A General Survey on Attention Mechanisms in Deep Learning

Gianni Brauwers and Flavius Frasincar

Abstract—Attention is an important mechanism that can be employed for a variety of deep learning models across many different domains and tasks. This survey provides an overview of the most important attention mechanisms proposed in the literature. The various attention mechanisms are explained by means of a framework consisting of a general attention model, uniform notation, and a comprehensive taxonomy of attention mechanisms. Furthermore, the various measures for evaluating attention models are reviewed, and methods to characterize the structure of attention models based on the proposed framework are discussed. Last, future work in the field of attention models is considered.

Index Terms—Attention models, deep learning, introductory and survey, neural nets, supervised learning

1 Introduction

The idea of mimicking human attention first arose in the field of computer vision [1], [2] in an attempt to reduce the computational complexity of image processing while improving performance by introducing a model that would only focus on specific regions of images instead of the entire picture. Although, the true starting point of the attention mechanisms we know today is often attributed to originate in the field of natural language processing [3]. Bahdanau et al. [3] implement attention in a machine translation model to address certain issues with the structure of recurrent neural networks. After Bahdanau et al. [3] emphasized the advantages of attention, the attention techniques were refined [4] and quickly became popular for a variety of tasks, such as text classification [5], [6], image captioning [7], [8], sentiment analysis [6], [9], and speech recognition [10], [11], [12].

Attention has become a popular technique in deep learning for several reasons. Firstly, models that incorporate attention mechanisms attain state-of-the-art results for all of the previously mentioned tasks, and many others. Furthermore, most attention mechanisms can be trained jointly with a base model, such as a recurrent neural network or a convolutional neural network using regular backpropagation [3]. Additionally, attention introduces a certain type of interpretation into neural network models [8] that are generally known to be highly complicated to interpret. Moreover, the popularity of attention mechanisms was additionally boosted after the introduction of the Transformer model [13] that further proved how effective attention can be. Attention was originally introduced as an extension to recurrent neural networks [14]. However, the Transformer model proposed in [13] poses a major development in attention research as it demonstrates that the attention mechanism is sufficient to build a state-of-the-art model. This means that disadvantages, such as the fact that recurrent neural networks are particularly difficult to parallelize, can be circumvented. As was the case for the introduction of the original attention mechanism [3], the Transformer model was created for machine translation, but was quickly adopted to be used for other tasks, such as image processing [15], video processing [16], and recommender systems [17].

The purpose of this survey is to explain the general form of attention, and provide a comprehensive overview of attention techniques in deep learning. Other surveys have already been published on the subject of attention models. For example, in [18], a survey is presented on attention in computer vision, [19] provides an overview of attention in graph models, and [20], [21], [22] are all surveys on attention in natural language processing. This paper partly builds on the information presented in the previously mentioned surveys. Yet, we provide our own significant contributions. The main difference between this survey and the previously mentioned ones is that the other surveys generally focus on attention models within a certain domain. This survey, however, provides a cross-domain overview of attention techniques. We discuss the attention techniques in a general way, allowing them to be understood and applied in a variety of domains. Furthermore, we found the taxonomies presented in previous surveys to be lacking the depth and structure needed to properly distinguish the various attention mechanisms. Additionally, certain significant attention techniques have not yet been properly discussed in previous surveys, while other presented attention mechanisms seem to be lacking either technical details or intuitive explanations. Therefore, in this paper, we present important attention techniques by means of a single framework using a uniform notation, a combination of both technical and intuitive explanations for each presented attention technique, and a comprehensive taxonomy of attention mechanisms.

The structure of this paper is as follows. Section 2 introduces a general attention model that provides the reader with a basic understanding of the properties of attention and how it can be applied. One of the main contributions of this paper is the taxonomy of attention techniques presented in Section 3. In this section, attention mechanisms are explained and categorized according to the presented

G. Brauwers and F. Frasincar are with the Erasmus School of Economics, Erasmus University Rotterdam, 3000 DR, Rotterdam, the Netherlands (e-mail: {frasincar, brauwers}@ese.eur.nl).

Manuscript received July 6, 2020; revised June 21, 2021; Corresponding author: F. Frasincar

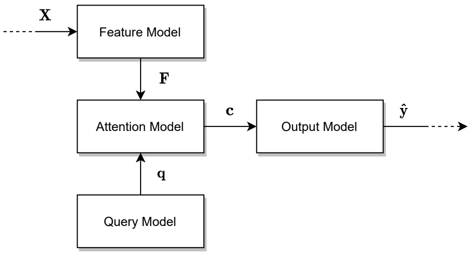

Fig. 1. An illustration of the general structure of the task model.

taxonomy. Section 4 provides an overview of performance measures and methods for evaluating attention models. Furthermore, the taxonomy is used to evaluate the structure of various attention models. Lastly, in Section 5, we give our conclusions and suggestions for further research.

2 General Attention Model

This section presents a general form of attention with corresponding notation. The notation introduced here is based on the notation that was introduced in [23] and popularized in [13]. The framework presented in this section is used throughout the rest of this paper.

To implement a general attention model, it is necessary to first describe the general characteristics of a model that can employ attention. We will refer to the complete model as the **task model**, of which the structure is presented in Fig. 1. This model simply takes an input, carries out the specified task, and produces the desired output. For example, the task model can be a language model that takes as input a piece of text, and produces as output a summary of the contents, a classification of the sentiment, or the text translated word for word to another language. Alternatively, the task model can take an image, and produce a caption or segmentation for that image. The task model consists of four submodels: the feature model, the query model, the attention model, and the output model. In Subsection 2.1, the feature model and query model are discussed, which are used to prepare the input for the attention calculation. In Subsection 2.2, the attention model and output model are discussed, which are concerned with producing the output. Last, in Subsection 2.3, we highlight the applications of attention.

2.1 Attention Input

Suppose the task model takes as input the matrix X 2 R$^{d_x \ \ n_x}$, where d_x represents the size of the input vectors and n_x represents the amount of input vectors. The columns in this matrix can represent the words in a sentence, the pixels in an image, the characteristics of an acoustic sequence, or any other collection of inputs. The **feature model** is then employed to extract the n_f feature vectors $f_1; : : : ; f_{n_f}$ 2 R^{d_f} from X, where d_f represents the size of the feature vectors. The feature model can be a recurrent neural network (RNN), a convolutional neural network (CNN), a simple embedding layer, a linear transformation of the original data, or no transformation at all. Essentially, the feature model consists of all the steps that transform the original input X into the feature vectors $f_1; : : : ; f_{n_f}$ that the attention model will attend to.

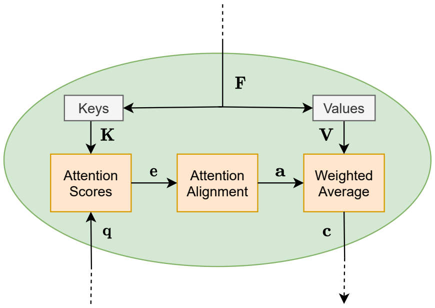

Fig. 2. The inner mechanisms of the general attention module.

To determine which vectors to attend to, the attention model requires the query q 2 R^{d_q}, where d_q indicates the size of the query vector. This query is extracted by the **query model**, and is generally designed based on the type of output that is desired of the model. A query tells the attention model which feature vectors to attend to. It can be interpreted literally as a query, or a question. For example, for the task of image captioning, suppose that one uses a decoder RNN model to produce the output caption based on feature vectors obtained from the image by a CNN. At each prediction step, the hidden state of the RNN model can be used as a query to attend to the CNN feature vectors. In each step, the query is a question in the sense that it asks for the necessary information from the feature vectors based on the current prediction context.

2.2 Attention Output

The feature vectors and query are used as input for the **attention model**. This model consists of a single, or a collection of **general attention modules**. An overview of a general attention module is presented in Fig. 2. The input of the general attention module is the query q 2 R^{d_q}, and the matrix of feature vectors $F = [f_1; : : : ; f_{n_f}]$ 2 R$^{d_f \ \ n_f}$. Two separate matrices are extracted from the matrix F: the keys matrix $K = [k_1; : : : ; k_{n_f}]$ 2 R$^{d_k \ \ n_f}$, and the values matrix $V = [v_1; : : : ; v_{n_f}]$ 2 R$^{d_v \ \ n_f}$, where d_k and d_v indicate, respectively, the dimensions of the key vectors (columns of K) and value vectors (columns of V). The general way of obtaining these matrices is through a linear transformation of F using the weight matrices W_K 2 R$^{d_k \ \ d_f}$ and W_V 2 R$^{d_v \ \ d_f}$, for K and V, respectively. The calculations of K and V are presented in (1). Both weight matrices can be learned during training or predefined by the researcher. For example, one can choose to define both W_K and W_V as equal to the identity matrix to retain the original feature vectors. Other ways of defining the keys and the values are also possible, such as using completely separate inputs for the keys and values. The only constraint to be obeyed is that the number of columns in K and V remains the same.

$$\underset{d_k \ \ n_f}{K} = \underset{d_k \ \ d_f}{W_K} \quad \underset{d_f \ \ n_f}{F} ; \qquad \underset{d_v \ \ n_f}{V} = \underset{d_v \ \ d_f}{W_V} \quad \underset{d_f \ \ n_f}{F} : \tag{1}$$

The goal of the attention module is to produce a weighted average of the value vectors in V. The weights used to produce this output are obtained via an attention scoring and alignment step. The query q and the keys matrix K are used to calculate the vector of attention scores

$\boldsymbol{e} = [e_1, \ldots, e_{n_f}] \in \mathbb{R}^{n_f}$. This is done via the score function score(), as illustrated in (2).

$$\underbrace{e_l}_{1\times 1} = \text{score}(\underbrace{\boldsymbol{q}}_{d_q\times 1}, \underbrace{\boldsymbol{k}_l}_{d_k\times 1}). \tag{2}$$

As discussed before, the query symbolizes a request for information. The attention score e_l represents how important the information contained in the key vector $\boldsymbol{k}_l$ is according to the query. If the dimensions of the query and key vectors are the same, an example of a score function would be to take the dot-product of the vectors. The different types of score functions are further discussed in Section 3.2.1.

Next, the attention scores are processed further through an alignment layer. The attention scores can generally have a wide range outside of $[0, 1]$. However, since the goal is to produce a weighted average, the scores are redistributed via an alignment function align() as defined in (3).

$$\underbrace{a_l}_{1\times 1} = \text{align}(\underbrace{e_l}_{1\times 1}; \underbrace{\boldsymbol{e}}_{n_f\times 1}), \tag{3}$$

where $a_l \in \mathbb{R}^1$ is the attention weight corresponding to the lth value vector. One example of an alignment function would be to use a softmax function, but the various other alignment types are discussed in Section 3.2.2. The attention weights provide a rather intuitive interpretation for the attention module. Each weight is a direct indication of how important each feature vector is relative to the others for this particular problem. This can provide us with a more in-depth understanding of the model behaviour, and the relations between inputs and outputs. The vector of attention weights $\boldsymbol{a} = [a_1, \ldots, a_{n_f}] \in \mathbb{R}^{n_f}$ is used to produce the context vector $\boldsymbol{c} \in \mathbb{R}^{d_v}$ by calculating a weighted average of the columns of the values matrix $\boldsymbol{V}$, as shown in (4).

$$\underbrace{\boldsymbol{c}}_{d_v\times 1} = \sum_{l=1}^{n_f} \underbrace{a_l}_{1\times 1} \times \underbrace{\boldsymbol{v}_l}_{d_v\times 1}. \tag{4}$$

As illustrated in Fig. 1, the context vector is then used in the **output model** to create the output $\hat{\boldsymbol{y}}$. This output model translates the context vector into an output prediction. For example, it could be a simple softmax layer that takes as input the context vector $\boldsymbol{c}$, as shown in (5).

$$\underbrace{\hat{\boldsymbol{y}}}_{d_y\times 1} = \text{softmax}(\underbrace{\boldsymbol{W}_c}_{d_y\times d_v} \times \underbrace{\boldsymbol{c}}_{d_v\times 1} + \underbrace{\boldsymbol{b}_c}_{d_y\times 1}), \tag{5}$$

where d_y is the number of output choices or classes, and $\boldsymbol{W}_c \in \mathbb{R}^{d_y\times d_v}$ and $\boldsymbol{b}_c \in \mathbb{R}^{d_y}$ are trainable weights.

2.3 Attention Applications

Attention is a rather general mechanism that can be used in a wide variety of problem domains. Consider the task of machine translation using an RNN model. Also, consider the problem of image classification using a basic CNN model. While an RNN produces a sequence of hidden state vectors, a CNN creates feature maps, where each region in the image is represented by a feature vector. The RNN hidden states are organized sequentially, while the CNN feature maps are organized spatially. Yet, attention can still be applied in both situations, since the attention mechanism does not inherently depend on the organization of the feature vectors. This characteristic makes attention easy to implement in a wide variety of models in different domains.

Another domain where attention can be applied is audio processing [24], [25]. Acoustic sequences can be represented by a sequence of feature vectors that relate to certain time periods of the audio sample. These vectors could simply be the raw input audio, or they can be extracted via, for example, an RNN or CNN. Video processing is another domain where attention can be applied intuitively [26], [27]. Video data consists of sequences of images, so attention can be applied to the individual images, as well as the entire sequence. Recommender systems often incorporate a user's interaction history to produce recommendations. Feature vectors can be extracted based on, for example, the id's or other characteristics of the products the user interacted with, and attention can be applied to them [28]. Attention can generally also be applied to many problems that use a time series as input, be it medical [29], financial [30], or anything else, as long as feature vectors can be extracted.

