Full-context semantic representation in sentence level*

Elsevier[1]

Radarweg 29, Amsterdam

Elsevier Inc[a,b], Global Customer Service[b,]*

[a] *1600 John F Kennedy Boulevard, Philadelphia*
[b] *360 Park Avenue South, New York*

Abstract

Deep models can take more context information in to account compared with shallow models. However, unrestricted stack layers will significantly affect computational performance. In this paper, we propose a single-layer architecture, comprising merely an input part and two sub-cells, to learn full-context semantic representation. Firstly, the input part compresses the sentence into three vectors representing the target word and its left and right contexts. Then, the context integration sub-cell establishes relations of the target word with its contexts through a gate-like structure. Last, semantic analysis sub-cell is further equipped to extract semantic features from these relations. With each of all words in the sentence as target word, the final representation is learned by weighting all the respective representations. Different with Transformers and multi-layer RNNs, our model directly extracts full-context information rather than gradually achieves the optimal solution. Consequently, our model can be viewed as a parallel version of RNN or Group Attention. Experimental results demonstrate that our model achieves or exceeds state of the art models while with overwhelmingly economical computation cost. Our code is available at `https://anonymous.com`.

*This is a footnote associated with the title.

*This is to indicate the corresponding author.
Email address: `support@elsevier.com` (Global Customer Service)
URL: `www.elsevier.com` (Elsevier Inc)

[1] Since 1880.

Keywords: Shallow Models, Text Representation, Full Context, Group Attention

1. Introduction

It could be derived from the history of deep learning that context information plays a crucial role in Natural Language Processing (NLP) tasks. Researchers have been working on designing context aware models for a long time and proposed a variety of outperforming models. Convolution Networks (CNNs) [1] and Recurrent Neural Networks (RNNs) [2] were designed respectively to extract local and long-range features. While Transformer based models [3, 4] were proposed to build direct connections between each word pairs to discover latent semantic relations. To make full use of context information, many variants of these models were proposed. For example, Bi-RNNs (GRUs/LSTMs) adopted an additional reverse order sequence to take bi-direction context into account [5]. Multi-Head attention is equipped to explorer more word combinations [3].

Despite the success of context-aware models, there still remains trade off between context scope and model depth. Generally, stacking more layers can expand the model's perception scope and make the model perform better. For example, the perception field of a CNN layer (kernel size = 3) is 3, and stacking two such layers can expand the perception field to 5, which means the context scope is proportional with the model depth. However, there is a natural situation that unnecessary information may disturb the model if the context scope exceeds the sequence length. This may partly explain why the optimal number of layers and kernel size has to be tuned and set up specially for different datasets. Besides, unconditionally stacking layers may become infeasible for the space and time complexities grow sharply with the increasing layers [6], especially in some computational resource constrained devices.

Considering the above concerns, researchers have proposed shallow and wide architectures. In the recent work [7], the authors proposed a 2-layer Shallow RNN, which can alleviate the computational bottleneck inherited in recurrent

models. The authors of [8] proposed to extract semantic information through stacking feature maps instead of layers. In comparison, shallow models have simpler architectures, which means lower memory cost and easier deployment. However, shallow models might ignore long-range relations or truncate the propagation of semantic information, and result in a sheer performance loss. As discussed above, shallow models are weak in incorporating context information while deep models need more computational resources, which sometimes is intolerable in practice.

In this paper, we take a bolder step towards the trade off between context scope and model depth, by proposing a computational module which contains only one layer while making full use of context knowledge for each word. We hence name our model Full-Context Semantic Representation (FCSR). Firstly, all left and right contexts of the target word are compressed into two fixed-length context vectors respectively, in which the words closer to the target word are endowed larger weights and vice versa. Then, the relation features are generated by context integration sub-cell. During the computation, gate structures are used to select informative words. Different with Transformers, we use word-context attentions rather than word-word attentions. And last, the semantic analysis sub-cell is introduced to extract semantic features from relation features. Consequently, FCSR can be interpreted as a group attention mechanism.

Transformer and FCSR have the same propose of studying relations among words. To be concrete, Transformer tries to connect all word pairs while FCSR selects informative features through group attention mechanism. Compared with RNNS, FCSR has no recurrent step and operates mainly based on element-wise operations and matrix multiplications, which means low computational cost. The computational complexity of our model is the same as matrix multiplications, which makes its deployment efficient.

FCSR maintains the full advantages of dot product attention [6, 9, 4], preserving full scale context information through constructing direct interaction between the target word and its contexts. Meanwhile, the time complexity is

drastically reduced to $O(m)$, where m is the sequence length. Furthermore, it could benefit from totally parallel computing since FCSR has little recurrent step, even layer loops.

We perform experiments with FCSR on several text classification benchmarks, and show that FCSR reaches or exceeds state of the art results on some of them. Meanwhile, our model needs much less training time comparing with competitors. To summarize, our contributions are as follows:

1 We propose FCSR, a new context aware of attention based models that achieves $O(m)$ time complexity.

2 We show strong performance of FCSR on various benchmarks, achieving or exceeding state-of-the-art results on some datasets and outperforming competitor models in time consumption.

2. Related work

Short text is prevalent in social media such as Twitter, Microblog, commodity reviews and so on. Since users seldom follow grammar laws strictly and are prone to use slang words, linguistic based algorithms may not function well on these datasets. In this paper, the conception of a sentence is extended to a contiguous text or word sequence rather than a linguistically plausible sentence [10].

The major purpose of text representation algorithms is to extract meaningful features from text data, and remove or suppress the semantic distortions. Deep models is a kind of representation algorithm, which extract hierarchical features from layer to layer, to reasoning the abstraction knowledge, and thus to improve the performance of downstream tasks such as Sentiment Analysis(SA) [11], Topic Tracking(TT) [12] and Natural Language Inference(NLI)[13], etc. multi-layer structures may suffer from gradient vanishing and explosion problems [14, 15]. Simply splitting the sentence into a group of segments [4] may truncate the propagation of semantic information. Equipping a fixed-size context window to LSTM seems a reasonable approximation [16].

The reason why deep learning based models can generate more representative vectors [17, 18] is partly because of the usage of word2vec algorithm [19]. Followed the idea of word2vec, researchers have developed different text representation algorithms based on context dependence and word co-occurrence [20].

CNNs were used to extract local features [21]. In [8], the author proposed a shallow CNN model. In [22], the authors proposed a joint structure of CNN and RNN. And in [23], the authors introduced a neural network architecture composed of bidirectional LSTM, CNN and CRF.

