Enhancing Recurrent Neural Networks with Sememes

Yujia Qin, Fanchao Qi, Sicong Ouyang, Zhiyuan Liu, Cheng Yang, Yasheng Wang, Qun Liu, Maosong Sun

***Abstract*—Sememes, the minimum semantic units of human languages, have been successfully utilized in various natural language processing applications. However, most existing studies exploit sememes in specific tasks and few efforts are made to utilize sememes more fundamentally. In this paper, we propose to incorporate sememes into recurrent neural networks (RNNs) to improve their sequence modeling ability, which is beneficial to all kinds of downstream tasks. We design three different sememe incorporation methods and employ them in typical RNNs including LSTM, GRU and their bidirectional variants. For evaluation, we use several benchmark datasets involving PTB and WikiText-2 for language modeling, SNLI for natural language inference. Experimental results show evident and consistent improvement of our sememe-incorporated models compared with vanilla RNNs, which proves the effectiveness of our sememe incorporation methods. Moreover, we find the sememe-incorporated models have great robustness and outperform adversarial training in defending adversarial attack. All the code and data of this work will be made available to the public.**

***Index Terms*—Sememe, Recurrent Neural Network.**

I. Introduction

A word is the smallest unit of language, but its meaning can be split into smaller elements, i.e., *sememes*. In linguistics, a sememe is defined as the minimum unit of semantics [1]. Some linguists have the opinion that the meanings of all the words can be represented with a limited set of sememes, which is similar to the idea of *semantic primitives* [2]. Considering sememes are usually implicit in words, researchers build sememe knowledge bases (KBs), which contain many words manually annotated with a set of predefined sememes, to utilize them. With the help of sememe KBs, sememes have been successfully applied to various natural language processing (NLP) tasks, e.g., word similarity computation [3], sentiment analysis [4], word representation learning [5] and lexicon expansion [6].

However, existing work usually exploits sememes for specific tasks and few efforts are made to utilize sememes in a more general and fundamental fashion. [7] make an attempt to incorporate sememes into a long short-term memory (LSTM) [8] language model to improve its performance. Nevertheless, their method uses sememes in the decoder step only and as a result, it is not applicable to other sequence modeling tasks. To the best of our knowledge, no previous work tries to employ sememes to model better text sequences and achieve higher performance of downstream tasks.

In this paper, we propose to incorporate sememes into recurrent neural networks (RNNs) to improve their general sequence modeling ability, which is beneficial to all kinds of downstream NLP tasks. Some studies have tried to incorporate other linguistic knowledge into RNNs [9], [10], [11], [12]. However, almost all of them utilize word-level KBs, which comprise relations between words, e.g., WordNet [13] and ConceptNet [14]. Different from these KBs, sememe KBs use semantically infra-word elements (sememes) to compositionally explain meanings of words and focus on the relations between sememes and words. Therefore, it is difficult to directly adopt previous methods to incorporate sememes into RNNs.

To tackle this challenge, we specifically design three methods of incorporating sememes into RNNs. All of them are highly adaptable and work on different RNN architecture. We employ these methods in two typical RNNs including LSTM, gated recurrent unit (GRU) and their bidirectional variants. In experiments, we evaluate the sememe-incorporated and vanilla RNNs on the benchmark datasets of two representative sequence modeling tasks, namely language modeling and natural language inference. Experimental results show that the sememe-incorporated RNNs achieve consistent performance improvement on all the tasks, which demonstrates the effectiveness of our sememe-incorporation methods and the usefulness of sememes. We also make a case study to explain the benefit of sememes in the task of natural language inference. Furthermore, we conduct an adversarial attack experiment, finding the sememe-incorporated RNNs display great robustness and perform much better than adversarial training.

In conclusion, our contribution consists in: (1) making the first exploration of utilizing sememes to improve the general sequence modeling ability of RNNs; and (2) proposing three effective and highly adaptable methods of incorporating sememes into RNNs.