The fact that attention does not rely on the organization of the feature vectors allows it to be applied to various problems that each use data with different structures, as illustrated by the previous domain examples. Yet, this can be taken even further by applying attention to data where there is irregular structure. For example, protein structures, city traffic flows, and communication networks cannot always be represented using neatly structured organizations, such as sequences, like time series, or grids, like images. In such cases, the different aspects of the data are often represented as nodes in a graph. These nodes can be represented by feature vectors, meaning that attention can be applied in domains that use graph-structured data as well [19], [31].

In general, attention can be applied to any problem for which a set of feature vectors can be defined or extracted. As such, the general attention model presented in Fig. 2 is applicable to a wide range of domains. The problem, however, is that there is a large variety of different applications and extensions of the general attention module. As such, in Section 3, a comprehensive overview is provided of a collection of different attention mechanisms.

3 Attention Taxonomy

There are many different types of attention mechanisms and extensions, and a model can use different combinations of these attention techniques. As such, we propose a taxonomy that can be used to classify different types of attention mechanisms. Fig. 3 provides a visual overview of the different categories and subcategories that the attention mechanisms can be organized in. The three major categories are based on whether an attention technique is designed to handle specific types of feature vectors (feature-related), specific types of model queries (query-related), or whether it is simply a general mechanism that is related to neither the feature model, nor the query model (general). Further explanations of these categories and their subcategories are provided in the following subsections. Each mechanism discussed in this section is either a modification to the existing inner mechanisms of the general attention module presented in Section 2, or an extension of it.

The presented taxonomy can also be used to analyze the architecture of attention models. Namely, the major categories and their subcategories can be interpreted as

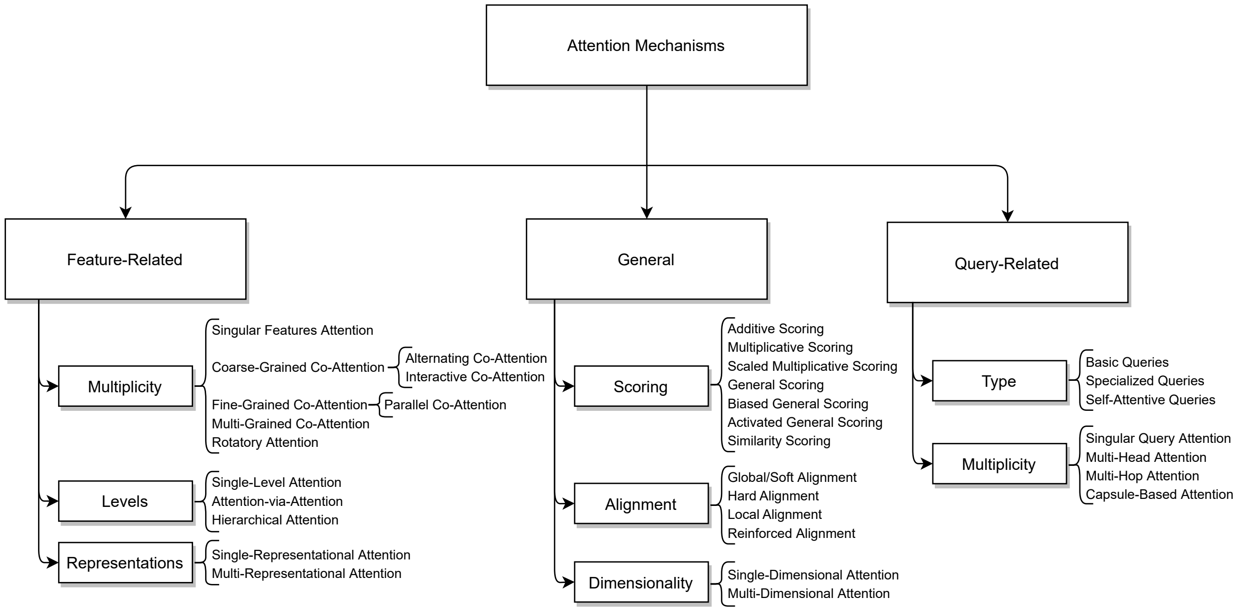

Fig. 3. A taxonomy of attention mechanisms.

TABLE 1
Notation.

Symbol	Description
$\boldsymbol{F}$	Matrix of size $d_f \times n_f$ containing the feature vectors $\boldsymbol{f}_1; : : : ; \boldsymbol{f}_{n_f} \in \mathrm{R}^{d_f}$ as columns. These feature vectors are extracted by the feature model.
$\boldsymbol{K}$	Matrix of size $d_k \times n_f$ containing the key vectors $\boldsymbol{k}_1; : : : ; \boldsymbol{k}_{n_f} \in \mathrm{R}^{d_k}$ as columns. These vectors are used to calculate the attention scores.
$\boldsymbol{V}$	Matrix of size $d_v \times n_f$ containing the value vectors $\boldsymbol{v}_1; : : : ; \boldsymbol{v}_{n_f} \in \mathrm{R}^{d_v}$ as columns. These vectors are used to calculate the context vector.
$\boldsymbol{W}_K$	Weights matrix of size $d_k \times \mathrm{d}_f$ used to create the $\boldsymbol{K}$ matrix from the $\boldsymbol{F}$ matrix.
$\boldsymbol{W}_V$	Weights matrix of size $d_v \times \mathrm{d}_f$ used to create the $\boldsymbol{V}$ matrix from the $\boldsymbol{F}$ matrix.
$\boldsymbol{q}$	Query vector of size d_q. This vector essentially represents a question, and is used to calculate the attention scores.
$\boldsymbol{c}$	Context vector of size d_v. This vector is the output of the attention model.
$\boldsymbol{e}$	Score vector of size d_{n_f} containing the attention scores $e_1; : : : ; e_{n_f} \in \mathrm{R}^1$. These are used to calculate the attention weights.
$\boldsymbol{a}$	Attention weights vector of size d_{n_f} containing the attention weights $a_1; : : : ; a_{n_f} \in \mathrm{R}^1$. These are the weights used in the calculation of the context vector.

orthogonal dimensions of an attention model. An attention model can consist of a combination of techniques taken from any or all categories. Some characteristics, such as the scoring and alignment functions, are generally required for any attention model. Other mechanisms, such as multi-head attention or co-attention are not necessary in every situation. Lastly, in Table 1, an overview of used notation with corresponding descriptions is provided.

3.1 Feature-Related Attention Mechanisms

Based on a particular set of input data, a feature model extracts feature vectors so that the attention model can attend to these various vectors. These features may have specific structures that require special attention mechanisms to handle them. These mechanisms can be categorized to deal with one of the following feature characteristics: the multiplicity of features, the levels of features, or the representations of features.

3.1.1 *Multiplicity of Features*

For most tasks, a model only processes a single input, such as an image, a sentence, or an acoustic sequence. We refer to such a mechanism as **singular features attention**. Other models are designed to use attention based on multiple inputs to allow one to introduce more information into the model that can be exploited in various ways. However, this does imply the presence of multiple feature matrices that require special attention mechanisms to be fully used. For example, [32] introduces a concept named **co-attention** to allow the proposed visual question answering (VQA) model to jointly attend to both an image and a question.

Co-attention mechanisms can generally be split up into two groups [33]: **coarse-grained co-attention** and **fine-grained co-attention**. The difference between the two groups is the way attention scores are calculated based on the two feature matrices. Coarse-grained attention mechanisms use a compact representation of one feature matrix as a query when attending to the other feature vectors. Fine-grained co-attention, on the other hand, uses all feature vectors of one input as queries. As such, no information is lost, which is why these mechanisms are called fine-grained.

As an example of coarse-grained co-attention, [32] proposes an **alternating co-attention** mechanism that uses the context vector (which is a compact representation) from one attention module as the query for the other module, and vice versa. Alternating co-attention is presented in Fig. 4. Given a set of two input matrices $X^{(1)}$ and $X^{(2)}$, features are extracted by a feature model to produce the feature matrices $F^{(1)}\ 2\ \mathrm{R}^{d_f^{(1)} \quad n_f^{(1)}}$ and $F^{(2)}\ 2\ \mathrm{R}^{d_f^{(2)} \quad n_f^{(2)}}$, where $d_f^{(1)}$

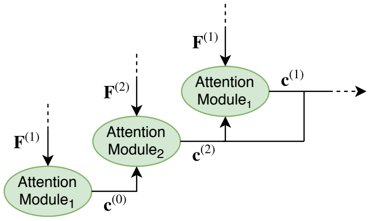

Fig. 4. An illustration of alternating co-attention.

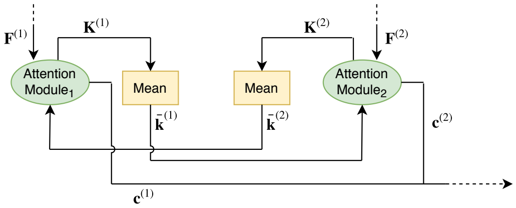

Fig. 5. An illustration of interactive co-attention.

and $d_f^{(2)}$ represent, respectively, the dimension of the feature vectors extracted from the first and second inputs, while $n_f^{(1)}$ and $n_f^{(2)}$ represent, respectively, the amount of feature vectors extracted from the first and second inputs. In [32], co-attention is used for VQA, so the two input matrices are the image data and the question data, for which the feature model for the image consists of a CNN model, and the feature model for the question consists of word embeddings, a convolutional layer, a pooling layer, and an LSTM model. Firstly, attention is calculated for the first set of features $F^{(1)}$ without the use of a query (Attention Module$_1$ in Fig. 4). In [32], an adjusted additive attention score function is used for this attention mechanism. The general form of the regular additive score function can be seen in (6).

$$\text{score}(\underbrace{q}_{d_q\times 1}, \underbrace{k_l}_{d_k\times 1}) = \underbrace{w^T}_{1\times d_w} \times \text{act}(\underbrace{W_1}_{d_w\times d_q} \times \underbrace{q}_{d_q\times 1} + \underbrace{W_2}_{d_w\times d_k} \times \underbrace{k_l}_{d_k\times 1} + \underbrace{b}_{d_w\times 1}); \quad (6)$$

where act() is a non-linear activation function, and $w \in \mathbb{R}^{d_w}$, $W_1 \in \mathbb{R}^{d_w \times d_q}$, $W_2 \in \mathbb{R}^{d_w \times d_k}$, and $b \in \mathbb{R}^{d_w}$ are trainable weights matrices, for which d_w is a predefined dimension of the weight matrices. A variant of this score function adapted to be calculated without a query for the application at hand can be seen in (7).

$$\underbrace{e_l^{(0)}}_{1\times 1} = \underbrace{w^{(1)T}}_{1\times d_w} \times \text{act}(\underbrace{W^{(1)}}_{d_w\times d_k^{(1)}} \times \underbrace{k_l^{(1)}}_{d_k^{(1)}\times 1} + \underbrace{b^{(1)}}_{d_w\times 1}); \quad (7)$$

where $w^{(1)} \in \mathbb{R}^{d_w}$, $W^{(1)} \in \mathbb{R}^{d_w \times d_k^{(1)}}$, and $b^{(1)} \in \mathbb{R}^{d_w}$ are trainable weight matrices for Attention Module$_1$, $k_l^{(1)} \in \mathbb{R}^{d_k^{(1)}}$ is the lth column of the keys matrix $K^{(1)}$ that was obtained from $F^{(1)}$ via a linear transformation (see (1)), for which d_w is a prespecified dimension of the weight matrices and $d_k^{(1)}$ is a prespecified dimension of the key vectors.

Perhaps one may wonder why the query is absent when calculating attention in this manner. Essentially, the query in this attention model is learned alongside the other trainable parameters. As such, the query can be interpreted as a general question: "Which feature vectors contain the most important information?". This is also known as a **self-attentive mechanism**, since attention is calculated based only on the feature vectors themselves. Self-attention is explained in more detail in Subsection 3.3.1.

The scores are combined with an alignment function (see (3)), such as the softmax function, to create attention weights used to calculate the context vector $c^{(0)} \in \mathbb{R}^{d_v^{(1)}}$ (see (4)). This context vector is not used as the output of the attention model, but rather as a query for calculating the context vector $c^{(2)} \in \mathbb{R}^{d_v^{(2)}}$, based on the second feature matrix $F^{(2)}$, where $d_v^{(2)}$ is the dimension of the value vectors obtained from $F^{(2)}$ via a linear transformation (see (1)). For this module (Attention Module$_2$ in Fig. 4), attention scores are calculated using another score function with c_0 as query input, as presented in (8). Any function can be used in this situation, but an additive function is used in [32].

$$\underbrace{e_l^{(2)}}_{1\times 1} = \text{score}(\underbrace{c^{(0)}}_{d_v^{(1)}\times 1}, \underbrace{k_l^{(2)}}_{d_k^{(2)}\times 1}). \quad (8)$$

These attention scores are then used to calculate attention weights using, for example, a softmax function as alignment function, after which the context vector $c^{(2)}$ can be derived as a weighted average of the second set of value vectors. Finally, the context vector $c^{(2)}$ is used as a query for the first attention module, which will produce the context vector $c^{(1)}$ for the first feature matrix $F^{(1)}$. Attention scores are calculated according to (9). In [32], the same function and weight matrices as seen in (7) are used, but with an added query making it the same as the general additive score function (see (6)). The rest of the attention calculation is similar as before.

$$\underbrace{e_l^{(1)}}_{1\times 1} = \text{score}(\underbrace{c^{(2)}}_{d_v^{(2)}\times 1}, \underbrace{k_l^{(1)}}_{d_k^{(1)}\times 1}). \quad (9)$$

The produced context vectors $c^{(1)}$ and $c^{(2)}$ are concatenated and used for prediction in the output model. Alternating co-attention inherently contains a form of sequentiality due to the fact that context vectors need to be calculated one after another. This may come with a computational disadvantage since it is not possible to parallelize. Instead of using a sequential mechanism like alternating co-attention, [34] proposes the **interactive co-attention** mechanism that can calculate attention on both feature matrices in parallel, as depicted in Fig. 5. Instead of using the context vectors as queries, unweighted averages of the key vectors are used as queries. The calculation of the average keys are provided in (10), and the calculation of the attention scores are shown in (11). Any score function can be used in this case, but an additive score function is used in [34].

$$\underbrace{\bar{k}^{(1)}}_{d_k^{(1)}\times 1} = \frac{1}{n_f^{(1)}} \sum_{l=1}^{n_f^{(1)}} \underbrace{k_l^{(1)}}_{d_k^{(1)}\times 1}, \qquad \underbrace{\bar{k}^{(2)}}_{d_k^{(2)}\times 1} = \frac{1}{n_f^{(2)}} \sum_{l=1}^{n_f^{(2)}} \underbrace{k_l^{(2)}}_{d_k^{(2)}\times 1}; \quad (10)$$

$$\underbrace{e_l^{(1)}}_{1\times 1} = \text{score}(\underbrace{\bar{k}^{(2)}}_{d_k^{(2)}\times 1}, \underbrace{k_l^{(1)}}_{d_k^{(1)}\times 1}), \quad \underbrace{e_l^{(2)}}_{1\times 1} = \text{score}(\underbrace{\bar{k}^{(1)}}_{d_k^{(1)}\times 1}, \underbrace{k_l^{(2)}}_{d_k^{(2)}\times 1}). \quad (11)$$

From the attention scores, attention weights are created via an alignment function, and are used to produce the context vectors $c^{(1)}$ and $c^{(2)}$.