One limitation of CNN is that long-range dependencies are neglected. In order to deal with this problem, RNNs and LSTMs were introduced since they are able to preserve sequence information. In [24], the authors proposed a hierarchical model corresponding to the hierarchical structure of documents. In [25], the authors proposed a simplified gating mechanism and built a hierarchical representation to capture long-range dependencies. These algorithms are designed to analysis document level data.

In the sentence level, the authors [16] proposed Sentence-state LSTM in which a dummy node was utilized to represent the sentence level state and all words in the context window can interact with this node. In [26], the authors proposed a multi-task learning algorithm to consistently improve the performance of CNN and LSTMs.

Transformers [3] are attention based models trying to study all word-pairs in the sentence. In real applications, researchers have found that the space complexity of transformers grows quadratically with the sequence length and the number of heads. To address this limitation, the authors [4] split the sentence into a group of segments and processed each segment separately. And in [6], the authors introduced a memory cell to store the information in previous segments to hold a global view.

3. Model

Given a sentence $\boldsymbol{S} \in \mathbb{R}^{m \times n}$, in which each $\boldsymbol{S}_t|_{t=1}^{m} \in \mathbb{R}^n$ denotes the word embedding of the t th word, m is the length of $\boldsymbol{S}$, n is the dimension of word embeddings. Our goal is to generate a representation vector $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ for $\boldsymbol{S}$, shown in Equation 1.

$$\boldsymbol{R} = \{FCSR(\boldsymbol{S}_1|\boldsymbol{S}), FCSR(\boldsymbol{S}_2|\boldsymbol{S}), ..., FCSR(\boldsymbol{S}_m|\boldsymbol{S})\} \tag{1}$$

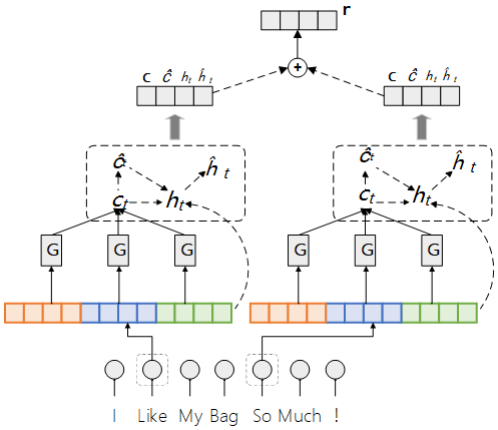

Figure 1: The process of generating a representation vector $\boldsymbol{r}$.

For classification tasks, a feature attention layer is adopted to compress $\boldsymbol{R}$ into a representation vector $\boldsymbol{r} \in \mathbb{R}^n$. The process of generating $\boldsymbol{r}$ is shown in Figure 1.

3.1. Structure of FCSR

With $\boldsymbol{S}_t$ as the target word, its representation vector $\boldsymbol{R}_t$ is generated through FCSR. Figure 2 illustrates the structure of FCSR.

First, we compress the left and right context words, $\boldsymbol{S}_1 \sim \boldsymbol{S}_{t-1}$ and $\boldsymbol{S}_{t+1} \sim \boldsymbol{S}_m$, of $\boldsymbol{S}_t$ into a fixed length vector respectively. Next, the context integration sub-cell is trained to generate the relation features, $\boldsymbol{C}_t$ and $\hat{\boldsymbol{C}}_t$. $\boldsymbol{C}_t$ is the weighted attention of $\boldsymbol{S}_t$ over its contexts, $\hat{\boldsymbol{C}}_t$ is $\boldsymbol{C}_t$'s non-linearity. Then, the semantic analysis sub-cell is further designed to impose these relations on the

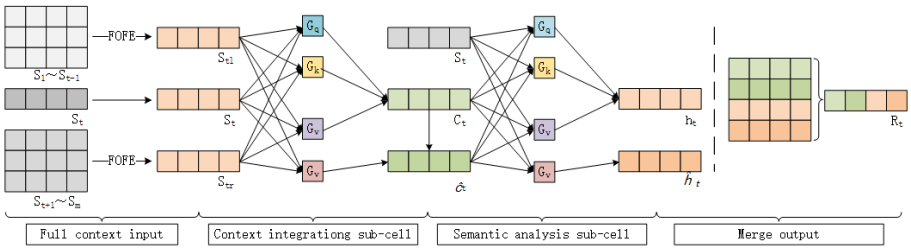

Figure 2: The process of generating a representation vector for $\boldsymbol{W}$.

target word to reason semantic features, $\boldsymbol{H}_t$ and $\hat{\boldsymbol{H}}_t$. With each target word cooperated with it's full-scale context in such a way, our model is set up to extract semantic features in a global aspect without multi-layer structure. At last, we compress all representation vectors into a fixed length vector through a feature attention layer.

3.2. Full scale context

We use a FOFE [27] based context collecting method to compress arbitrary number of context words into a fixed size vector. The FOFE algorithm is as Equation 2.

$$FOFE(\boldsymbol{S}_i, \boldsymbol{S}_j) = \begin{cases} \alpha FOFE(\boldsymbol{S}_i, \boldsymbol{S}_{j-1}) + \boldsymbol{S}_j & j > i, \\ \boldsymbol{S}_j & j = i, \end{cases} \tag{2}$$

where $FOFE(\boldsymbol{S}_i, \boldsymbol{S}_j)$ is the compressed vector of the sentence $\boldsymbol{S}$ from $\boldsymbol{S}_i$ to $\boldsymbol{S}_j$, α is the weight to control the influence of history vectors.

For each $\boldsymbol{S}_t$, as shown in Equation 3, we collect its full-scale contexts by constructing left context $\boldsymbol{S}_{t_l}$ and right context $\boldsymbol{S}_{t_r}$. Particularly, for the boundary case, $\boldsymbol{S}_{1_l}$ and $\boldsymbol{S}_{m_r}$ are set with zero.

$$\begin{aligned} \boldsymbol{S}_{t_l} &= FOFE(\boldsymbol{S}_1, \boldsymbol{S}_{t-1}), \\ \boldsymbol{S}_{t_r} &= FOFE(\boldsymbol{S}_m, \boldsymbol{S}_{t+1}). \end{aligned} \tag{3}$$

3.3. Sub-Cells

We propose two structure similar sub-cells, Context Integration Sub-cell (CIS) and Semantic Analysis Sub-cell (SAS). each of which contains an additive

attention $f_a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x}; \boldsymbol{G}_q, \boldsymbol{G}_k, \boldsymbol{G}_v)$ and a non-linearity transformation $f_t(\boldsymbol{x}; \boldsymbol{G}_o)$. f_a contains three gates $\boldsymbol{G}_q$, $\boldsymbol{G}_k$ and $\boldsymbol{G}_v$, refer to $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ in Transformer respectively. Each gate is learned to decide what information should be integrated into representation vectors, and f_t performs non linear transformations. f_a and f_t is as Equation 4.

$$\begin{aligned} f_a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}; \boldsymbol{G}_q, \boldsymbol{G}_k, \boldsymbol{G}_v) &= \boldsymbol{x} * \boldsymbol{G}_q + \boldsymbol{y} * \boldsymbol{G}_k + \boldsymbol{z} * \boldsymbol{G}_v, \\ f_t(\boldsymbol{x}; \boldsymbol{G}_o) &= tanh(\boldsymbol{x}) * \boldsymbol{G}_o. \end{aligned} \tag{4}$$

Where $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ are input vectors, $\boldsymbol{G}_*$ are gates defined by each sub-cell respectively, $*$ denotes point wise multiplications (Hadamard Products).

