module. Such an attention-based gating allows context-aware updates based on global knowledge of previous memory, and easier interpretability of importance of each element.

Similarly in text comprehension, memory-based attention gate $\tilde{\boldsymbol{q_i}}$ is constructed (Dhingra et al. 2016) based on the interaction between the query Q and each token $\boldsymbol{v_i}$ in the document and iteratively update each token embeddings:

$$\alpha_i = softmax(Q^T \boldsymbol{v_i}) \tag{38}$$

$$\tilde{\boldsymbol{q_i}} = Q\alpha_i \tag{39}$$

$$\tilde{\boldsymbol{v_i}} = \boldsymbol{v_i} \circ \tilde{\boldsymbol{q_i}} \tag{40}$$

$$\boldsymbol{v_i}, Q = GRU_v(\tilde{\boldsymbol{v_i}}), GRU_Q(Q) \tag{41}$$

aiming to build up deep query-specific token representation.

4.3 Attention for Pre-training

Pre-trained word embeddings are crucial to many NLP tasks. Traditional methods such as *Skipgram, Cbow, and Glove* (Mikolov et al. 2013a; Pennington, Socher, and Manning 2014; Mikolov et al. 2013b) take use of large text corpora to train an upsupervised prediction model based on contexts and learn a high dimensional distributed representation of each token. On the contrary, recently proposed pre-training

methods integrate attention-based techniques with deep neural architectures, aiming to learn higher quality token representation that incorporates syntactic and semantic information from the surrounding contexts. and then the model is fine-tuned to adapt to a downstream supervised task. BERT (Devlin et al. 2018) is a bi-directional pre-training model backboned by the Transformer Encoder (Vaswani et al. 2017), a deep hybrid neural network with feed forward layers and self-attention layers which we have briefly discussed in section 3.3. During pre-training, one task is to learn a bi-directional masked language model, meaning that a small percent of tokens in a sentence are masked and the goal is to predict these tokens based on their context. The other task is binary next sentence prediction, where two spans of texts are sampled from the corpora and the model is trained to predict whether they are contiguous. As discussed in Section 3.3, each token redistributes attention across the sequence and reconstruct its interaction with other tokens in a self-adaptive manner as the training proceeds, aiming to learn its contextual representation based on the entire sequence. When the pre-trained model is integrated with a supervised task, an additional layer is added on top of the model and fine-tuned to adapt to its supervised downstream task. The new model has achieved ground-breaking results on various NLP tasks, by focusing on pre-training the deep hybrid architecture on large text corpora and then sparing minimal efforts on fine-tuning. Other attention-based pre-training models include OpenAI GPT, which instead uses a Transformer Decoder (with backward disabling mask) to pre-train a deep left-to-right language model based on a different set of tasks.

5 Evaluation

Compared to the universal usage of attention, only a few attempts are made either to give a rigorous mathematical justification of why it works in various scenarios. Nevertheless, there are several works that have attempted to set up standards on evaluating its performance, either qualitatively or quantitatively, task-specific or general, and here we give a short summarization of these approaches.

5.1 Quantitative

Quantitative evaluation on attention can be further divided into intrinsic or extrinsic based on whether the contribution of attention is assessed on itself or along within a downstream supervised task.

Intrinsic evaluation methods are typically proposed in machine translation (Luong, Pham, and Manning 2015), where attention is analogous to word alignment, performance of attention could be directly measured by comparing the attention distribution with the gold alignment data, and quantified using alignment error rate (AER). Similarly, Liu et al. proposed a method to manually construct "gold attention vectors" (Liu et al. 2017) by first identifying labelled key words within a sentence and then conducting post-processing procedures such as smoothing and normalization, given abundant well-annotated data. For example, for the sentence ***Mohamad*** *fired* ***Anwar****, his former* ***protege****, in* ***1998***, the four tokens in boldface are labelled as argument words and receive an attention score of 0.25 each ($0.25 \times 4 = 1$), and