While coarse-grained co-attention mechanisms use a compact representation of one input to use as a query when calculating attention for another input, fine-grained co-attention considers every element of each input individually when calculating attention scores. In this case, the query becomes a matrix. An example of fine-grained co-attention is **parallel co-attention** [32]. Similarly to interactive co-attention, parallel co-attention calculates attention on the two feature matrices at the same time, as shown in Fig. 6. We start by evaluating the keys matrices $K^{(1)} \in \mathbb{R}^{d_k^{(1)} \times n_f^{(1)}}$ and $K^{(2)} \in \mathbb{R}^{d_k^{(2)} \times n_f^{(2)}}$ that are obtained by linearly transforming the feature matrices $F^{(1)}$ and $F^{(2)}$, where $d_k^{(1)}$ and $d_k^{(2)}$ are prespecified dimensions of the keys. The idea is to use the keys matrix from one input as the query for calculating attention on the other input. However, since $K^{(1)}$ and $K^{(2)}$ have completely different dimensions, an affinity matrix $A \in \mathbb{R}^{n_f^{(1)} \times n_f^{(2)}}$ is calculated that is used to essentially translate one keys matrix to the space of the other keys. In [32], A is calculated as shown in (12).

$$\underbrace{A}_{n_f^{(1)} \times n_f^{(2)}} = \text{act}(\underbrace{K^{(1)^T}}_{n_f^{(1)} \times d_k^{(1)}} \times \underbrace{W_A}_{d_k^{(1)} \times d_k^{(2)}} \times \underbrace{K^{(2)}}_{d_k^{(2)} \times n_f^{(2)}}), \tag{12}$$

where $W_A \in \mathbb{R}^{d_k^{(1)} \times d_k^{(2)}}$ is a trainable weights matrix and act() is an activation function for which the tanh() function is used in [32]. [35] proposes a different way of calculating this matrix, i.e., one can use (13) to calculate each individual element $A_{i,j}$ of the matrix A.

$$\underbrace{A_{i,j}}_{1 \times 1} = \underbrace{w_A^T}_{1 \times 3d_k} \times \text{concat}(\underbrace{k_i^{(1)}}_{d_k \times 1}, \underbrace{k_j^{(2)}}_{d_k \times 1}, \underbrace{k_i^{(1)}}_{d_k \times 1} \circ \underbrace{k_j^{(2)}}_{d_k \times 1}), \tag{13}$$

where $w_A \in \mathbb{R}^{3d_k}$ denotes a trainable vector of weights, concat() denotes vector concatenation, and $\circ$ denotes element-wise multiplication, also known as the Hadamard product. Note that the keys of each keys matrix in this case must have the same dimension d_k for the element-wise multiplication to work. The affinity matrix can be interpreted as a similarity matrix for the columns of the two keys matrices, and helps translate, for example, image keys to the same space as the keys of the words in a sentence, and vice versa. The vectors of attention scores $e^{(1)}$ and $e^{(2)}$ can be calculated using an altered version of the additive score function as presented in (14) and (15). The previous attention score examples in this survey all used a score function to calculate each attention score for each value vector individually. However, (14) and (15) are used to calculate the complete vector of all attention scores. Essentially, the attention scores are calculated in an aggregated form.

$$\underbrace{e^{(1)}}_{1 \times n_f^{(1)}} = \underbrace{w_1}_{1 \times d_w} \times \text{act}(\underbrace{W_2}_{d_w \times d_k^{(2)}} \times \underbrace{K^{(2)}}_{d_k^{(2)} \times n_f^{(2)}} \times \underbrace{A^T}_{n_f^{(2)} \times n_f^{(1)}} + \underbrace{W_1}_{d_w \times d_k^{(1)}} \times \underbrace{K^{(1)}}_{d_k^{(1)} \times n_f^{(1)}}), \tag{14}$$

$$\underbrace{e^{(2)}}_{1 \times n_f^{(2)}} = \underbrace{w_2}_{1 \times d_w} \times \text{act}(\underbrace{W_1}_{d_w \times d_k^{(1)}} \times \underbrace{K^{(1)}}_{d_k^{(1)} \times n_f^{(1)}} \times \underbrace{A}_{n_f^{(1)} \times n_f^{(2)}} + \underbrace{W_2}_{d_w \times d_k^{(2)}} \times \underbrace{K^{(2)}}_{d_k^{(2)} \times n_f^{(2)}}), \tag{15}$$

where $w_1 \in \mathbb{R}^{d_w}$, $w_2 \in \mathbb{R}^{d_w}$, $W_1 \in \mathbb{R}^{d_w \times d_k^{(1)}}$, and $W_2 \in \mathbb{R}^{d_w \times d_k^{(2)}}$ are trainable weight matrices, for which d_w is a prespecified dimension of the weight matrices. Note that tanh() is used in [32] for the activation function, and the feature matrices are used as the key matrices. In that case, the affinity matrix A can be seen as a translator between feature spaces. As mentioned before, the affinity matrix is essentially a similarity matrix for the key vectors of the two inputs. In [33], this fact is used to propose a different way of determining attention scores. Namely, one could take the maximum similarity value in a row or column as the attention score, as shown in (16).

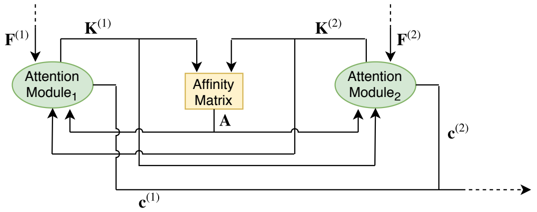

Fig. 6. An illustration of parallel co-attention.

$$\underbrace{e_i^{(1)}}_{1 \times 1} = \max_{j=1,\ldots,n_f^{(2)}} \underbrace{A_{i,j}}_{1 \times 1}, \qquad \underbrace{e_j^{(2)}}_{1 \times 1} = \max_{i=1,\ldots,n_f^{(1)}} \underbrace{A_{i,j}}_{1 \times 1}. \tag{16}$$

Next, the attention scores are used to calculate attention weights using an alignment function, so that two context vectors $c^{(1)}$ and $c^{(2)}$ can be derived as weighted averages of the value vectors that are obtained from linearly transforming the features. For the alignment function, [32] proposes to use a softmax function, and the value vectors are simply set equal to the feature vectors. The resulting context vectors can be either concatenated or added together.

Finally, coarse-grained and fine-grained co-attention can be combined to create an even more complex co-attention mechanism. [33] proposes the **multi-grained co-attention** mechanism that calculates both coarse-grained and fine-grained co-attention for two inputs. Each mechanism produces one context vector per input. The four resulting context vectors are concatenated and used in the output model for prediction.

A mechanism separate from co-attention that still uses multiple inputs is the **rotatory attention** mechanism [36]. This technique is typically used in a text sentiment analysis setting where there are three inputs involved: the phrase for which the sentiment needs to be determined (target phrase), the text before the target phrase (left context), and the text after the target phrase (right context). The words in these three inputs are all encoded by the feature model, producing the following feature matrices: $F^t = [f_1^t, \ldots, f_{n_f^t}^t] \in \mathbb{R}^{d_f^t \times n_f^t}$, $F^l = [f_1^l, \ldots, f_{n_f^l}^l] \in \mathbb{R}^{d_f^l \times n_f^l}$, and $F^r = [f_1^r, \ldots, f_{n_f^r}^r] \in \mathbb{R}^{d_f^r \times n_f^r}$, for the target phrase words, left context words, and right context words, respectively, where d_f^t, d_f^l, and d_f^r represent the dimensions of the feature vectors for the corresponding inputs, and n_f^t, n_f^l, and n_f^r represent the number of feature vectors for the corresponding inputs. The feature model used in [36] consists of word embeddings and separate Bi-LSTM models for the target phrase, the left context, and the right context. This means that the feature vectors are in fact the hidden state vectors obtained from the Bi-LSTM models. Using these features, the idea is to extract a single vector r from the inputs such that a softmax layer can be used for classification. As such, we are now faced with two challenges: how to represent the inputs as a single vector, and how to incorporate the information from the left and right context into that vector. [36] proposes to use the rotatory attention mechanism for this purpose.

Firstly, a single target phrase representation is created by using a pooling layer that takes the average over the columns of F^t, as shown in (17).

$$\underset{d_f^t \times 1}{r^t} = \frac{1}{n_f^t} \sum_{i=1}^{n_f^t} \underset{d_f^t \times 1}{f_i^t}. \tag{17}$$

r^t is then used as a query to create a context vector out of the left and right contexts, separately. For example, for the left context, the key vectors $k_1^l, \ldots, k_{n_f^l}^l \in \mathbb{R}^{d_k^l}$ and value vectors $v_1^l, \ldots, v_{n_f^l}^l \in \mathbb{R}^{d_v^l}$ are extracted from the left context feature vectors $f_1^l, \ldots, f_{n_f^l}^l \in \mathbb{R}^{d_f^l}$, similarly as before, where d_k^l and d_v^l are the dimensions of the key and value vectors, respectively. Note that [36] proposes to use the original feature vectors as keys and values, meaning that the linear transformation consists of a multiplication by an identity matrix. Next, the scores are calculated using (18).

$$\underset{1 \times 1}{e_i^l} = \text{score}(\underset{d_f^t \times 1}{r^t}, \underset{d_k^l \times 1}{k_i^l}). \tag{18}$$

For the score function, [36] proposes to use an activated general score function [34] with a tanh activation function. The attention scores can be combined with an alignment function and the corresponding value vectors to produce the context vector $r^l \in \mathbb{R}^{d_v^l}$. The alignment function used in [36] takes the form of a softmax function. An analogous procedure can be performed to obtain the representation of the right context, r^r. These two context representations can then be used to create new representations of the target phrase, again, using attention. Firstly, the key vectors $k_1^t, \ldots, k_{n_f^t}^t \in \mathbb{R}^{d_k^t}$ and value vectors $v_1^t, \ldots, v_{n_f^t}^t \in \mathbb{R}^{d_v^t}$ are extracted from the target phrase feature vectors $f_1^t, \ldots, f_{n_f^t}^t \in \mathbb{R}^{d_f^t}$, similarly as before, using a linear transformation, where d_k^t and d_v^t are the dimensions of the key and value vectors, respectively. Note, again, that the original feature vectors as keys and values in [36]. The attention scores for the left-aware target representation are then calculated using (19).

$$\underset{1 \times 1}{e_i^{l_t}} = \text{score}(\underset{d_v^l \times 1}{r^l}, \underset{d_k^t \times 1}{k_i^t}). \tag{19}$$

The attention scores can be combined with an alignment function and the corresponding value vectors to produce the context vector $r^{l_t} \in \mathbb{R}^{d_v^t}$. For this attention calculation, [34] proposes to use the same score and alignment functions as before. The right-aware target representation r^{r_t} can be calculated in a similar manner. Finally, to obtain the full representation vector r that is used to determine the classification, the vectors r^l, r^r, r^{l_t}, and r^{r_t} are concatenated together, as shown in (20).

$$\underset{(d_v^l + d_v^r + d_v^t + d_v^t) \times 1}{r} = \text{concat}(\underset{d_v^l \times 1}{r^l}, \underset{d_v^r \times 1}{r^r}, \underset{d_v^t \times 1}{r^{l_t}}, \underset{d_v^t \times 1}{r^{r_t}}). \tag{20}$$

To summarize, rotatory attention uses the target phrase to compute new representations for the left and right context using attention, and then uses these left and right representations to calculate new representations for the target phrase. The first step is designed to capture the words in the left and right contexts that are most important to the target phrase. The second step is there to capture the most important information in the actual target phrase itself. Essentially, the mechanism rotates attention between the target and the contexts to improve the representations.

There are many applications where combining information from different inputs into a single model can be highly beneficial. For example, in the field of medical data, there are often many different types of data available, such as various scans or documents, that can provide different types of information. In [37], a co-attention mechanism is used for automatic medical report generation to attend to both images and semantic tags simultaneously. Similarly, in [38], a co-attention model is proposed that combines general demographics features and patient medical history features to predict future health information. Additionally, an ablation study is used in [38] to show that the co-attention part of the model specifically improves performance. A field where multi-feature attention has been extensively explored is the domain of recommender systems. For example, in [39], a co-attention network is proposed that attends to both product reviews and the reviews a user has written. In [40], a model is proposed for video recommendation that attends to both user features and video features. Co-attention techniques have also been used in combination with graph networks for the purpose of, for example, reading comprehension across multiple documents [41] and fake news detection [42]. In comparison to co-attention, rotatory attention has typically been explored only in the field of sentiment analysis, which is most likely due to the specific structure of the data that is necessary to use this technique. An implementation of rotatory attention is proposed in [43] for sentiment analysis, where the mechanism is extended by repeating the attention rotation to iteratively further improve the representations.