3.3.1. Context Integration Sub-Cell

Context Integration Sub-cell defines $\boldsymbol{G}_q$, $\boldsymbol{G}_k$ and $\boldsymbol{G}_v$ by compressing $\boldsymbol{S}_t$, $\boldsymbol{S}_{t_l}$ and $\boldsymbol{S}_{t_r}$ into one vector respectively. Then, activate the compressed vectors by ReLU function which is used to deactivate non-important information and weight useful features. Meanwhile, the output gate $\boldsymbol{G}_o$ has similar form. The definitions of gates are shown in Equation 5.

$$\begin{aligned} \boldsymbol{G}_{t_q} &= \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_q} + \boldsymbol{S}_{t_c} \times \boldsymbol{W}_{c_q} + \boldsymbol{b}_q), \\ \boldsymbol{G}_{t_k} &= \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_k} + \boldsymbol{S}_{t_c} \times \boldsymbol{W}_{c_k} + \boldsymbol{b}_k), \\ \boldsymbol{G}_{t_v} &= \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_v} + \boldsymbol{S}_{t_c} \times \boldsymbol{W}_{c_v} + \boldsymbol{b}_v), \\ \boldsymbol{G}_{t_o} &= \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_o} + \boldsymbol{S}_{t_c} \times \boldsymbol{W}_{c_o} + \boldsymbol{b}_o). \end{aligned} \tag{5}$$

Where $\boldsymbol{G}_{t_*} \in \mathbb{R}^n$ denotes the gate of the t th word, $\boldsymbol{S}_{t_c} = concat(\boldsymbol{S}_{t_l}, \boldsymbol{S}_{t_r})$ denotes the context of $\boldsymbol{S}_t$, $\boldsymbol{W}_{t_*} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{W}_{c_*} \in \mathbb{R}^{2n \times n}$ are trainable parameters, $\boldsymbol{b_*} \in \mathbb{R}^n$ are bias, $tanh$ denotes tanh function, $\times$ denotes matrix multiplications.

According to Equation 4 and 5, the Context Integration Sub-cell is defined as Equation 6.

$$\begin{aligned} \boldsymbol{C}_t &= f_a(\boldsymbol{S}_t, \boldsymbol{S}_{t_l}, \boldsymbol{S}_{t_r}; \boldsymbol{G}_{t_q}, \boldsymbol{G}_{t_k}, \boldsymbol{G}_{t_v}), \\ \hat{\boldsymbol{C}}_t &= f_t(\boldsymbol{C}_t; \boldsymbol{G}_{t_o}), \end{aligned} \tag{6}$$

As shown in Equation 4, 5 and 6, $\boldsymbol{C}_t$ is composite of three terms, the multiplications of word vectors with its corresponding gates. These gates are used

to control how much information should flow in $\boldsymbol{c}_t$ from vectors. Precisely, we implement the global scale attention by using item-wise multiplications instead of dot productions. Furthermore it is helpful to learn abstract features by applying a non linear activation function to $\boldsymbol{c}_t$.Since $\boldsymbol{S}_{t_l}$ and $\boldsymbol{S}_{t_r}$ contain all left and right context knowledge, the relation vectors $\boldsymbol{C}_t$ and $\hat{\boldsymbol{C}}_t$ can be viewed as the combination of all words in $\boldsymbol{S}$ with $\boldsymbol{S}_t$ as the central word.

3.4. Semantic Analysis Sub-Cell

Following the same paradigm, Semantic Analysis Sub-cell also defines four gates. Comparing to CIS, linear to non-linear mode, SAS adopts a non-linear to non-linear mode that take non-linear features as input to generate representation vectors. Precisely. SAS emphasizes the impacts of full-scale relations on the target word. The definitions of gates are as Equation 7.

$$\begin{aligned}
\boldsymbol{G}_{t_q} =& \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_q} + \boldsymbol{C}_t \times \boldsymbol{W}_{l_q} + \hat{\boldsymbol{C}}_t \times \boldsymbol{W}_{r_q} + \boldsymbol{b}_q), \\
\boldsymbol{G}_{t_k} =& \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_k} + \boldsymbol{C}_t \times \boldsymbol{W}_{l_k} + \hat{\boldsymbol{C}}_t \times \boldsymbol{W}_{r_k} + \boldsymbol{b}_k), \\
\boldsymbol{G}_{t_v} =& \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_v} + \boldsymbol{C}_t \times \boldsymbol{W}_{l_v} + \hat{\boldsymbol{C}}_t \times \boldsymbol{W}_{r_v} + \boldsymbol{b}_v), \\
\boldsymbol{G}_{t_o} =& \sigma(\boldsymbol{S}_t \times \boldsymbol{W}_{t_o} + \boldsymbol{C}_t \times \boldsymbol{W}_{l_o} + \hat{\boldsymbol{C}}_t \times \boldsymbol{W}_{r_o} + \boldsymbol{b}_o),
\end{aligned} \tag{7}$$

where $\boldsymbol{G}_{t_*} \in \mathbb{R}^n$ are gates used to control information flowing, $\boldsymbol{W}_* \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b}_* \in \mathbb{R}^n$ are trainable weights and bias.

Similarly, the activated semantic vector is learned, as shown in Equation 8.

$$\begin{aligned}
\boldsymbol{H}_t =& f_a(\boldsymbol{S}_t, \boldsymbol{C}_t, \hat{\boldsymbol{C}}_t; \boldsymbol{G}_{t_q}, \boldsymbol{G}_{t_k}, \boldsymbol{G}_{t_v}), \\
\hat{\boldsymbol{H}}_t =& f_t(\boldsymbol{H}_t; \boldsymbol{G}_{t_o}),
\end{aligned} \tag{8}$$

In order to synthesize the contribution of each feature, we set the representation $\boldsymbol{R}_t = concat(\boldsymbol{C}_t, \hat{\boldsymbol{C}}_t, \boldsymbol{H}_t, \hat{\boldsymbol{H}}_t)$. The sentence representation $\boldsymbol{R}$ is thus represented as $\boldsymbol{R} = \{\boldsymbol{R}_t|_{t=1}^m\}$.