System	Ppl.	BLEU
WMT'15 systems		
SOTA – *phrase-based* (Edinburgh)		**29.2**
NMT + 5-gram rerank (MILA)		27.6
Our NMT systems		
Base (reverse)	14.3	16.9
+ global (*location*)	12.7	19.1 (+2.2)
+ global (*location*) + feed	10.9	20.1 (+*1.0*)
+ global (*dot*) + drop + feed	9.7	22.8 (+2.7)
+ global (*dot*) + drop + feed + unk		24.9 (+2.*1*)

Figure 4: An illustration of extrinsic evaluation of different attention mechanisms under a downstream machine translation task. The image is modified from (Luong, Pham, and Manning 2015)

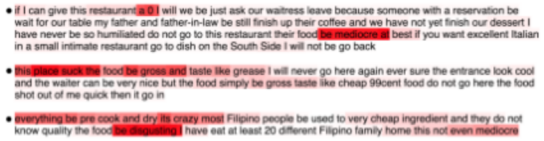

Figure 5: An illustration of qualitative methods based on heatmaps. Higher color intensity indicates higher attention score. The image is modified from (Lin et al. 2017)

then smoothing is optionally applied around each attended token. Though intrinsic evaluation methods produce precise measurements on performance, they tend to be restricted to their specific tasks and rely heavily on plentitude of labelled data.

On the other hand, extrinsic evaluation methods (Figure 4) are more general and widely used. This can be easily formulated by comparing the overall performance across different models under the downstream task. However, the result could be misleading since whether or not the improvements should be attributed to the attention component can not be determined.

5.2 Qualitative

Qualitative evaluation for attention is currently the most widely used evaluation technique, due to its simplicity and convenience for visualization (Figure 5). To be specific, a heat-map is constructed across the entire sentence where the intensity is proportional to the normalized attention score each element receives. Intuitively, attention is expected to be focused on key words for the corresponding task. However, such an approach turns out to be better for visualization than for analysis.

6 Conclusion and Prospects

In this paper, we have surveyed through recent works on the attention mechanism and conducted an introductory summary based on its formulation, variation, application and evaluation. Compared to its wide usage in various NLP tasks, attempts to explore its mathematical justification still

remain scarce. Recent works that explore its application in embedding pre-training have attained great success and might be a prospective area of future research.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhingra, B.; Liu, H.; Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Du, J.; Han, J.; Way, A.; and Wan, D. 2018. Multi-level structured self-attentions for distantly supervised relation extraction. *arXiv preprint arXiv:1809.00699*.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ji, J.; Wang, Q.; Toutanova, K.; Gong, Y.; Truong, S.; and Gao, J. 2017. A nested attention neural hybrid model for grammatical error correction. *arXiv preprint arXiv:1707.02026*.

Kalchbrenner, N., and Blunsom, P. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700–1709.

Kiela, D.; Wang, C.; and Cho, K. 2018. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1466–1477.

Kim, Y.; Denton, C.; Hoang, L.; and Rush, A. M. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Kumar, A.; Irsoy, O.; Ondruska, P.; Iyyer, M.; Bradbury, J.; Gulrajani, I.; Zhong, V.; Paulus, R.; and Socher, R. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, 1378–1387.

Lin, Z.; Feng, M.; Santos, C. N. d.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Liu, S.; Chen, Y.; Liu, K.; and Zhao, J. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1789–1798.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Miller, A.; Fisch, A.; Dodge, J.; Karimi, A.-H.; Bordes, A.; and Weston, J. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; and Zhang, C. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, 2440–2448.

Tan, J.; Wan, X.; and Xiao, J. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1171–1181.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Wang, W.; Pan, S. J.; Dahlmeier, D.; and Xiao, X. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *AAAI*, 3316–3322.

Xiong, C.; Merity, S.; and Socher, R. 2016. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, 2397–2406.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1480–1489.

Yu, L.; Bansal, M.; and Berg, T. L. 2017. Hierarchically-attentive rnn for album summarization and storytelling. *arXiv preprint arXiv:1708.02977*.

Zeng, J.; Li, J.; Song, Y.; Gao, C.; Lyu, M. R.; and King, I. 2018. Topic memory networks for short text classification.

Zheng, L.; Lu, C.-T.; He, L.; Xie, S.; Noroozi, V.; Huang, H.; and Yu, P. S. 2018. Mars: Memory attention-aware recommender system. *arXiv preprint arXiv:1805.07037*.