3.1.2 Feature Levels

The previously discussed attention mechanisms process data at a single level. We refer to these attention techniques as **single-level attention** mechanisms. However, some data types can be analyzed and represented on multiple levels. For example, when analyzing documents, one can analyze the document at the sentence level, word level, or even the character level. When representations or embeddings of all these levels are available, one can exploit the extra levels of information. For example, one could choose to perform translation based on either just the characters, or just the words of the sentence. However, in [44], a technique named **attention-via-attention** is introduced that allows one to incorporate information from both the character, and the word levels. The idea is to predict the sentence translation character-by-character, while also incorporating information from a word-level attention module.

To begin with, a feature model (consisting of, for example, word embeddings and RNNs) is used to encode the input sentence into both a character-level feature matrix $F^{(c)} \in \mathbb{R}^{d_f^{(c)} \times n_f^{(c)}}$, and a word-level feature matrix $F^{(w)} \in \mathbb{R}^{d_f^{(w)} \times n_f^{(w)}}$, where $d_f^{(c)}$ and $n_f^{(c)}$ represent, respectively, the dimension of the embeddings of the characters, and the number of characters, while $d_f^{(w)}$ and $n_f^{(w)}$ represent the same but at the word level. It is crucial for this method that each level in the data can be represented or embedded. When attempting to predict a character in the translated

Fig. 7. An illustration of attention-via-attention.

sentence, a query $q^{(c)} \in \mathbb{R}^{d_q}$ is created by the query model (like a character-level RNN), where d_q is the dimension of the query vectors. As illustrated in Fig. 7, the query is used to calculate attention on the word-level feature vectors $F^{(w)}$. This generates the context vector $c^{(w)} \in \mathbb{R}^{d_v^{(w)}}$, where $d_v^{(w)}$ represents the dimension of the value vectors for the word-level attention module. This context vector summarizes which words contain the most important information for predicting the next character. If we know which words are most important, then it becomes easier to identify which characters in the input sentence are most important. Thus, the next step is to attend to the character-level features in $F^{(c)}$, with an additional query input: the word-level context vector $c^{(w)}$. The actual query input for the attention model will therefore be the concatenation of the query $q^{(c)}$ and the word context vector $c^{(w)}$. The output of this character-level attention module is the context vector $c^{(c)}$. The complete context output of the attention model is the concatenation of the word-level, and character-level context vectors.

The attention-via-attention technique uses representations for each level. However, accurate representations may not always be available for each level of the data, or it may be desirable to let the model create the representations during the process by building them from lower level representations. A technique referred to as hierarchical attention [5] can be used in this situation. Hierarchical attention is another technique that allows one to apply attention on different levels of the data. Yet, the exact mechanisms work quite differently compared to attention-via-attention. The idea is to start at the lowest level, and then create representations, or summaries, of the next level using attention. This process is repeated till the highest level is reached. To make this a little clearer, suppose one attempts to create a model for document classification, similarly to the implementation from [5]. We analyze a document containing n_S sentences, with the sth sentence containing n_s words, for $s = 1, \ldots, n_S$. One could use attention based on just the collection of words to classify the document. However, a significant amount of important context is then left out of the analysis, since the model will consider all words as a single long sentence, and will therefore not consider the context within the separate sentences. Instead, one can use the hierarchical structure of a document (words form sentences, and sentences form the document).

Fig. 8 illustrates the structure of hierarchical attention. For each sentence in the document, a sentence representation $c^{(s)} \in \mathbb{R}^{d_v^{(S)}}$ is produced, for $s = 1, \ldots, n_S$, where $d_v^{(S)}$ is the dimension of the value vectors used in the attention model for sentence representations (Attention Module $_S$ in Fig. 8). The representation is a context vector from an attention module that essentially summarizes the sentence. Each sentence is first put through a feature model to extract the feature matrix $F^{(s)} \in \mathbb{R}^{d_f^{(S)} \times n_s}$, for $s = 1, \ldots, n_S$, where $d_f^{(S)}$ represents the dimension of the feature vector for each word, and n_s represents the amount of words in sentence s. For extra clarification, the columns of $F^{(s)}$ are feature vectors that correspond to the words in sentence s. As shown in Fig. 8, each feature matrix $F^{(s)}$ is used as input for an attention model, which produces the context vector $c^{(s)}$, for each $s = 1, \ldots, n_S$. No queries are used in this step, so it can be considered a self-attentive mechanism. The context vectors are essentially summaries of the words in the sentences. The matrix of context vectors $C = [c^{(1)}, \ldots, c^{(n_S)}] \in \mathbb{R}^{d_v^{(S)} \times n_S}$ is constructed by grouping all the obtained context vectors together as columns. Finally, attention is calculated using C as feature input, producing the representation of the entire document in the context vector $c^{(D)} \in \mathbb{R}^{d_v^{(D)}}$, where $d_v^{(D)}$ is the dimension of the value vectors in the attention model for document representation (Attention Module $_D$ in Fig. 8). This context vector can be used to classify the document, since it is essentially a summary of all the sentences (and therefore also the words) in the document.

Fig. 8. An illustration of hierarchical attention.

Multi-level models can be used in a variety of tasks. For example, in [28], hierarchical attention is used in a recommender system to model user preferences at the long-term level and the short-term level. Similarly, [45] proposes a hierarchical model for recommending social media images based on user preferences. Hierarchical attention has also been successfully applied in other domains. For example, [46] proposes to use hierarchical attention in a video action recognition model to capture motion information at the the long-term level and the short-term level. Furthermore, [47] proposes a hierarchical attention model for cross-domain sentiment classification. In [48], a hierarchical attention model for chatbot response generation is proposed. Lastly, using image data, [49] proposes a hierarchical attention model for crowd counting.

3.1.3 Feature Representations

In a basic attention model, a single embedding or representation model is used to produce feature representations for the model to attend to. This is referred to as single-representational attention . Yet, one may also opt to incorporate multiple representations into the model. In [50], it is argued that allowing a model access to multiple embeddings can allow one to create even higher quality representations. Similarly, [51] incorporates multiple representations of the same book (textual, syntactic, semantic, visual etc.) into the feature model. Feature representations are an important part of the attention model, but attention can also be an important part of the feature model. The idea is to create a new representation by taking a weighted average of multiple representations, where the weights are determined via attention. This technique is referred to as multi-representational attention, and allows one to create so-called meta-embeddings. Suppose one wants to create a meta-embedding for a word x for which E embeddings $x^{(e_1)}, \ldots, x^{(e_E)}$ are available. Each embedding $x^{(e_i)}$ is of size d_{e_i}, for $i = 1, \ldots, E$. Since not all embeddings are of the same size, a transformation is performed to normalize the embedding dimensions. Using embedding-specific weight parameters, each embedding $x^{(e_i)}$ is transformed into the size-normalized embedding $x^{(t_i)} \in \mathbb{R}^{d_t}$, where d_t is the size of every transformed word embedding, as shown in (21).

$$\underset{d_t \times 1}{x^{(t_i)}} = \underset{d_t \times d_{e_i}}{W_{e_i}} \times \underset{d_{e_i} \times 1}{x^{(e_i)}} + \underset{d_t \times 1}{b_{e_i}}, \tag{21}$$

where $W_{e_i} \in \mathbb{R}^{d_t \times d_{e_i}}$, and $b_{e_i} \in \mathbb{R}^{d_t}$ are trainable, embedding-specific weights matrices. The final embedding $x^{(e)} \in \mathbb{R}^{d_t}$ is a weighted average of the previously calculated transformed representations, as shown in (22).

$$\underset{d_t \times 1}{x^{(e)}} = \sum_{i=1}^{E} \underset{1 \times 1}{a_i} \times \underset{d_t \times 1}{x^{(t_i)}}. \tag{22}$$

The final representation $x^{(e)}$ can be interpreted as the context vector from an attention model, meaning that the weights $a_1, \ldots, a_E \in \mathbb{R}^1$ are attention weights. Attention can be calculated as normally, where the columns of the features matrix F are the transformed representations $x^{(t_1)}, \ldots, x^{(t_E)}$. The query in this case can be ignored since it is constant in all cases. Essentially, the query is “Which representations are the most important?” in every situation. As such, this is a self-attentive mechanism.

While an interesting idea, applications of multi-representational attention are limited. One example of the application of this technique is found in [52], where a multi-representational attention mechanism has been applied to generate multi-lingual meta-embeddings. Another example is [53], where a multi-representational text classification model is proposed that incorporates different representations of the same text. For example, the proposed model uses embeddings from part-of-speech tagging, named entity recognizers, and character-level and word-level embeddings.

3.2 General Attention Mechanisms

This major category consists of attention mechanisms that can be applied in any type of attention model. The structure of this component can be broken down into the following sub-aspects: the attention score function, the attention alignment, and attention dimensionality.

3.2.1 Attention Scoring

The attention score function is a crucial component in how attention is calculated. Various approaches have been developed that each have their own advantages and disadvantages. An overview of these functions is provided in Table 2.

TABLE 2
Overview of score function ($\text{score}(q, k_l)$) forms.

Name	Function	Parameters
Additive (Concatenate) [3]	$w^T \times \text{act}(W_1 \times q + W_2 \times k_l) + b)$	$w \in \mathbb{R}^{d_w}$ $W_1 \in \mathbb{R}^{d_w \times d_q}$ $W_2 \in \mathbb{R}^{d_w \times d_k}$ $b \in \mathbb{R}^{d_w}$
Multiplicative (Dot-Product) [4]	$q^T \times k_l$	-
Scaled Multiplicative [13]	$\frac{q^T \times k_l}{\sqrt{d_k}}$	-
General [4]	$k_l^T \times W \times q$	$W \in \mathbb{R}^{d_k \times d_q}$
Biased General [54]	$k_l^T \times (W \times q + b)$	$W \in \mathbb{R}^{d_k \times d_q}$ $b \in \mathbb{R}^{d_k}$
Activated General [34]	$\text{act}(k_l^T \times W \times q + b)$	$W \in \mathbb{R}^{d_k \times d_q}$, $b \in \mathbb{R}^1$
Similarity [55]	$\text{similarity}(q, k_l)$	-

Each row of Table 2 presents a possible form for the function $\text{score}(q, k_l)$, as seen in (23), where q is the query vector, and k_l is the lth column of K. Note that the score functions presented in this section can be more efficiently calculated in matrix form using K instead of each column separately. Nevertheless, the score functions are presented using k_l to more clearly illustrate the relation between a key and query.

$$\underset{1 \times 1}{e_l} = \text{score}(\underset{d_q \times 1}{q}, \underset{d_k \times 1}{k_l}). \tag{23}$$

Due to their simplicity, the most popular choices for the score function are the concatenate score function [3] and the multiplicative score function [4]. The multiplicative score function has the advantage of being computationally inexpensive due to highly optimized vector operations. However, the multiplicative function may produce non-optimal results when the dimension d_k is too large [56]. When d_k is large, the dot-product between q and k_l can grow large in magnitude. To illustrate this, in [13], an example is used where the elements of q and k_l are all normally distributed with a mean equal to zero, and a variance equal to one. Then, the dot-product of the vectors has a variance of d_k. A higher variance means a higher chance of numbers that are large in magnitude. When the softmax function of the alignment step is then applied using these large numbers, the gradient will become very small, meaning the model will have trouble converging [13]. To adjust for this, [13] proposes to scale the multiplicative function by the factor $\frac{1}{\sqrt{d_k}}$, producing the scaled multiplicative score function.

In [4], the multiplicative score function is extended by introducing a weights matrix W. This form, referred to as the general score function, allows for an extra transformation of k_l. The biased general score function [54] is a further extension of the general function that introduces a bias weight vector b. A final extension on this function named the activated general score function is introduced in [34], and includes the use of both a bias weight b, and an activation function $\text{act}()$.

The previously presented score functions are all based on determining a type of similarity between the key vector and the query vector. As such, more typical similarity measures, such as the Euclidean (L_2) distance and cosine similarity, can also be implemented [55]. These scoring methods are summarized under the similarity score function which is represented by the $\text{similarity}()$ function.

There typically is no common usage across domains regarding score functions. The choice of score function for a particular task is most often based on empirical experiments. However, there are exceptions when, for example, efficiency is vital. In models where this is the case, the multiplicative or scaled multiplicative score functions are typically the best choice. An example of this is the Transformer model, which is generally computationally expensive.

3.2.2 Attention Alignment

The attention alignment is the step after the attention scoring. This alignment process directly determines which parts of the input data the model will attend to. The alignment function is denoted as $\text{align}()$ and has various forms. The $\text{align}()$ function takes as input the previously calculated attention score vector $\mathbf{e}$ and calculates for each element e_l of $\mathbf{e}$ the attention weight a_l. These attention weights can then be used to create the context vector $\mathbf{c}$ by taking a weighted average of the value vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{n_f}$:

$$\underset{d_v\times 1}{\mathbf{c}} = \sum_{l=1}^{n_f} \underset{1\times 1}{a_l} \times \underset{d_v\times 1}{\mathbf{v}_l}. \tag{24}$$

The most popular alignment method to calculate these weights is a simple softmax function, as depicted in (25).

$$\underset{1\times 1}{a_l} = \text{align}(\underset{1\times 1}{e_l}; \underset{n_f\times 1}{\mathbf{e}}) = \frac{\exp(e_l)}{\sum_{j=1}^{n_f}\exp(e_j)}. \tag{25}$$

This alignment method is often referred to as "soft alignment" in computer vision settings [8], or "global alignment" for sequence data [4]. Nevertheless, both these terms represent the same function and can be interpreted similarly. Soft/global alignment can be interpreted as the model attending to all feature vectors. For example, the model attends to all regions in an image, or all words in a sentence. Even though the attention model generally does focus more on specific parts of the input, every part of the input will receive at least some amount of attention due to the nature of the softmax function. Furthermore, an advantage of the softmax function is that it introduces a probabilistic interpretation to the input vectors. This allows one to easily analyze which parts of the input are important to the output predictions.