3.5. Feature Attention

It is necessary to transform the representation vectors $\boldsymbol{R}$ to a fixed size sentence level representation $\boldsymbol{r}$ to implement classification tasks. Meanwhile,

the importance of different feature to the representative performance may not be same. Consequently, it is worth selecting out which features should be granted large weight. We apply a feature attention module to get those important features, and the definitions are as Equation 9.

$$\begin{aligned} \hat{\boldsymbol{R}} &= \sigma\left[\frac{(\boldsymbol{R}\times\boldsymbol{W}_q)\times(\boldsymbol{R}\times\boldsymbol{W}_k)}{\sqrt{n}}\right]\times\boldsymbol{W}_f, \\ \boldsymbol{r} &= flat(\hat{\boldsymbol{R}}). \end{aligned} \tag{9}$$

Where $\boldsymbol{W_q}, \boldsymbol{W_q} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{W_f} \in \mathbb{R}^{n\times 1}$ are parameters used to select features, $flat$ denotes to reshape the input vector into $[1 \times n]$.

4. Interpretation

It is well known that the basic idea of Transformer is to learn, from $\boldsymbol{Q}$ and $\boldsymbol{K}$ for each sentence, an attention matrix, which can instruct how to select items from $\boldsymbol{V}$ and merge them to form representation vectors. As shown in Equation 10, softmax and other functions were omitted for simplicity, the representation matrix $\boldsymbol{R}$ is derived, by multiplying attention matrix $\boldsymbol{T}$, from linear combinations of $\boldsymbol{V}$. For clarity and emphasis the parallel character of FCSR, formulas in this chapter will be present in matrix form.

$$\boldsymbol{T} = \boldsymbol{Q}\times\boldsymbol{K}^{\mathsf{T}}, \quad \boldsymbol{R} = \boldsymbol{T}\times\boldsymbol{V}, \quad \boldsymbol{Q},\boldsymbol{K},\boldsymbol{V},\boldsymbol{R} \in \mathbb{R}^{m\times n}, \quad \boldsymbol{T} \in \mathbb{R}^{m\times m}. \tag{10}$$

In practice, FCSR splits the Transformer into two parts: the context collection part correlates the first term while the two sub-cells correlate to the second term of Equation 10 while .

4.1. Full context

Transformer utilizes context information through traditional Linear Model (LM), which transforms word embeddings into another semantic space. Essentially, LM is equal to perform column elementary transformations that do not utilize context information. To reasoning semantic information, it needs row, instead of column, transformations to make use of context knowledge. Since

the length of different sentences may not be same, it is infeasible to define row transformation matrices $\boldsymbol{T}$. As an approximation, Transformer tactfully converts this row transformation problem into a column transformation problem through self attention. As shown in Equation 11, it imposes LMs on query, key and value firstly to form $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, and then merge $\boldsymbol{Q}$ and $\boldsymbol{K}$ to form $\boldsymbol{T}$. Apparently, this approximation may be disturbed by the word embeddings (columns). Consequently, multi-head attention, a natural solution, is proposed to split word embeddings into several segments and perform self attention independently. This method can undoubtedly acquire more context information, but it may results in space cost increase significantly.

$$\begin{aligned} &\boldsymbol{Q} = query \times \boldsymbol{W}_q + \boldsymbol{b}_q, \quad \boldsymbol{K} = key \times \boldsymbol{W}_k + \boldsymbol{b}_k, \\ &\boldsymbol{V} = value \times \boldsymbol{W}_v + \boldsymbol{b}_v, \quad \boldsymbol{W}_* \in \mathbb{R}^{n \times n}. \end{aligned} \tag{11}$$

FCSR achieves similar results in a more intuitive way, combining context vectors directly with the help of FOFE. As shown in Equation 12, FOFE can be viewed as left multiplying a Transformation matrix on the left of word embeddings.

$$FOFE(\boldsymbol{S}_i, \boldsymbol{S}_j) = \boldsymbol{T}_{i,j} \times [\boldsymbol{S}_i, ..., \boldsymbol{S}_j]^{\mathsf{T}}, \quad \boldsymbol{T}_{i,j} = [\alpha^{j-i}, ..., \alpha^2, \alpha, 1]. \tag{12}$$

FCSR collects context information for the target word $\boldsymbol{S}_t$ by merging contexts $\{\boldsymbol{T}_{1,t-1}|_{t=1}^{m}\}$ and $\{\boldsymbol{T}_{t+1,m}|_{t=1}^{m}\}$ into $\boldsymbol{T}_l$ and $\boldsymbol{T}_r$ firstly. Then, performing row transformations, shown in Equation 13, over word embeddings through $\boldsymbol{T}_l$ and $\boldsymbol{T}_r$. Comparing to Transformer, FSCR can also takes full scale context into consideration. In fact, attention matrix $\boldsymbol{T}$ learned by Transformer has similar distribution related to $\boldsymbol{T}_l$ and $\boldsymbol{T}_r$. Furthermore, It is possible for FCSR to restore, by left multiplying the inverse matrix $\boldsymbol{T}_l^{-1}$ and $\boldsymbol{T}_r^{-1}$, the compressed context to its original form with little information lost.

$$\boldsymbol{S}_l = \boldsymbol{T}_l \times S, \quad \boldsymbol{S}_r = \boldsymbol{T}_r \times S. \tag{13}$$

Where $\boldsymbol{S}_l$ and $\boldsymbol{S}_r \in \mathbb{R}^{m \times n}$ are left and right context vectors related to word embedding $\boldsymbol{W}$. FCSR artificially splits and weights full context into $\boldsymbol{S}$, $\boldsymbol{S}_l$ and $\boldsymbol{S}_r$.