Y. Qin and F. Qi contribute equally to this work. *(Corresponding author: Zhiyuan Liu)*

Y. Qin is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (email: qinyj16@mails.tsinghua.edu.cn).

F. Qi, Z. L and M. Sun are with Department of Computer Science and Technology, Institute for Artificial Intelligence, Tsinghua University and Beijing National Research Center for Information Science and Technology, Beijing 100084, China (email: qfc17@mails.tsinghua.edu.cn; liuzy@tsinghua.edu.cn; sms@tsinghua.edu.cn).

S. Ouyang and C. Yang are with Beijing University of Posts and Telecommunications, Beijing 100876, China (email: scouyang4354@gmail.com; albertyang33@gmail.com).

Y. Wang and Q. Liu are with Huawei Noah's Ark Lab, Hong Kong, China (email: wangyasheng@huawei.com; qun.liu@huawei.com).

II. Background

In this section, we first introduce the sememe annotation in HowNet, the sememe KB we utilize. Then we give a brief introduction to two typical RNNs, namely LSTM and GRU, together with their bidirectional variants.

A. *Sememe Annotation in HowNet*

HowNet [15] is one of the most famous sememe KBs, which contains over 100 thousand Chinese and English words annotated with about 2,000 predefined sememes. Sememe annotation in HowNet is sense-level. In other words, each sense of polysemous words is annotated with one or more sememes with hierarchical structures. Fig. 1 illustrates the sememe annotation of the word "cardinal" in HowNet. As shown in the figure, "cardinal" has two senses in HowNet, namely "cardinal (important)" and "cardinal (bishop)". The former sense is annotated with only one sememe `important`, while the latter sense has one main sememe `human` and three subsidiary sememes including `religion`, `official` and `ProperName`.

In this paper, we focus on the meanings of sememes and ignore their hierarchical structures for simplicity. Thus, we simply equip each word with a sememe set, which comprises all the sememes annotated to the senses of the word. For instance, the sememe set of "cardinal" is {`important`, `human`, `religion`, `official`, `ProperName`}. We leave the utilization of sememe structures for future work.

B. *Introduction to Typical RNNs*

RNN is a class of artificial neural network designed for processing temporal sequences. LSTM and GRU are two of the most predominant RNNs. They have been widely used in recent years owing to their superior sequence modeling ability.

An LSTM consists of multiple identical cells and each cell corresponds to a token in the input sequence. For each cell, it takes the embedding of the corresponding token $\mathbf{x}_t$ as input to update its cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$. Different from the basic RNN, LSTM integrates a forget gate $\mathbf{f}_t$, an input gate $\mathbf{i}_t$ and an output gate $\mathbf{o}_t$ into its cell, which can alleviate the gradient vanishing issue of the basic RNN. Given the hidden state $\mathbf{h}_{t-1}$ and the cell state $\mathbf{c}_{t-1}$ of the previous cell, the cell state $\mathbf{c}_t$ and the hidden state $\mathbf{h}_t$ of the current cell can be computed by:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{x}_t;\mathbf{h}_{t-1}] + \mathbf{b}_f),\\
\mathbf{i}_t &= \sigma(\mathbf{W}_I[\mathbf{x}_t;\mathbf{h}_{t-1}] + \mathbf{b}_I),\\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c[\mathbf{x}_t;\mathbf{h}_{t-1}] + \mathbf{b}_c),\\
\mathbf{c}_t &= \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{c}}_t,\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{x}_t;\mathbf{h}_{t-1}] + \mathbf{b}_o),\\
\mathbf{h}_t &= \mathbf{o}_t * \tanh(\mathbf{c}_t),
\end{aligned} \tag{1}$$

where $\mathbf{W}_f$, $\mathbf{W}_I$, $\mathbf{W}_c$ and $\mathbf{W}_o$ are weight matrices, and $\mathbf{b}_f$, $\mathbf{b}_I$, $\mathbf{b}_c$ and $\mathbf{b}_o$ are bias vectors. σ is the sigmoid function, $[\ ]$ denotes the concatenation operation and $*$ indicates element wise multiplication. The structure of an LSTM cell is illustrated in the upper left corner of Fig. 2.