In contrast to soft/global alignment, other methods aim to achieve a more focused form of alignment. For example, "hard alignment" [8], also known as hard attention or non-deterministic attention, is an alignment type that forces the attention model to focus on exactly one feature vector. Firstly, this method implements the softmax function in the exact same way as global alignment. However, the outputs $a_1, \ldots, a_{n_f}$ are not used as weights for the context vector calculation. Instead, these values are used as probabilities to draw the choice of the one value vector from. A value $m \in \mathbb{R}^1$ is drawn from a multinomial distribution with $a_1, \ldots, a_{n_f}$ as parameters for the probabilities. Then, the context vector is simply defined as follows:

$$\underset{d_v\times 1}{\mathbf{c}} = \underset{d_v\times 1}{\mathbf{v}_m}. \tag{26}$$

Hard alignment is typically more efficient at inference compared to soft alignment. On the other hand, the main disadvantage of hard attention is that, due to the stochastic alignment of attention, the training of the model cannot be done via the regular backpropagation method. Instead, simulation and sampling, or reinforcement learning [57] are required to calculate the gradient at the hard attention layer. As such, soft/global attention is generally preferred. However, a compromise can be made in certain situations. "Local alignment" [4] is a method that implements a softmax distribution, similarly to soft/global alignment. But, the softmax distribution is calculated based only on a subset of the inputs. This method is generally used in combination with sequence data. One has to specify a variable $p \in \mathbb{R}^1$ that determines the position of the region. Feature vectors close to p will be attended to by the model, and vectors too far from p will be ignored. The size of the subset will be determined by the variable $D \in \mathbb{R}^1$. Summarizing, the attention model will apply a softmax function on the attention scores in the subset $[p - D, p + D]$. In other words, a window is placed on the input and soft/global attention is calculated within that window:

$$\underset{1\times 1}{a_l} = \text{align}(\underset{1\times 1}{e_l}; \underset{n_f\times 1}{\mathbf{e}}) = \frac{\exp(e_l)}{\sum_{j=p-D}^{p+D}\exp(e_j)}. \tag{27}$$

The question that remains is how to determine the location parameter p. The first method is referred to as "monotonic alignment". This straightforward method entails simply setting the location parameter equal to the location of the prediction in the output sequence. Another method of determining the position of the region is referred to as "predictive alignment". As the name entails, the model attempts to actually predict the location of interest in the sequence:

$$\underset{1\times 1}{p} = \underset{1\times 1}{S} \times \text{sigmoid}(\underset{1\times d_p}{\mathbf{w}_p^T} \times \tanh(\underset{d_p\times d_q}{\mathbf{W}_p} \times \underset{d_q\times 1}{\mathbf{q}})), \tag{28}$$

where $S \in \mathbb{R}^1$ is the length of the input sequence, and $\mathbf{w}_p \in \mathbb{R}^{d_p}$ and $\mathbf{W}_p \in \mathbb{R}^{d_p\times d_q}$ are both trainable weights parameters. The sigmoid function multiplied by S makes sure that p is in the range $[0, S]$. Additionally, in [4], it is recommended to add an additional term to the alignment function to favor alignment around p:

$$\underset{1\times 1}{a_l} = \text{align}(\underset{1\times 1}{e_l}; \underset{n_f\times 1}{\mathbf{e}})\exp(-\frac{(l-p)^2)}{2\sigma^2}), \tag{29}$$

where $\sigma \in \mathbb{R}^1$ is empirically set equal to $\frac{D}{2}$ according to [4]. Another proposed method for compromising between soft and hard alignment is "reinforced alignment" [58]. Similarly to local alignment, a subset of the feature vectors is determined, for which soft alignment is calculated. However, instead of using a window to determine the subset, reinforced alignment uses a reinforcement learning agent [57], similarly to hard alignment, to choose the subset of feature vectors. The attention calculation based on these chosen feature vectors is the same as regular soft alignment.

Soft alignment is often regarded as the standard alignment function for attention models in practically every domain. Yet, the other alignment methods have also seen interesting uses in various domains. For example, hard attention is used in [59] for the task of visual question answering. In [60], both soft and hard attention are used in a graph attention model for multi-agent game abstraction. Similarly, in [61], both global and local alignment are used for review rating predictions. Reinforced alignment has been employed in combination with a co-attention structure in [62] for the task of aspect sentiment classification. In [63], reinforced alignment is used for the task of person re-identification using surveillance images.

3.2.3 Attention Dimensionality

All previous model specifications of attention use a scalar weight a_l for each value vector $\boldsymbol{v}_l$. This technique is referred to as single-dimensional attention. However, instead of determining a single attention score and weight for the entire vector, [64] proposes to calculate weights for every single feature in those vectors separately. This technique is referred to as multi-dimensional attention, since the attention weights now become higher dimensional vectors. The idea is that the model no longer has to attend to entire vectors, but it can instead pick and choose specific elements from those vectors. More specifically, attention is calculated for each dimension. As such, the model must create a vector of attention weights $\boldsymbol{a}_l \in \mathbb{R}^{d_v}$ for each value vector $\boldsymbol{v}_l \in \mathbb{R}^{d_v}$. The context vector can then be calculated by summing the element-wise multiplications ($\circ$) of the value vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_{n_f} \in \mathbb{R}^{d_v}$ and the corresponding attention weight vectors $\boldsymbol{a}_1, \dots, \boldsymbol{a}_{n_f} \in \mathbb{R}^{d_v}$, as follows:

$$\underset{d_v\times 1}{\boldsymbol{c}} = \sum_{l=1}^{n_f} \underset{d_v\times 1}{\boldsymbol{a}_l} \circ \underset{d_v\times 1}{\boldsymbol{v}_l}. \tag{30}$$

However, since one needs to create attention weight vectors, this technique requires adjusted attention score and weight calculations. For example, the concatenate score function found in Table 2 can be adjusted by changing the $\boldsymbol{w} \in \mathbb{R}^{d_w}$ weights vector to the weight matrix $\boldsymbol{W}_d \in \mathbb{R}^{d_w \times d_v}$:

$$\underset{d_v\times 1}{\boldsymbol{e}_l} = \underset{d_v\times d_w}{\boldsymbol{W}_d^T} \times \text{act}(\underset{d_w\times d_q}{\boldsymbol{W}_1} \times \underset{d_q\times 1}{\boldsymbol{q}} + \underset{d_w\times d_k}{\boldsymbol{W}_2} \times \underset{d_k\times 1}{\boldsymbol{k}_l} + \underset{d_w\times 1}{\boldsymbol{b}}). \tag{31}$$

This new score function produces the attention score vectors $\boldsymbol{e}_1, \dots, \boldsymbol{e}_{n_f} \in \mathbb{R}^{d_v}$. These score vectors can be combined into a matrix of scores $\boldsymbol{e} = [\boldsymbol{e}_1, \dots, \boldsymbol{e}_{n_f}] \in \mathbb{R}^{d_v \times n_f}$. To produce multi-dimensional attention weights, the alignment function stays the same, but it is applied for each feature across the attention score columns. To illustrate, when implementing soft attention, the attention weight produced from the ith element of score vector $\boldsymbol{e}_l$ is defined as follows:

$$\underset{1\times 1}{a_{l,i}} = \text{align}(\underset{1\times 1}{e_{l,i}}; \underset{d_v\times n_f}{\boldsymbol{e}}) = \frac{\exp(e_{l,i})}{\sum_{j=1}^{n_f} \exp(e_{j,i})}, \tag{32}$$

where $e_{l,i}$ represents the ith element of score vector $\boldsymbol{e}_l$, and $a_{l,i}$ is the ith element of the attention weights vector $\boldsymbol{a}_l$. Finally, these attention weight vectors can be used to compute the context vector as presented in (30).

Multi-dimensional attention is a very general mechanism that can be applied in practically every attention model, but actual applications of the technique have been relatively sparse. One application example is [65], where multi-dimensional attention is used in a model for named entity recognition based on text and visual context from multimedia posts. In [66], multi-dimensional attention is used in a model for answer selection in community question answering. In [67], the U-net model for medical image segmentation is extended with a multi-dimensional attention mechanism. Similarly, in [68], the Transformer model is extended with the multi-dimensional attention mechanism for the task of dialogue response generation. In [69], multi-dimensional attention is used to extend graph attention networks for dialogue state tracking. Lastly, for the task of next-item recommendation, [70] proposes a model that incorporates multi-dimensional attention.

3.3 Query-Related Attention Mechanisms

Queries are an important part of any attention model, since they directly determine which information is extracted from the feature vectors. These queries are based on the desired output of the task model, and can be interpreted as literal questions. Some queries have specific characteristics that require specific types of mechanisms to process them. As such, this category encapsulates the attention mechanisms that deal with specific types of query characteristics. The mechanisms in this category deal with one of the two following query characteristics: the type of queries or the multiplicity of queries.

3.3.1 Type of Queries

Different attention models employ attention for different purposes, meaning that distinct query types are necessary. There are basic queries, which are queries that are typically straightforward to define based on the data and model. For example, the hidden state for one prediction in an RNN is often used as the query for the next prediction. One could also use a vector of auxiliary variables as query. For example, when doing medical image classification, general patient characteristics can be incorporated into a query.

Some attention mechanisms, such as co-attention, rotatory attention, and attention-over-attention, use specialized queries. For example, rotatory attention uses the context vector from another attention module as query, while interactive co-attention uses an averaged keys vector based on another input. Another case one can consider is when attention is calculated based purely on the feature vectors. This concept has been mentioned before and is referred to as self-attention or intra-attention [71]. We say that the models use self-attentive queries. There are two ways of interpreting such queries. Firstly, one can say that the query is constant. For example, document classification requires only a single classification as the output of the model. As such, the query is always the same, namely: “What is the class of the document?”. The query can be ignored and attention can be calculated based only on the features themselves. Score functions can be adjusted for this by making the query vector a vector of constants or removing it entirely:

$$\text{score}(\underset{d_k\times 1}{\boldsymbol{k}_l}) = \underset{1\times d_w}{\boldsymbol{w}^T} \times \text{act}(\underset{d_w\times d_k}{\boldsymbol{W}} \times \underset{d_k\times 1}{\boldsymbol{k}_l} + \underset{d_w\times 1}{\boldsymbol{b}}). \tag{33}$$

Additionally, one can also interpret self-attention as learning the query along the way, meaning that the query can be defined as a trainable vector of weights. For example, the dot-product score function may take the following form:

$$\text{score}(\underset{d_k\times 1}{\boldsymbol{k}_l}) = \underset{1\times d_k}{\boldsymbol{q}^T} \times \underset{d_k\times 1}{\boldsymbol{k}_l}, \tag{34}$$

where $\boldsymbol{q} \in \mathbb{R}^{d_k}$ is a trainable vector of weights. One could also interpret vector $\boldsymbol{b} \in \mathbb{R}^{d_w}$ as the query in (33). Another use of self-attention is to uncover the relations between the feature vectors $f_1; : : : ; f_{n_f}$. These relations can then be used as additional information to incorporate into new representations of the feature vectors. With basic attention mechanisms, the keys matrix K, and the values matrix V are extracted from the features matrix F, while the query q is produced separately. For this type of self-attention, the query vectors are extracted in a similar process as the keys and values, via a transformation matrix of trainable weights W_Q 2 $R^{d_q \quad d_f}$. We define the matrix $Q = [q_1; : : : ; q_{n_f}]$ 2 $R^{d_q \quad n_f}$, which can be obtained as follows:

$$\underset{d_q \quad n_f}{Q} = \underset{d_q \quad d_f}{W_Q} \quad \underset{d_f \quad n_f}{F} : \tag{35}$$

Each column of Q can be used as the query for the attention model. When attention is calculated using a query q, the resulting context vector c will summarize the information in the feature vectors that is important to the query. Since the query, or a column of Q, is now also a feature vector representation, the context vector contains the information of all feature vectors that are important to that specific feature vector. In other words, the context vectors capture the relations between the feature vectors. For example, self-attention allows one to extract the relations between words: which verbs refer to which nouns, which pronouns refer to which nouns, etc. For images, self-attention can be used to determine which image regions relate to each other.

While self-attention is placed in the query-related category, it is also very much related to the feature model. Namely, self-attention is a technique that is often used in the feature model to create improved representations of the feature vectors. For example, the Transformer model for language processing [13], and the Transformer model for image processing [15], both use multiple rounds of (multi-head) self-attention to improve the representation of the feature vectors. The relations captured by the self-attention mechanism are incorporated into new representations. A simple method of determining such a new representation is to simply set the feature vectors equal to the acquired self-attention context vectors [71], as presented in (36).

$$\underset{d_f \quad 1}{f^{(new)}} = \underset{d_f \quad 1}{c} ; \tag{36}$$

where $f^{(new)}$ is the updated feature vector. Another possibility is to add the context vectors to the previous feature vectors with an additional normalization layer [13]:

$$\underset{d_f \quad 1}{f^{(new)}} = \text{Normalize}(\underset{d_f \quad 1}{f^{(old)}} + \underset{d_f \quad 1}{c}); \tag{37}$$

where $f^{(old)}$ is the previous feature vector, and Normalize() is a normalization layer [72]. Using such techniques, self-attention has been used to create improved word or sentence embeddings that enhance model accuracy [71].