4.2. Sub-cells

The two sub-cells have similar structures except the definition of gates. Specially, CIS concatenates the compressed context matrices $\boldsymbol{S}_l$ and $\boldsymbol{S}_r$ and then transforms $\boldsymbol{S}_c$ into a lower feature space. This process is agree with Transformer in that each gate is composited with $\boldsymbol{S}$ and its full scale context (see Equation 5). Beyond Transformer, FCSR does not output gates directly, it performs extra pair wise attentions on the context through gates (see Equation 4). SAS treats $\boldsymbol{C}$ and $\hat{\boldsymbol{C}}$ independently, aims at detecting latent semantic features from compressed context and non-linear relations (see Equation 7). The comparison between Transformer and the two seb-cells is shown in Equation 11 and 14. SAS gates are omitted for simplicity.

$$\begin{aligned}
\boldsymbol{G}_q &= \sigma(\boldsymbol{S} \times \boldsymbol{W}_{qt} + \boldsymbol{S}_c \times \boldsymbol{W}_{qc} + \boldsymbol{b}_q), \\
\boldsymbol{G}_k &= \sigma(\boldsymbol{S} \times \boldsymbol{W}_{kt} + \boldsymbol{S}_c \times \boldsymbol{W}_{kc} + \boldsymbol{b}_k), \\
\boldsymbol{G}_v &= \sigma(\boldsymbol{S} \times \boldsymbol{W}_{vt} + \boldsymbol{S}_c \times \boldsymbol{W}_{vc} + \boldsymbol{b}_v), \\
\boldsymbol{C} &= \boldsymbol{G}_q * S + \boldsymbol{G}_k * \boldsymbol{S}_l + \boldsymbol{G}_v * \boldsymbol{W}_r
\end{aligned} \tag{14}$$

As shown in Equation 14, FCSR adopts one more LM than Transformer. Instead of dot production, item wise attention is performed to select and merge $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ into $\boldsymbol{C}$. Meanwhile, SAS has similar structure except it incorporates three LMs to compute $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$.

To discover potential semantic relations, we set up an additional output gate to filter information from $\boldsymbol{C}$ and implement non linearity. As shown in Equation 15, the structure of output gate is similar with input gates.

$$\begin{aligned}
\boldsymbol{G}_o &= \sigma(\boldsymbol{S} \times W_{ot} + \boldsymbol{S}_c \times W_{oc} + \boldsymbol{b}_o), \\
\boldsymbol{H} &= \boldsymbol{O} * tanh(\boldsymbol{C}).
\end{aligned} \tag{15}$$

Essentially, the function of a sub-cell is similar with a Transformer layer except in three aspects: 1) FCSR uses pre-defined context collection policy which has little information lost. 2) FCSR uses more LMs to learn attention matrices. and 3) Additional output gate endows FCSR extra non-linear semantic information.

5. Experiments

We compared our model with: 1) Shallow CNN, 2) LSTM, 3) stacked LSTM (stkLSTM), 4) Bi-LSTM, 5) stacked BiLSTM (stkBiLSTM), 6)Transformer, 7) S-LSTM and 8) multi-task models. Stacked architectures were compared to show that multi-layer structure can consistently improve the performance of original model. S-LSTM was compared to show the difference between fixed-size context window based algorithm and full-scale context model. For multi-task learning, we compared with the models of [26] to validate if our model can function well on complicated tasks. All experiments were conducted using a Nvidia GTX 1080Ti GPU with 11GB memory, Pytorch 1.6.

Table 1: Statistics of 5 datasets.

Dataset	**Class**	**Type**	**Ins.**	**L.**	**Vocab**
AGNews	4	topic	400	43	21 K
Polarity	2	news	10 L	50	21 K
Final Sentiment Data	4	sentiment	3090	27	1.1 K
SST-2	2	review	400	19	26 K
Yelp F	2	sentiment	700 K	149	62 K

5.1. Datasets and hyper parameters

To make an extensive evaluation, 6 kinds of datasets were adopted in the experiments: SST-2 [28], AGNews [29], Yelp F [30], Polarity [31], Final Sentiment Data [2] and a group of 16 widely used review datasets [26].

To make fairly comparisons, we used Global Vectors for Word Representation(GloVe) [19] as word embedding, all word or phrase in the sentence was

[2]This dataset contains cleaned tweets from India on topics like corona-virus, COVID-19 and lock-down etc. The tweets have been collected between dates 23th March 2020 and 15th July 2020. Then the text have been labeled into four sentiment categories fear, sad, anger and joy. The dataset is available at `https://www.kaggle.com/surajkum1198/twitterdata`

Table 2: Statistics of datasets [26].

Dataset	**Trn**	**Dev**	**Tst**	**Avg.L**	**Vocab**
Apparel	1400	200	400	57	21K
Baby	1400	200	400	104	26K
Books	1400	200	400	159	62K
Camera	1397	200	400	130	26K
DVD	1400	200	400	173	69K
Electronic	1398	200	400	101	30K
Health	1400	200	400	81	26K
IMDB	1400	200	400	269	44K
Kitchen	1400	200	400	89	28K
Magazine	1370	200	400	117	30K
MR	1400	200	400	21	12K
Music	1400	200	400	136	60K
Software	1315	200	400	129	26K
Sports	1400	200	400	94	30K
Toys	1400	200	400	90	28K
Video	1400	200	400	156	57K

converted to 300 dimensional embedding vector. All weights were randomly initialized by the Normal distribution. Dropout [32] rate, batch size and other hyper parameters was set according to datasets and the memory capability. All texts in same batch were padded to same length. Forgetting parameter α of FOFE was fixed to 0.2. All models were optimized using the Adam optimizer [33]. Particullarly, to select useful features, we used FP-net to extract features in an anti-graident direction. Furthermore, an epoch related learning rate updating policy, shown in Equation 16, is used with an initial learning rate of 0.01.

$$lr_{i+1} = 0.8 * lr_i * 0.01^{\frac{i+0.01}{epoch_{max}+0.01}}. \quad (16)$$

Where lr denotes the learning rate, i is the current epoch index, $epoch_{max}$ is the max epoch, 40 in our experiments.

5.2. Simple case analysis

Firstly, we show three examples, which were correctly classified by our model and wrongly by others. These examples selected from the experimental datasets have some ambiguous attitudes or adversative relations.

1) *The film feels [uncomfortably real, its language and locations bearing] the unmistakable stamp of authority. (Positive, MR)*

2) *I am solidly a size medium, but I was swimming in this jacket. I think it is more of an XL. I would say the medium if for someone 5'10" and 250lbs. (Negative, apparel)*

3) *A real clunker. A well-made, thoughtful, well-acted clunker, but a clunker nonetheless. (Negative, MR)*

The first review is to express the film gave people a real feeling. The key words in the sentence is "*feels uncomfortably real*". If a model uses the key word "language" and set the size of the context window with 3, it may incorrectly associate "*uncomfortably*" with "*bearing*" and result in negative attitude. In our model, full-scale context is utilized by each target word and hence all words are taken into account.

The second review is to express the man did not fit the jacket. The model with large but fixed-size context window may build connections among three red words and lead to misunderstanding. Our model uses all other words to decorate the target word "*swimming*", and hence better representation can be expected.

The third review is to express the man did not like old cars, but there were many positive keywords in the review. If a model is weak in filtering out noisy information, it may get completely misunderstand. With each word as subject word, our model can be expected to find out what is the most important word.