Self-attention is arguably one of the more important types of attention, partly due to its vital role in the highly popular Transformer model. Self-attention is a very general mechanism and can be applied to practically any problem. As such, self-attention has been extensively explored in many different fields in both Transformer-based architectures and other types of models. For example, in [73], self-attention is explored for image recognition tasks, and results indicate that the technique may have substantial advantages with regards to robustness and generalization. In [74], self-attention is used in a generative adversarial network (GAN) [75] to determine which regions of the input image to focus on when generating the regions of a new image. In [76], self-attention is used to design a state-of-the-art medical image segmentation model. Naturally, self-attention can also be used for video processing. In [77], a self-attention model is proposed for the purpose of video summarization that reaches state-of-the-art results. In other fields, like audio processing, self-attention has been explored as well. In [78], self-attention is used to create a speech recognition model. Self-attention has also been explored in overlapping domains. For example, in [79], the self-attention Transformer architecture is used to create a model that can recognize phrases from audio and by lip-reading from a video. For the problem of next item recommendation, [80] proposes a Transformer model that explicitly captures item-item relations using self-attention. Self-attention also has applications in any natural language processing fields. For example, in [81], self-attention is used for sentiment analysis. Self-attention is also highly popular for graph models. For example, self-attention is explored in [82] for the purpose of representation learning in communication networks and rating networks. Additionally, the first attention model for graph networks was based on self-attention [83].

3.3.2 Multiplicity of Queries

In previous examples, the attention model generally used a single query for a prediction. We say that such models use singular query attention . However, there are attention architectures that allow the model to compute attention using multiple queries. Note that this is different from, for example, an RNN that may involve multiple queries to produce a sequence of predictions. Namely, such a model still requires only a single query per prediction.

One example of a technique that incorporates multiple queries is multi-head attention [13], as presented in Fig. 9. Multi-head attention works by implementing multiple attention modules in parallel by utilizing multiple different versions of the same query. The idea is to linearly transform the query q using different weight matrices. Each newly formed query essentially asks for a different type of relevant information, allowing the attention model to introduce more information into the context vector calculation. An attention model implements d 1 heads with each attention head having its own query vector, keys matrix, and values matrix: $q^{(j)}$, $K^{(j)}$ and $V^{(j)}$, for $j = 1; : : : ; d$. The query $q^{(j)}$ is obtained by linearly transforming the original query q, while the matrices $K^{(j)}$ and $V^{(j)}$ are obtained through linear transformations of F. As such, each attention head has its own learnable weights matrices $W_q^{(j)}$, $W_K^{(j)}$ and $W_V^{(j)}$ for these transformations. The calculation of the query, keys, and values for the j th head are defined as follows:

$$\underset{d_q \quad 1}{q^{(j)}} = \underset{d_q \quad d_q}{W_q^{(j)}} \quad \underset{d_q \quad 1}{q} ; \qquad \underset{d_k \quad n_f}{K^{(j)}} = \underset{d_k \quad d_f}{W_K^{(j)}} \quad \underset{d_f \quad n_f}{F} ;$$
$$\underset{d_v \quad n_f}{V^{(j)}} = \underset{d_v \quad d_f}{W_V^{(j)}} \quad \underset{d_f \quad n_f}{F} : \tag{38}$$

Thus, each head creates its own representations of the query q, and the input matrix F. Each head can therefore

Fig. 9. An illustration of multi-head attention.

learn to focus on different parts of the inputs, allowing the model to attend to more information. For example, when training a machine translation model, one attention head can learn to focus on which nouns (e.g., student, car, apple) do certain verbs (e.g., walking, driving, buying) refer to, while another attention head learns to focus on which nouns refer to certain pronouns (e.g., he, she, it) [13]. Each head will also create its own vector of attention scores $e^{(j)} = [e_1^{(j)}, \dots, e_{n_f}^{(j)}] \in \mathbb{R}^{n_f}$, and a corresponding vector of attention weights $a^{(j)} = [a_1^{(j)}, \dots, a_{n_f}^{(j)}] \in \mathbb{R}^{n_f}$. As can be expected, each attention model produces its own context vector $c^{(j)} \in \mathbb{R}^{d_v}$, as follows:

$$\underset{d_v \times 1}{c^{(j)}} = \sum_{l=1}^{n_f} \underset{1 \times 1}{a_l^{(j)}} \times \underset{d_v \times 1}{v_l^{(j)}}. \tag{39}$$

The goal is still to create a single context vector as output of the attention model. As such, the context vectors produced by the individual attention heads are concatenated into a single vector. Afterwards, a linear transformation is applied using the weight matrix $W_O \in \mathbb{R}^{d_c \times d_v d}$ to make sure the resulting context vector $c \in \mathbb{R}^{d_c}$ has the desired dimension. This calculation is presented in (40). The dimension d_c can be pre-specified by, for example, setting it equal to d_v, so that the context vector dimension is unchanged.

$$\underset{d_c \times 1}{c} = \underset{d_c \times d_v d}{W_O} \times \text{concat}(\underset{d_v \times 1}{c^{(1)}}, \dots, \underset{d_v \times 1}{c^{(d)}}). \tag{40}$$

Multi-head attention processes multiple attention modules in parallel, but attention modules can also be implemented sequentially to iteratively adjust the context vectors. Each of these attention modules are referred to as “repetitions” or “rounds” of attention. Such attention architectures are referred to as multi-hop attention models, also known as multi-step attention models. An important note to consider is the fact that multi-hop attention is a mechanism that has been proposed in various forms throughout various works. While the mechanism always involves multiple rounds of attention, the multi-hop implementation proposed in [84] differs from the mechanism proposed in [85] or [86]. Another interesting example is [87], where a “multi-hop” attention model is proposed that would actually be considered alternating co-attention in this survey, as explained in Subsection 3.1.1.

We present a general form of multi-hop attention that is largely a generalization of the techniques introduced in [85] and [88]. Fig. 10 provides an example implementation of a multi-hop attention mechanism. The general idea is to iteratively transform the query, and use the query to transform the context vector, such that the model can extract different information in each step. Remember that a query is similar to a literal question. As such, one can interpret the transformed queries as asking the same question in a different manner or from a different perspective, similarly to the queries in multi-head attention. The query that was previously denoted by q is now referred to as the initial query, and is denoted by $q^{(0)}$. At hop s, the current query $q^{(s)}$ is transformed into a new query representation $q^{(s+1)}$, possibly using the current context vector $c^{(s)}$ as another input, and some transformation function transform$()$:

Fig. 10. An example illustration of multi-hop attention. Solid arrows represent the base multi-hop model structure, while dotted arrows represent optional connections.

$$\underset{d_q \times 1}{q^{(s+1)}} = \text{transform}(\underset{d_q \times 1}{q^{(s)}}, \underset{d_v \times 1}{c^{(s)}}). \tag{41}$$

For the specific form of the transformation function transform$()$, [85] proposes to use a mechanism similar to self-attention. Essentially, the queries used by the question answer matching model proposed in [85] were originally based on a set of feature vectors extracted from a question. [85] also defines the original query $q^{(0)}$ as the unweighted average of these feature vectors. At each hop s, attention can be calculated on these feature vectors using the previous query $q^{(s)}$ as the query in this process. The resulting context vector of this calculation is the next query vector. Using the context vector $c^{(s)}$ instead of $q^{(s)}$ as the query for this process is also a possibility, which is similar to the LCR-Rot-hop model proposed in [43] and the multi-step model proposed in [88]. Such a connection is represented by the dotted arrows in Fig. 10. The transformation mechanism uses either the $q^{(s)}$ or the context vector $c^{(s)}$ as query, but a combination via concatenation is also possible.

Each query representation is used as input for the attention module to compute attention on the columns of the feature matrix F, as seen previously. One main difference, however, is that the context vector $c^{(s)}$ is also used as input, so that the actual query input for the attention model is the concatenation of $c^{(s)}$ and $q^{(s+1)}$. The adjusted attention score function is presented in (42). Note that the initial context vector $c^{(0)}$ is predefined. One way of doing this is by setting it equal to the unweighted average of the value vectors $v_1, \dots, v_{n_f} \in \mathbb{R}^{d_v}$ extracted from F.

$$\underset{1 \times 1}{e_l^{(s)}} = \text{score}(\text{concat}(\underset{d_q \times 1}{q^{(s+1)}}, \underset{d_v \times 1}{c^{(s)}}), \underset{d_k \times 1}{k_l}). \tag{42}$$

An alignment function and the value vectors are then used to produce the next context vector $c^{(s+1)}$. One must note that in [85], the weights used in each iteration are the same weights, meaning that the number of parameters do not scale with the number of repetitions. Yet, using multiple hops with different weight matrices can also be viable, as shown by the Transformer model [13] and in [88]. It may be difficult to grasp why $c^{(s)}$ is part of the query input for the attention model. Essentially, this technique is closely related to self-attention in the sense that, in each iteration, a new context representation is created from the feature vectors and the context vector. The essence of this mechanism is that one wants to iteratively alter the query and the context vector, while attending to the feature vectors. In the process, the new representations of the context vector absorb more different kinds of information. This is also the main difference between this type of attention and multi-head attention. Multi-head attention creates multiple context vectors from multiple queries and combines them to create a final context vector as output. Multi-hop attention iteratively refines the context vector by incorporating information from the different queries. This does have the disadvantage of having to calculate attention sequentially.

Interestingly, due to the variations in which multi-hop attention has been proposed, some consider the Transformer model's encoder and decoder to consist of several single-hop attention mechanisms [84] instead of being a multi-hop model. However, in the context of this survey, we consider the Transformer model to be an alternative form of the multi-hop mechanism, as the features matrix F is not directly reused in each step. Instead, F is only used as an input for the first hop, and is transformed via self-attention into a new representation. The self-attention mechanism uses each feature vector in F as a query, resulting in a matrix of context vectors as output of each attention hop. The intermediate context vectors are turned into matrices and represent iterative transformations of the matrix F, which are used in the consecutive steps. Thus, the Transformer model iteratively refines the features matrix F by extracting and incorporating new information.

When dealing with a classification task, another idea is to use a different query for each class. This is the basic principle behind capsule-based attention [89], as inspired by the capsule networks [90]. Suppose we have the feature vectors $f_1, \ldots, f_{n_f} \in \mathbb{R}^{d_f}$, and suppose there are are d_y classes that the model can predict. Then, a capsule-based attention model defines a capsule for each of the d_y classes that each take as input the feature vectors. Each capsule consists of, in order, an attention module, a probability module, and a reconstruction module, which are depicted in Fig. 11. The attention modules all use self-attentive queries, so each module learns its own query: "Which feature vectors are important to identify this class?". In [89], a self-attentive multiplicative score function is used for this purpose:

$$\underset{1\times 1}{e_{c,l}} = \underset{1\times d_k}{q_c^T}\ \underset{d_k\times 1}{k_l}\ ; \tag{43}$$

where $e_{c,l} \in \mathbb{R}^1$ is the attention score for vector l in capsule c, and $q_c \in \mathbb{R}^{d_k}$ is a trainable query for capsule c, for $c = 1, \ldots, d_y$. Each attention module then uses an alignment function, and uses the produced attention weights to determine a context vector $c_c \in \mathbb{R}^{d_v}$. Next, the context vector c_c is fed through a probability layer consisting of a linear transformation with a sigmoid activation function:

$$\underset{1\times 1}{p_c} = \text{sigmoid}(\underset{1\times d_v}{w_c^T}\ \underset{d_v\times 1}{c_c} + \underset{1\times 1}{b_c}); \tag{44}$$

Fig. 11. An illustration of capsule-based attention.

where $w_c \in \mathbb{R}^{d_v}$ and $b_c \in \mathbb{R}^1$ are trainable capsule-specific weights parameters, and $p_c \in \mathbb{R}^1$ is the predicted probability that the correct class is class c. The final layer is the reconstruction module that creates a class vector representation. This representation $r_c \in \mathbb{R}^{d_v}$ is determined by simply multiplying the context vector c_c by the probability p_c:

$$\underset{d_v\times 1}{r_c} = \underset{1\times 1}{p_c} \times \underset{d_v\times 1}{c_c}\ . \tag{45}$$

The capsule representation is used when training the model. First of all, the model is trained to predict the probabilities $p_1, \ldots, p_{d_y}$ as accurately as possible compared to the true values. Secondly, via a joint loss function, the model is also trained to accurately construct the capsule representations $r_1, \ldots, r_{d_y}$. A features representation $f \in \mathbb{R}^{d_f}$ is defined which is simply the unweighted average of the original feature vectors. The idea is to train the model such that vector representations from capsules that are not the correct class differ significantly from f while the representation from the correct capsule is very similar to f. A dot-product between the capsule representations and the features representation is used in [89] as a measure of the distance between the vectors. Note that d_v must equal d_f in this case, otherwise the vectors would have incompatible dimensions. Interestingly, since attention is calculated for each class individually, one can track which specific feature vectors are important for which specific class. In [89], this idea is used to discover which words correspond to which sentiment class.

The number of tasks that can make use of multiple queries is substantial, due to how general the mechanisms are. As such, the techniques described in this section have been extensively explored in various domains. For example, multi-head attention has been used for speaker recognition based on audio spectrograms [91]. In [92], multi-head attention is used for recommendation of news articles. Additionally, multi-head attention can be beneficial for graph attention models as well [83]. As for multi-hop attention, quite a few papers have been mentioned before, but there are still many other interesting examples. For example, in [93], a multi-hop attention model is proposed for medication recommendation. Furthermore, practically every Transformer model makes use of both multi-head and multi-hop attention. The Transformer model has been extensively explored in various domains. For example, in [94], a Transformer model is implemented for image captioning. In [95], Transformers are explored for medical image segmentation. In [96], a Transformer model is used for emotion recognition in text messages. A last example of an application of Transformers is [17], which proposes a Transformer model for recommender systems. In comparison with multi-head and multi-hop attention, capsule-based attention is arguably the least popular of the mechanisms discussed for the multiplicity of queries. One example is [97], where an attention-based capsule network is proposed that also includes a multi-hop attention mechanism for the purpose of visual question answering. Another example is [98], where capsule-based attention is used for aspect-level sentiment analysis of restaurant reviews.