These simple examples show that our model has some advantages over existing algorithms. It can detect important association information among words in the global scope and omit unnecessary relations.

5.3. Full-scale context

In Figure 3, we show the results of CNN and FCSR with different size of context window, from 1 to full-scale, on Finalsentiemt2 dataset, average length is 27. FCSR and CNN incorporate dynamic context window by adjusting context matrix and stacking multi layers. Both model have two versions, direct and tuned. For direct versions, we set the batch size, weight decay learning rate with 20, 1e-4 and 1e-3 respectively, Meanwhile, in tuned versions, the parameters were tuned artificially to achieve the best performance.

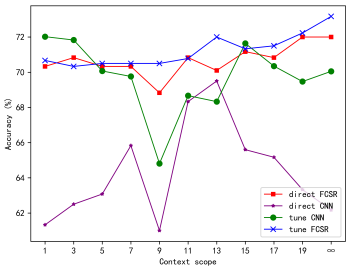

Figure 3: The accuracy of different context window of FCSR on apparel dataset.

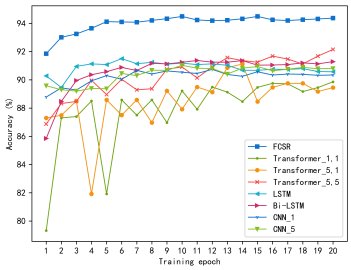

Figure 4: The accuracy of transformer with different heads and layers (batch size was set with 20).

It can be noticed that there has been a general increasing trend of accuracy with the growth of context window, which implies that representation models need more context knowledge. Particularly, all models reported a performance descent when the context window was set with 9, which shows that there may exists semantic transition. Compared with other models, direct CNN reports a sheer performance inferior, which insists that the parameters need to be tuned carefully once the structure changed. In contrast, tuned CNN has superior results, and it reports the highest performance under 15 context area, insists

that appropriate context scope will benefit the classification task. However, little accuracy gain achieved when the context window exceeds 15 since the phenomena perceptual fields overlap and disturb emerged.

For FCSR, the accuracy trend agree with CNN's, shows that context information indeed plays an important role. There is little difference between direct FCSR and tuned FCSR, shows that our model is not sensitive to parameters and can make full use of context information. The reason is FCSR do not need multi layers and the size of context window is only correlated with the context matrix, which makes the structure of FCSR fixed and has little variants. Furthermore, the training time of our model is about 2.2 seconds per epoch, independent with size of the context window. More discussions about time cost can be found in Figure 7 and later comparison.

In Figure 4, we show the accuracy trends along with training steps on the AGnews dataset. FCSR outperforms all of the competitors with a notable gap. Meanwhile, the advantage of 5-layer LSTM over 1-layer is not that big, insists that simply stacking layers may not acquire adorable performance gain. For multi-layer(head) transformer, as we dissected above, its performance is affected by dimensions, it needs several heads to make full use of context information. From the figure, multi-layer(head) indeed works, make the accuracy increases from 89% to 92% and we believe the performance can be increased insistently through more heads. However, the computational cost may grows sharply and make the model become infeasible in some resource limit environments.

To validate this situation, we process Transformer on Apparel dataset, the average length is longer than AGnews, with various layer and head. Figure 5 illustrates the number of heads and layers changing from 1 to 6 respectively. It can be noticed that the accuracy of the model increases with the increase of number of heads and layers. This is because multi-head can provide more word pairs information and multi-layer structure can learn hierarchical features. These results imply that context knowledge will benefit generating representative vectors. However, when the number of heads and layers grew large, the model failed quickly due to the limited space of our device, and hence the re-

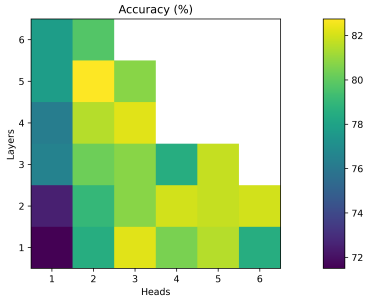

Figure 5: The accuracy of transformer with different heads and layers (batch size was set with 20).

sults were not obtained and shown in the figure. In fact, the model's memory consumption will grow very quickly for long sentences, as analyzed above. The average sentence length of apparel is 57. For a longer dataset, such as IMDB with average length of 269, the number of heads and layers, and the batch size have to be decreased to run the model, which will unexpectedly degrade the performance as well.

5.4. Result statistics

ROC curves and AUC scores, the area that covered by ROC curve and axis, of the 5 datasets are reported in Figure 6. In comparison, our model achieved the highest average macro and micro AUC score in the experiments. It firmly implies that it is feasible to learn full-context semantic representation with a simple FSCR process, for the proposed model takes full scale information into account and extracts relations in global scale. Tough our model didn't achieve best results in Yelp, it still reported smooth ROC curves, which implied that our model generates average distributions across multi classes. Furthermore, for the sake of only one-layer structure, it can significantly improve the computational efficiency and support parallel computing. Since some competitor models didn't report their AUC score, following their original research, the accuracy is also listed in Table 3.

To further examine the proposed model, we compared it with competitor

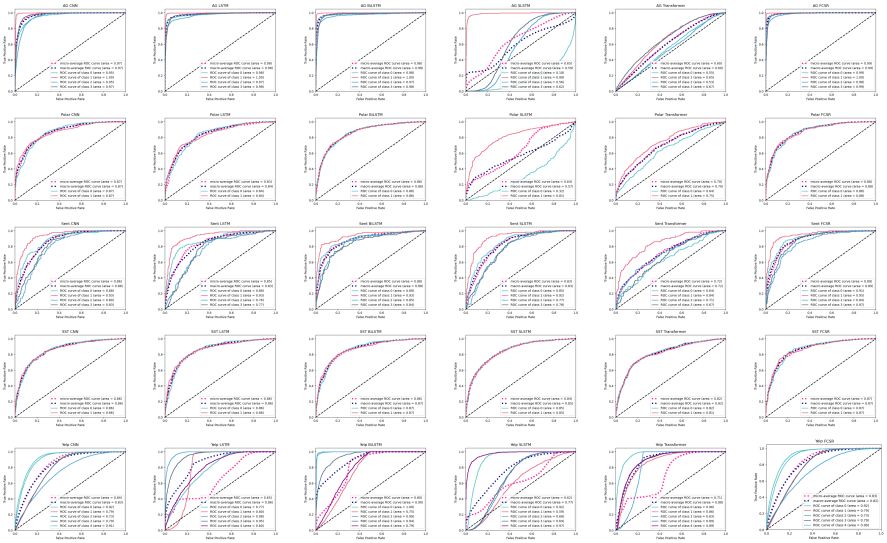

Figure 6: AUC scores

models on both single and multi dataset tasks on a group of 16 datasets. Table 4 and 5, illustrated the experimental results.