The multiplicity of queries is a particularly interesting category due to the Transformer model [13], which combines a form of multi-hop and multi-head attention. Due to the initial success of the Transformer model, many improvements and iterations of the model have been produced that typically aim to improve the predictive performance, the computational efficiency, or both. For example, the Transformer-XL [99] is an extension of the original Transformer that uses a recurrence mechanism to not be limited by a context window when processing the outputs. This allows the model to learn significantly longer dependencies while also being computationally more efficient during the evaluation phase. Another extension of the Transformer is known as the Reformer model [100]. This model is significantly more efficient computationally, by means of locality-sensitive hashing, and memory-wise, by means of reversible residual layers. Such computational improvements are vital, since one of the main disadvantages of the Transformer model is the sheer computational cost due to the complexity of the model scaling quadratically with the amount of input feature vectors. The Linformer model [101] manages to reduce the complexity of the model to scale linearly, while achieving similar performance as the Transformer model. This is achieved by approximating the attention weights using a low-rank matrix. The Lite-Transformer model proposed in [102] achieves similar results by implementing two branches within the Transformer block that specialize in capturing global and local information. Another interesting Transformer architecture is the Synthesizer [103]. This model replaces the pairwise self-attention mechanism with “synthetic” attention weights. Interestingly, the performance of this model is relatively close to the original Transformer, meaning that the necessity of the pairwise self-attention mechanism of the Transformer model may be questionable. For a more comprehensive overview of Transformer architectures, we refer to [104].

4 Evaluation of Attention Models

In this section, we present various types of evaluation for attention models. Firstly, one can evaluate the structure of attention models using the taxonomy presented in Section 3. For such an analysis, we consider the attention mechanism categories (see Fig. 3) as orthogonal dimensions of a model. The structure of a model can be analyzed by determining which mechanism a model uses for each category. Table 3 provides an overview of attention models found in the literature with a corresponding analysis based on the attention mechanisms the models implement.

Secondly, we discuss various techniques for evaluating the performance of attention models. The performance of attention models can be evaluated using extrinsic or intrinsic performance measures, which are discussed in Subsections 4.1 and 4.2, respectively.

4.1 Extrinsic Evaluation

In general, the performance of an attention model is measured using extrinsic performance measures . For example, performance measures typically used in the field of natural language processing are the BLEU [107], METEOR [108], and Perplexity [109] metrics. In the field of audio processing, the Word Error Rate [110] and Phoneme Error Rate [111] are generally employed. For general classification tasks, error rates, precision, and recall are generally used. For computer vision tasks, the PSNR [112], SSIM [113], or IoU [114] metrics are used. Using these performance measures, an attention model can either be compared to other state-of-the-art models, or an ablation study can be performed. If possible, the importance of the attention mechanism can be tested by replacing it with another mechanism and observing whether the overall performance of the model decreases [105], [115]. An example of this is replacing the weighted average used to produce the context vector with a simple unweighted average and observing whether there is a decrease in overall model performance [35]. This ablation method can be used to evaluate whether the attention weights can actually distinguish important from irrelevant information.

4.2 Intrinsic Evaluation

Attention models can also be evaluated using attention-specific intrinsic performance measures . In [4], the attention weights are formally evaluated via the Alignment Error Rate (AER) to measure the accuracy of the attention weights with respect to annotated attention vectors. [116] incorporates this idea into an attention model by supervising the attention mechanism using gold attention vectors. A joint loss function consisting of the regular task-specific loss and the attention weights loss function is constructed for this purpose. The gold attention vectors are based on annotated text data sets where keywords are hand-labelled. However, since attention is inspired by human attention, one could evaluate attention models by comparing them to the attention behaviour of humans.

4.2.1 Evaluation via Human Attention

In [117], the concept of attention correctness is proposed, which is a quantitative intrinsic performance metric that evaluates the quality of the attention mechanism based on actual human attention behaviour. Firstly, the calculation of this metric requires data that includes the attention behaviour of a human. For example, a data set containing images with the corresponding regions that a human focuses on when performing a certain task, such as image captioning. The collection of regions focused on by the human is referred to as the ground truth region. Suppose an attention model attends to the n_f feature vectors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{n_f} \in \mathbb{R}^{d_f}$. Feature vector $\boldsymbol{f}_i$ corresponds to region R_i of the given

TABLE 3
Attention models analyzed based on the proposed taxonomy. A plus sign (+) between two mechanisms indicates that both techniques were combined in the same model, while a comma (,) indicates that both mechanisms were tested in the same paper, but not necessarily as a combination in the same model.

	Feature-Related			General			Query-Related	
	Multiplicity	Levels	Representations	Scoring	Alignment	Dimensionality	Type	Multiplicity
Bahdanau et al. [3]	Singular	Single-Level	Single-Representational	Additive	Global	Single-Dimensional	Basic	Singular
Luong et al. [4]	Singular	Single-Level	Single-Representational	Multiplicative, Location	Global, Local	Single-Dimensional	Basic	Singular
Xu et al. [8]	Singular	Single-Level	Single-Representational	Additive	Soft, Hard	Single-Dimensional	Basic	Singular
Lu et al. [32]	Parallel Co-attention	Hierarchical	Single-Representational	Additive	Global	Single-Dimensional	Specialized	Singular
Yang et al. [5]	Singular	Hierarchical	Single-Representational	Additive	Global	Single-Dimensional	Self-Attentive	Singular
Li et al. [47]	Singular	Hierarchical	Single-Representational	Additive	Global	Single-Dimensional	Self-Attentive	Singular
Vaswani et al. [13]	Singular	Single-Level	Single-Representational	Scaled-Multiplicative	Global	Single-Dimensional	Self-Attentive + Basic	Multi-Head + Multi-Hop
Wallaart and Frasincar [43]	Rotatory	Single-Level	Single-Representational	Activated General	Global	Single-Dimensional	Specialized	Multi-Hop
Kiela et al. [50]	Singular	Single-Level	Multi-Representational	Additive	Global	Single-Dimensional	Self-Attentive	Singular
Shen et al. [64]	Singular	Single-Level	Single-Representational	Additive	Global	Multi-Dimensional	Self-Attentive	Singular
Zhang et al. [74]	Singular	Single-Level	Single-Representational	Multiplicative	Global	Single-Dimensional	Self-Attentive	Singular
Li et al. [105]	Parallel Co-attention	Single-Level	Single-Representational	Scaled-Multiplicative	Global	Single-Dimensional	Self-Attentive + Specialized	Singular
Yu et al. [106]	Parallel Co-attention	Single-Level	Single-Representational	Multiplicative	Global	Single-Dimensional	Self-Attentive + Specialized	Multi-Head
Wang et al. [62]	Parallel Co-attention	Single-Level	Single-Representational	Additive	Reinforced	Single-Dimensional	Specialized	Singular
Oktay et al. [67]	Singular	Single-Level	Single-Representational	Additive	Global	Multi-Dimensional	Self-Attentive + Specialized	Singular
Winata et al. [52]	Singular	Single-Level	Multi-Representational	Additive	Global	Single-Dimensional	Self-Attentive	Multi-Head
Wang et al. [89]	Singular	Single-Level	Single-Representational	Multiplicative	Global	Single-Dimensional	Self-Attentive	Capsule-Based

image, for $i = 1, \ldots, n_f$. We define the set G as the set of regions that belong to the ground truth region, such that $R_i \in G$ if R_i is part of the ground truth region. The attention model calculates the attention weights $a_1, \ldots, a_{n_f} \in \mathbb{R}^1$ via the usual attention process. The Attention Correctness (AC) metric can then be calculated using (46).

$$AC = \sum_{i: R_i \in G} a_i . \tag{46}$$

Thus, this metric is equal to the sum of the attention weights for the ground truth regions. Since the attention weights sum up to 1 due to, for example, a softmax alignment function, the AC value will be a value between 0 and 1. If the model attends to only the ground truth regions, then AC is equal to 1, and if the attention model does not attend to any of the ground truth regions, AC will be equal to 0.

In [118], a rank correlation metric is used to compare the generated attention weights to the attention behaviour of humans. The conclusion of this work is that attention maps generated by standard attention models generally do not correspond to human attention. Attention models often focus on much larger regions or multiple small non-adjacent regions. As such, a technique to improve attention models is to allow the model to learn from human attention patterns via a joint loss of the regular loss function and an attention weight loss function based on the human gaze behaviour, similarly to how annotated attention vectors are used in [116] to supervise the attention mechanism. [117] proposes to use human attention data to supervise the attention mechanism in such a manner. Similarly, a state-of-the-art video captioning model is proposed in [119] that learns from human gaze data to improve the attention mechanism.

4.2.2 Manual Evaluation

A method that is often used to evaluate attention models is the manual inspection of attention weights. As previously mentioned, the attention weights are a direct indication of which parts of the data the attention model finds most important. Therefore, observing which parts of the inputs the model focuses on can be helpful in determining if the model is behaving correctly. This allows for some interpretation of the behaviour of models that are typically known to be black boxes. However, rather than checking if the model focuses on the most important parts of the data, some use the attention weights to determine which parts of the data are most important. This would imply that attention models provide a type of explanation, which is a subject of contention among researchers. Particularly, in [120], extensive experiments are conducted for various natural language processing tasks to investigate the relation between attention weights and important information to determine whether attention can actually provide meaningful explanations. In this paper titled “Attention is not Explanation”, it is found that attention weights do not tend to correlate with important features. Additionally, the authors are able to replace the produced attention weights with completely different values while keeping the model output the same. These so-called “adversarial” attention distributions show that an attention model may focus on completely different information and still come to the same conclusions, which makes interpretation difficult. Yet, in another paper titled “Attention is not not Explanation” [121], the claim that attention is not explanation is questioned by challenging the assumptions of the previous work. It is found that the adversarial attention distributions do not perform as reliably well as the learned attention weights, indicating that it was not proved that attention is not viable for explanation.

In general, the conclusion regarding the interpretability of attention models is that researchers must be extremely careful when drawing conclusions based on attention patterns. For example, problems with an attention model can be diagnosed via the attention weights if the model is found to focus on the incorrect parts of the data, if such information is available. Yet, conversely, attention weights may only be used to obtain plausible explanations for why certain parts of the data are focused on, rather than concluding that those parts are significant to the problem [121]. However, one should still be cautious as the viability of such approaches can depend on the model architecture [122].

5 Conclusion

In this survey, we have provided an overview of recent research on attention models in deep learning. Attention mechanisms have been a prominent development for deep learning models as they have shown to improve model performance significantly, producing state-of-the-art results for various tasks in several fields of research. We have presented a comprehensive taxonomy that can be used to categorize and explain the diverse number of attention mechanisms proposed in the literature. The organization of the taxonomy was motivated based on the structure of a task model that consists of a feature model, an attention model, a query model, and an output model. Furthermore, the attention mechanisms have been discussed using a framework based on queries, keys, and values. Last, we have shown how one can use extrinsic and intrinsic measures to evaluate the performance of attention models, and how one can use the taxonomy to analyze the structure of attention models.

The attention mechanism is typically relatively simple to understand and implement and can lead to significant improvements in performance. As such, it is no surprise that this is a highly active field of research with new attention mechanisms and models being developed constantly. Not only are new mechanisms consistently being developed, but there is also still ample opportunity for the exploration of existing mechanisms for new tasks. For example, multi-dimensional attention [64] is a technique that shows promising results and is general enough to be implemented in almost any attention model. However, it has not seen much application in current works. Similarly, multi-head attention [13] is a technique that can be efficiently parallelized and implemented in practically any attention model. Yet, it is mostly seen only in Transformer-based architectures. Lastly, similarly to how [43] combines rotatory attention with multi-hop attention, combining multi-dimensional attention, multi-head attention, capsule-based attention, or any of the other mechanisms presented in this survey may produce new state-of-the-art results for the various fields of research mentioned in this survey.

This survey has mainly focused on attention mechanisms for supervised models, since these comprise the largest proportion of the attention models in the literature. In comparison to the total amount of research that has been done on attention models, research on attention models for semi-supervised learning [123], [124] or unsupervised learning [125], [126] has received limited attention and has only become active recently. Attention may play a more significant role for such tasks in the future as obtaining large amounts of labeled data is a difficult task. Yet, as larger and more detailed data sets become available, the research on attention models can advance even further. For example, we mentioned the fact that attention weights can be trained directly based on hand-annotated data [116] or actual human attention behaviour [117], [119]. As new data sets are released, future research may focus on developing attention models that can incorporate those types of data.

While attention is intuitively easy to understand, there still is a substantial lack of theoretical support for attention. As such, we expect more theoretical studies to additionally contribute to the understanding of the attention mechanisms in complex deep learning systems. Nevertheless, the practical advantages of attention models are clear. Since attention models provide significant performance improvements in a variety of fields, and as there are ample opportunities for more advancements, we foresee that these models will still receive significant attention in the time to come.

References

[1] H. Larochelle and G. E. Hinton, “Learning to combine foveal glimpses with a third-order Boltzmann machine,” in *24th Annual Conference in Neural Information Processing Systems (NIPS 2010)* Curran Associates, Inc., 2010, pp. 1243–1251.

[2] V. Mnih, N. Heess, A. Graves, and k. kavukcuoglu, “Recurrent models of visual attention,” in *27th Annual Conference on Neural Information Processing Systems (NIPS 2014)*Curran Associates, Inc., 2014, pp. 2204–2212.

[3] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representation (ICLR 2015)*, 2015.

[4] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*. ACL, 2015, pp. 1412–1421.

[5] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*. ACL, 2016, pp. 1480–1489.

[6] Y. Wang, M. Huang, X. Zhu, and L. Zhao, “Attention-based LSTM for aspect-level sentiment classification,” in *2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)* ACL, 2016, pp. 606–615.