In Table 4, the results of CNN were from [8]. The results of LSTM and stack LSTM were from [26]. The results of bidirectional LSTM (BiLSTM), stack BiLSTM and S-LSTM were from [16]. The number of heads and layers of the transformer is set with 6 (batch size was set to 2 otherwise the memory will be exhausted). From Table 4 we can see that both BiLSTM and stack BiLSTM performed better than LSTM and stack LSTM, which shows that the bidirectional propagation of hidden states can effectively improve the representation ability of LSTM. Meanwhile, the accuracy of stack BiLSTM exceeds BiLSTM for most of the datasets, which indicates that additional multi-layer structure can generate deep semantic representative vectors, though they reported inferior results compared to S-LSTM and transformers. FCSR achieves the best results on 11 datasets, indicates that full context information is an important factor to extract semantic features. Compared with existing mainstream models, FCSR reports more than 0.46% average accuracy improvement.

In Table 5, SP-MTL, ASP-MTL, SP-MTL-SC and ASP-MTL-SC were pro-

Table 3: Accuracy of 5 datasets on competitor models.

Medel	AG	Yelp F.	Polar	SST	Final
CNN	91.4	Yelp F.	78.7	77.2	66.7
BiLSTM	91.5	58.2	77.3	78.7	70.8
Transformer [3]	90.8	59.3	76.1	77.8	69.1
Char-CNN [29]	87.2	62.0	79.5	78.2	66.7
Encoder-CNN-S [34]	92.5	66.2	78.6	76.4	72.2
Encoder-CNN-A [34]	93.0	66.5	78.6	77.1	70.1
Encoder-DRNN-S [34]	93.0	66.8	79.2	79.6	69.9
Encoder-DRNN-A [34]	93.2	**67.0**	78.9	78.4	71.5
S-LSTM [16]	91.4	61.8	74.4	76.2	68.0
VAMPIRE [35]	88.0	62.2	77.8	80.3	73.0
CSPAN (base) [36]	93.7	65.9	76.8	78.5	69.1
CSPAN (big) [36]	93.6	66.0	75.4	78.5	70.0
FCSR	**94.5**	63.4	**80.5**	**80.8**	**73.3**

Table 4: Accuracy of 16 datasets.

Dataset	CNN	LSTM	stkLSTM	Bi-LSTM	stkBiLSTM	Transformer	SLSTM	FCSR
Apparel	85.25	83.20	83.70	86.05	86.35	86.33	85.75	**88.52**
Baby	84.25	84.70	84.20	84.51	85.45	84.25	**86.25**	84.89
Books	81.75	79.50	82.00	82.12	82.77	84.75	83.44	**85.64**
Camera	88.50	85.20	85.00	87.05	88.07	90.01	90.02	**90.47**
DVD	81.75	81.70	78.00	83.71	84.77	85.17	85.52	**85.53**
Electronic	81.75	80.50	76.70	82.51	82.33	82.50	83.25	**83.92**
Health	85.25	84.50	83.50	85.52	85.89	84.92	86.50	**87.42**
IMDB	81.25	81.70	81.50	86.02	86.55	85.42	**87.15**	85.53
Kitchen	80.25	78.00	81.50	82.22	83.77	**85.67**	84.54	84.03
Magazine	88.75	89.20	87.70	92.52	92.89	90.83	**93.75**	91.67
MR	74.50	72.70	72.00	75.73	75.98	76.50	76.20	**78.19**
Music	80.00	76.70	77.00	78.74	80.45	80.67	82.04	**83.81**
Software	86.50	84.70	85.50	86.73	86.97	**89.50**	87.75	88.19
Sports	84.25	81.70	82.50	84.04	84.78	84.42	85.75	**85.92**
Toys	85.50	83.20	83.20	85.72	85.82	85.67	85.25	**86.72**
Video	81.50	81.50	83.70	84.73	85.23	85.25	86.75	**86.89**
Average	83.20	81.79	81.73	84.25	84.88	85.12	85.62	**86.08**

posed in [26]. In the experiments, we shared the parameters of our model for all 16 datasets. It can be seen that our model achieved the highest results for

Table 5: Accuracy of 16 datasets with multi-task.

Dataset	CNN	SPMTL	ASPMTL	A-MTLSC	A-MTLBC	Transformer	SLSTM	FCSR
Apparel	85.00	86.50	87.00	87.50	86.20	85.58	87.00	**88.50**
Baby	86.50	86.70	88.20	86.50	88.00	87.00	89.00	**89.92**
Books	85.00	81.20	84.00	83.20	83.70	85.00	**88.50**	88.00
Camera	88.50	88.00	89.20	88.20	89.70	85.33	92.00	**92.67**
DVD	84.75	84.70	86.80	85.50	85.70	86.00	86.00	**87.25**
Electronic	85.50	84.70	85.20	82.20	83.20	83.50	88.25	**88.25**
Health	87.00	87.20	88.20	87.70	86.50	85.42	88.50	**90.50**
IMDB	84.75	84.70	85.50	87.50	86.70	87.33	88.00	**88.17**
Kitchen	84.25	85.20	86.20	84.70	85.00	84.25	87.25	**88.75**
Magazine	89.50	92.00	92.20	91.20	90.50	87.50	94.25	**94.67**
MR	72.25	76.00	**76.70**	75.20	76.50	75.08	71.75	76.42
Music	83.25	83.00	82.50	82.50	81.70	82.58	86.25	**86.67**
Software	89.00	87.00	87.20	85.50	88.20	88.08	88.00	**90.08**
Sports	89.25	87.20	85.70	86.70	86.50	84.91	88.75	**89.75**
Toys	88.50	85.20	88.00	87.00	88.20	87.50	88.50	**91.33**
Video	87.00	83.20	84.50	85.20	85.20	84.08	86.75	**88.08**
Average	85.63	85.15	86.01	85.39	85.72	84.92	87.42	**88.69**

most of these datasets. The average accuracy achieved about 1.5 percent improvement. The results again indicate that our full-scale policy can effectively collect context information. Besides, this knowledge can be beneficial for other learning tasks.

Figure 7 shows the training time of the compared models. It can be seen that the FCSR achieved the highest computational efficiency. The efficiency of Transformer looks relatively competitive. However, when the number of heads and layers is increased, for improving the performance, the time consumption, together with space consumption, will scale up rapidly as well. The training time of S-LSTM was obviously shorter than that of LSTMs. This is because the main operations of LSTMs are transforming hidden states serially, which limits the parallel computing power of GPU. Similarly, when the layers of S-LSTM is increased, for enhancing the performance, the time consumption will also grow sharply. In contrast, our model used two sub-cells instead of multi-layer structure to collect full context information for each target word. Consequently, it didn't undergo any hidden states update steps and made full use of parallel computing power of GPU.

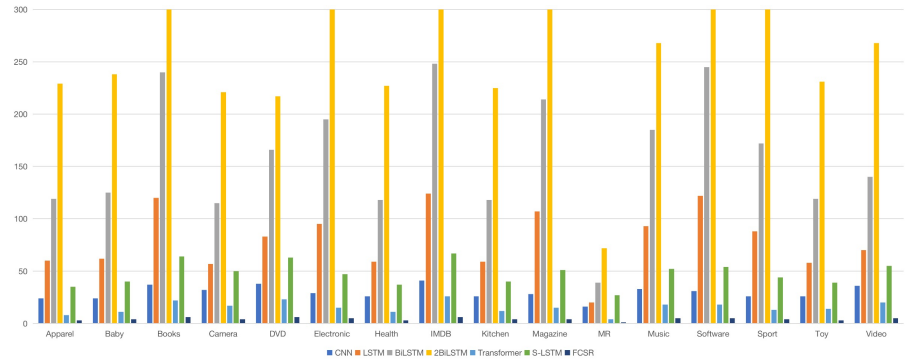

Figure 7: Training time (s/batch). In the experiment, when the number of heads and layers grew bigger than 6, it was unable to run transformer for lack of memory. Therefore, we reduced the batch size to 2 (20 for other models), and its real time consumption is about 10 times higher than that shown.

6. Conclusion

In this paper, we introduce a shallow model for learning full-context semantic representation. With the lightweight context encoding algorithm, FOFE, the left and right contexts of a word are transformed into two context vectors. Then the word and its both left and right context vectors are fed into two sub-cells to generate the semantic representation with respect to this target word. The overall semantic representation is finally learned by combining all the individual representations. The comparisons and experiments show that the proposed model is capable of learning deep semantic representation with a simple one-layer structure, rather than a multi-layer structure and a full connection and multi-attention based structure. The unique structure also ensures the superior computation efficiency of the model.

References

[1] B. Hu, Z. Lu, H. Li, Q. Chen, Convolutional neural network architectures for matching natural language sentences, in: Advances in neural information processing systems, 2014, pp. 2042–2050.

[2] J. L. Elman, Finding structure in time, Cognitive science 14 (2) (1990) 179–211.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[4] R. Al-Rfou, D. Choe, N. Constant, M. Guo, L. Jones, Character-level language modeling with deeper self-attention, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3159–3166.

[5] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Networks 18 (5-6) (2005) 602–610.

[6] Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, R. Salakhutdinov, Transformer-xl: Attentive language models beyond a fixed-length context, arXiv preprint arXiv:1901.02860.

[7] D. Dennis, D. A. E. Acar, V. Mandikal, V. S. Sadasivan, V. Saligrama, H. V. Simhadri, P. Jain, Shallow rnn: accurate time-series classification on resource constrained devices, in: Advances in Neural Information Processing Systems, 2019, pp. 12896–12906.

[8] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.

[9] S. Merity, Single headed attention rnn: Stop thinking with your head, arXiv preprint arXiv:1911.11423.

[10] X. Liu, P. He, W. Chen, J. Gao, Multi-task deep neural networks for natural language understanding, arXiv preprint arXiv:1901.11504.

[11] K. Schouten, F. Frasincar, Survey on aspect-level sentiment analysis, IEEE Transactions on Knowledge and Data Engineering 28 (3) (2016) 813–830.

[12] J. Huang, M. Peng, H. Wang, J. Cao, W. Gao, X. Zhang, A probabilistic method for emerging topic tracking in microblog stream, World Wide Web 20 (2) (2017) 325–350.

[13] Y. Gong, H. Luo, J. Zhang, Natural language inference over interaction space, arXiv preprint arXiv:1709.04348.

[14] S. Hochreiter, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, A Field Guide to Dynamical Recurrent Neural Networks (2001) 237–244.

[15] A. Graves, Generating sequences with recurrent neural networks, arXiv preprint arXiv:1308.0850.

[16] Y. Zhang, Q. Liu, L. Song, Sentence-state lstm for text representation, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 317–327.

[17] X. Qiu, X. Huang, Convolutional neural tensor network architecture for community-based question answering, in: Proceedings of the 24th International Conference on Artificial Intelligence, 2015, pp. 1305–1311.

[18] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, X. Cheng, A deep architecture for semantic matching with multiple positional sentence representations, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 2835–2841.

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

[20] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of NAACL-HLT, 2018, pp. 2227–2237.

[21] A. Conneau, H. Schwenk, L. Barrault, Y. Lecun, Very deep convolutional networks for text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, 2017, pp. 1107–1116.

[22] X. Wang, W. Jiang, Z. Luo, Combination of convolutional and recurrent neural network for sentiment analysis of short texts, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 2428–2437.

[23] X. Ma, E. Hovy, End-to-end sequence labeling via bi-directional lstm-cnns-crf, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2016, pp. 1064–1074.

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

[25] Y. N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: International Conference on Machine Learning, 2017, pp. 933–941.

[26] P. Liu, X. Qiu, X. Huang, Adversarial multi-task learning for text classification, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 1–10.

[27] S. Zhang, H. Jiang, M. Xu, J. Hou, L. Dai, The fixed-size ordinally-forgetting encoding method for neural network language models, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), Vol. 2, 2015, pp. 495–500.

[28] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1631–1642.

[29] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Advances in neural information processing systems, 2015, pp. 649–657.

[30] A. Conneau, H. Schwenk, L. Barrault, Y. Lecun, Very deep convolutional networks for natural language processing, arXiv preprint arXiv:1606.01781 2.

[31] B. Pang, L. Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, in: Proceedings of the 43rd annual meeting on association for computational linguistics, Association for Computational Linguistics, 2005, pp. 115–124.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[33] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[34] G. Niu, H. Xu, B. He, X. Xiao, H. Wu, S. Gao, Enhancing local feature extraction with global representation for neural text classification, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 496–506. doi:10.18653/v1/D19-1047.
URL https://www.aclweb.org/anthology/D19-1047

[35] S. Gururangan, T. Dang, D. Card, N. A. Smith, Variational pretraining for semi-supervised text classification, in: Proceedings of the 57th Annual

Meeting of the Association for Computational Linguistics, 2019, pp. 5880–5894.

[36] J. Jiang, J. Zhang, K. Zhang, Cascaded semantic and positional self-attention network for document classification, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, 2020, pp. 669–677.