[7] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, “Bottom-up and top-down attention for image captioning and visual question answering,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, 2018, pp. 6077–6086.

[8] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *32nd International Conference on Machine Learning (ICML 2015)*, vol. 37. PMLR, 2015, pp. 2048–2057.

[9] Y. Ma, H. Peng, and E. Cambria, “Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM,” in *32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press, 2018, pp. 5876–5883.

[10] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *28th Annual Conference on Neural Information Processing Systems (NIPS 2015)* Curran Associates, Inc., 2015, pp. 577–585.

[11] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*IEEE Signal Processing Society, 2016, pp. 4945–4949.

[12] S. Kim, T. Hori, and S. Watanabe, “Joint CTC-attention based end-to-end speech recognition using multi-task learning,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)* IEEE Signal Processing Society, 2017, pp. 4835–4839.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *31st Annual Conference on Neural Information Processing Systems (NIPS 2017)* Curran Associates, Inc., 2017, pp. 5998–6008.

[14] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder–decoder approaches,” in *8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST 2014)*. ACL, 2014, pp. 103–111.

[15] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image Transformer," in *35th International Conference on Machine Learning (ICML 2018)*, vol. 80. PMLR, 2018, pp. 4055–4064.

[16] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*. IEEE Computer Society, 2018, pp. 8739–8748.

[17] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *28th ACM International Conference on Information and Knowledge Management (CIKM 2019)*. ACM, 2019, p. 1441–1450.

[18] F. Wang and D. M. J. Tax, "Survey on the attention based RNN model and its applications in computer vision," *arXiv:1601.06823*, 2016.

[19] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh, "Attention models in graphs: A survey," *ACM Transitions on Knowledge Discovery from Data*, vol. 13, pp. 62:1–62:25, 2019.

[20] S. Chaudhari, G. Polatkan, R. Ramanath, and V. Mithal, "An attentive survey of attention models," *arXiv:1904.02874*, 2019.

[21] D. Hu, "An introductory survey on attention mechanisms in NLP problems," *arXiv:1811.05544*, 2018.

[22] A. Galassi, M. Lippi, and P. Torroni, "Attention, please! a critical review of neural attention models in natural language processing," *arXiv:1902.02181*, 2019.

[23] M. Daniluk, T. Rocktäschel, J. Welbl, and S. Riedel, "Frustratingly short attention spans in neural language modeling," in *5th International Conference on Learning Representations (ICLR 2017)*, 2017.

[24] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging," *arXiv:1703.06052*, 2017.

[25] C. Yu, K. S. Barsim, Q. Kong, and B. Yang, "Multi-level attention model for weakly supervised audio classification," *arXiv:1803.02353*, 2018.

[26] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action recognition using visual attention," *arXiv:1511.04119*, 2015.

[27] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based LSTM and semantic consistency," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2045–2055, 2017.

[28] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention networks," in *27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*. IJCAI, 2018, pp. 3926–3932.

[29] H. Song, D. Rajan, J. Thiagarajan, and A. Spanias, "Attend and diagnose: Clinical time series analysis using attention models," in *32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press, 2018, pp. 4091–4098.

[30] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1407–1418, 2019.

[31] P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

[32] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*. Curran Associates, Inc., 2016, pp. 289–297.

[33] F. Fan, Y. Feng, and D. Zhao, "Multi-grained attention network for aspect-level sentiment classification," in *2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*. ACL, 2018, pp. 3433–3442.

[34] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification," in *26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*. IJCAI, 2017, pp. 4068–4074.

[35] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," in *4th International Conference on Learning Representations (ICLR 2016)*, 2016.

[36] S. Zheng and R. Xia, "Left-center-right separated neural network for aspect-based sentiment analysis with rotatory attention," *arXiv:1802.00892*, 2018.

[37] B. Jing, P. Xie, and E. Xing, "On the automatic generation of medical imaging reports," in *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*. ACL, 2018, pp. 2577–2586.

[38] J. Gao, X. Wang, Y. Wang, Z. Yang, J. Gao, J. Wang, W. Tang, and X. Xie, "CAMP: Co-attention memory networks for diagnosis prediction in healthcare," in *2019 IEEE International Conference on Data Mining (ICDM 2019)*. IEEE, 2019, pp. 1036–1041.

[39] Y. Tay, A. T. Luu, and S. C. Hui, "Multi-pointer co-attention networks for recommendation," in *24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD 2018)*. ACM, 2018, pp. 2309–2318.

[40] S. Liu, Z. Chen, H. Liu, and X. Hu, "User-video co-attention network for personalized micro-video recommendation," in *2019 World Wide Web Conference (WWW 2019)*. ACM, 2019, pp. 3020–3026.

[41] M. Tu, G. Wang, J. Huang, Y. Tang, X. He, and B. Zhou, "Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs," in *57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*. Association for Computational Linguistics, 2019, pp. 2704–2713.

[42] Y.-J. Lu and C.-T. Li, "GCAN: Graph-aware co-attention networks for explainable fake news detection on social media," in *58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*. ACL, 2020, pp. 505–514.

[43] O. Wallaart and F. Frasincar, "A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models," in *16th Extended Semantic Web Conference (ESWC 2019)*, ser. LNCS, vol. 11503. Springer, 2019, pp. 363–378.

[44] S. Zhao and Z. Zhang, "Attention-via-attention neural machine translation," in *32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press, 2018, pp. 563–570.

[45] L. Wu, L. Chen, R. Hong, Y. Fu, X. Xie, and M. Wang, "A hierarchical attention model for social contextual image recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[46] Y. Wang, S. Wang, J. Tang, N. O'Hare, Y. Chang, and B. Li, "Hierarchical attention network for action recognition in videos," *arXiv:1607.06416*, 2016.

[47] Z. Li, Y. Wei, Y. Zhang, and Q. Yang, "Hierarchical attention transfer network for cross-domain sentiment classification," in *32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press, 2018, pp. 5852–5859.

[48] C. Xing, Y. Wu, W. Wu, Y. Huang, and M. Zhou, "Hierarchical recurrent attention network for response generation," in *32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press, 2018, pp. 5610–5617.

[49] V. A. Sindagi and V. M. Patel, "HA-CCN: Hierarchical attention-based crowd counting network," *IEEE Transactions on Image Processing*, vol. 29, pp. 323–335, 2019.

[50] D. Kiela, C. Wang, and K. Cho, "Dynamic meta-embeddings for improved sentence representations," in *2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*. ACL, 2018, pp. 1466–1477.

[51] S. Maharjan, M. Montes, F. A. González, and T. Solorio, "A genre-aware attention model to improve the likability prediction of books," in *2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*. ACL, 2018, pp. 3381–3391.

[52] G. I. Winata, Z. Lin, and P. Fung, "Learning multilingual meta-embeddings for code-switching named entity recognition," in *4th Workshop on Representation Learning for NLP (RepL4NLP 2019)*. ACL, 2019, pp. 181–186.

[53] R. Jin, L. Lu, J. Lee, and A. Usman, "Multi-representational convolutional neural networks for text classification," *Computational Intelligence*, vol. 35, no. 3, pp. 599–609, 2019.

[54] A. Sordoni, P. Bachman, A. Trischler, and Y. Bengio, "Iterative alternating neural attention for machine reading," *arXiv:1606.02245*, 2016.

[55] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing machines," *arXiv:1410.5401*, 2014.

[56] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," in *2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*. ACL, 2017, pp. 1442–1451.

[57] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[58] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang, "Reinforced self-attention network: a hybrid of hard and soft attention

for sequence modeling," in *27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*. IJCAI, 2018, pp. 4345–4352.

[59] M. Malinowski, C. Doersch, A. Santoro, and P. Battaglia, "Learning visual question answering by bootstrapping hard attention," in *2018 European Conference on Computer Vision (ECCV 2018)*, 2018.

[60] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, vol. 34, no. 05. AAAI Press, 2020, pp. 7211–7218.

[61] S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable convolutional neural networks with dual local and global attention for review rating prediction," in *11th ACM Conference on Recommender Systems (RecSys 2017)*. ACM, 2017, pp. 297–305.

[62] J. Wang, C. Sun, S. Li, X. Liu, L. Si, M. Zhang, and G. Zhou, "Aspect sentiment classification towards question-answering with reinforced bidirectional attention network," in *57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*. ACL, 2019, pp. 3548–3557.

[63] M. Jiang, C. Li, J. Kong, Z. Teng, and D. Zhuang, "Cross-level reinforced attention network for person re-identification," *Journal of Visual Communication and Image Representation*, vol. 69, p. 102775, 2020.

[64] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "DiSAN: Directional self-attention network for RNN/CNN-free language understanding," in *32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press, 2018, pp. 5446–5455.

[65] O. Arshad, I. Gallo, S. Nawaz, and A. Calefati, "Aiding intra-text representations with visual context for multimodal named entity recognition," in *2019 International Conference on Document Analysis and Recognition (ICDAR 2019)*. IEEE, 2019, pp. 337–342.

[66] W. Wu, X. Sun, and H. Wang, "Question condensing networks for answer selection in community question answering," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*. ACL, 2018, pp. 1746–1755.

[67] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention U-Net: Learning where to look for the pancreas," in *1st Medical Imaging with Deep Learning Conference (MIDL 2018)*, 2018.

[68] R. Tan, J. Sun, B. Su, and G. Liu, "Extending the transformer with context and multi-dimensional mechanism for dialogue response generation," in *8th International Conference on Natural Language Processing and Chinese Computing (NLPCC 2019)*, ser. LNCS, J. Tang, M.-Y. Kan, D. Zhao, S. Li, and H. Zan, Eds., vol. 11839. Springer, 2019, pp. 189–199.

[69] L. Chen, B. Lv, C. Wang, S. Zhu, B. Tan, and K. Yu, "Schema-guided multi-domain dialogue state tracking with graph attention neural networks," in *34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, vol. 34, no. 05. AAAI Press, 2020, pp. 7521–7528.

[70] H. Wang, G. Liu, A. Liu, Z. Li, and K. Zheng, "Dmran:a hierarchical fine-grained attention-based network for recommendation," in *28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*.

[71] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *5th International Conference on Learning Representations (ICLR 2017)*, 2017.

[72] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv:1607.06450*, 2016.

[73] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 2020, pp. 10 076–10 085.

[74] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *36th International Conference on Machine Learning (ICML 2019)*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 7354–7363.

[75] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *27th Annual Conference on Neural Information Processing Systems (NIPS 2014)*. Curran Associates, Inc., 2014, pp. 2672–2680.

[76] A. Sinha and J. Dolz, "Multi-scale self-guided attention for medical image segmentation," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 1, pp. 121–130, 2021.

[77] J. Fajtl, H. S. Sokeh, V. Argyriou, D. Monekosso, and P. Remagnino, "Summarizing videos with attention," in *2018 Asian Conference on Computer Vision (ACCV 2018)*, ser. LNCS, vol. 11367. Springer, 2018, pp. 39–54.

[78] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*. IEEE, 2019, pp. 7115–7119.

[79] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Deep audio-visual speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[80] S. Zhang, Y. Tay, L. Yao, and A. Sun, "Next item recommendation with self-attention," *arXiv preprint arXiv:1808.06414*, 2018.

[81] G. Letarte, F. Paradis, P. Giguère, and F. Laviolette, "Importance of self-attention for sentiment analysis," in *2018 Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (BlackboxNLP 2018)*. ACL, 2018, pp. 267–275.

[82] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *13th International Conference on Web Search and Data Mining (WSDM 2020)*, 2020, pp. 519–527.

[83] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *5th International Conference on Learning Representations (ICLR 2017)*, 2017.

[84] S. Iida, R. Kimura, H. Cui, P.-H. Hung, T. Utsuro, and M. Nagata, "Attention over heads: A multi-hop attention for neural machine translation," in *57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop (ACL-SRW 2019)*. ACL, 2019, pp. 217–222.

[85] N. K. Tran and C. Niedereée, "Multihop attention networks for question answer matching," in *41st ACM SIGIR International Conference on Research & Development in Information Retrieval (SIGIR 2018)*. ACM, 2018, pp. 325–334.

[86] Y. Gong and S. R. Bowman, "Ruminating reader: Reasoning with gated multi-hop attention," in *5th International Conference on Learning Representation (ICLR 2017)*, 2017.

[87] S. Yoon, S. Byun, S. Dey, and K. Jung, "Speech emotion recognition using multi-hop attention mechanism," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*. IEEE, 2019, pp. 2822–2826.

[88] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering," in *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 2016, pp. 21–29.

[89] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, "Sentiment analysis by capsules," in *2018 World Wide Web Conference (WWW 2018)*. ACM, 2018, p. 1165–1174.

[90] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *arXiv preprint arXiv:1710.09829*, 2017.

[91] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," *arXiv preprint arXiv:1906.09890*, 2019.

[92] C. Wu, F. Wu, S. Ge, T. Qi, Y. Huang, and X. Xie, "Neural news recommendation with multi-head self-attention," in *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*. ACL, 2019, pp. 6389–6394.

[93] Y. Wang, W. Chen, D. Pi, and L. Yue, "Adversarially regularized medication recommendation model with multi-hop memory network," *Knowledge and Information Systems*, vol. 63, no. 1, pp. 125–142, 2021.

[94] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara, "Meshed-memory transformer for image captioning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 2020, pp. 10 578–10 587.

[95] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "TransUnet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.

[96] P. Zhong, D. Wang, and C. Miao, "Knowledge-enriched transformer for emotion detection in textual conversations," in *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*. ACL, 2019, pp. 165–176.

[97] Y. Zhou, R. Ji, J. Su, X. Sun, and W. Chen, "Dynamic capsule attention for visual question answering," in *33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, vol. 33, no. 01. AAAI Press, 2019, pp. 9324–9331.