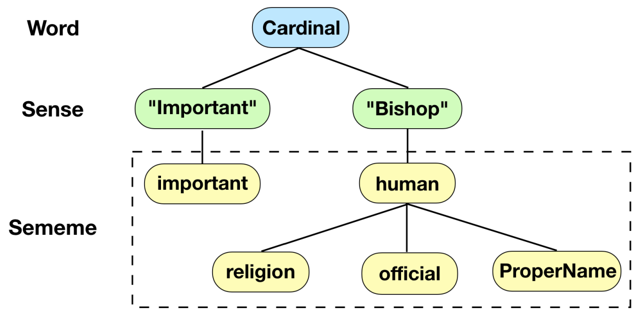

Fig. 1. An example of how words are annotated with sememes in HowNet.

GRU is another popular extension of the basic RNN. It has fewer gates than LSTM. In addition to the input $\mathbf{x}_t$ and hidden state $\mathbf{h}_t$, each GRU cell embodies a update gate $\mathbf{z}_t$ and a reset gate $\mathbf{r}_t$. The transition equations of GRU are as follows:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z[\mathbf{x}_t;\mathbf{h}_{t-1}] + \mathbf{b}_z),\\
\mathbf{r}_t &= \sigma(\mathbf{W}_r[\mathbf{x}_t;\mathbf{h}_{t-1}] + \mathbf{b}_r),\\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_h[\mathbf{x}_t;\mathbf{r}_t * \mathbf{h}_{t-1}] + \mathbf{b}_h),\\
\mathbf{h}_t &= (\mathbf{1} - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}}_t,
\end{aligned} \tag{2}$$

where $\mathbf{W}_z$, $\mathbf{W}_r$ $\mathbf{W}_h$ are weight matrices, and $\mathbf{b}_z$, $\mathbf{b}_r$, $\mathbf{b}_h$ are bias vectors. The lower left corner of Fig. 2 shows the structure of a GRU cell.

Since both LSTM and GRU can only process sequences unidirectionally, their bidirectional variants, namely BiLSTM and BiGRU, are proposed to eliminate the restriction. BiLSTM and BiGRU have two sequences of cells: one processes the input sequence from left to right and the other from right to left. Hence, each token in the input sequence corresponds two unidirectional hidden states, which are concatenated into bidirectional hidden states:

$$\left[\overleftrightarrow{\mathbf{h}}_1, \overleftrightarrow{\mathbf{h}}_2, ..., \overleftrightarrow{\mathbf{h}}_T\right] = \begin{bmatrix} \overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, ..., \overrightarrow{\mathbf{h}}_T, \\ \overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, ..., \overleftarrow{\mathbf{h}}_T, \end{bmatrix}, \tag{3}$$

where T denotes the length of the input sequence.

III. Sememe Incorporation Methods

In this section, we elaborately describe three methods of incorporating sememes into RNNs, namely simple concatenation (+concat), adding sememe output gate (+gate) and introducing sememe-RNN cell (+cell). All the three methods are applicable to both LSTM and GRU, both unidirectional and bidirectional RNNs. The following descriptions are based on unidirectional LSTM and GRU, and the newly added variables and formulae are underlined for clarity. Bidirectional sememe-incorporated LSTMs and GRUs are just a trivial matter of concatenating the hidden states of their unidirectional versions. The cell structures of three different sememe-incorporated LSTMs and GRUs are exhibited in Fig. 2.

A. *Simple Concatenation*

The first sememe incorporation approach is quite straightforward. It simply concatenates the sum of sememe embeddings of a word with the corresponding word embedding. In this

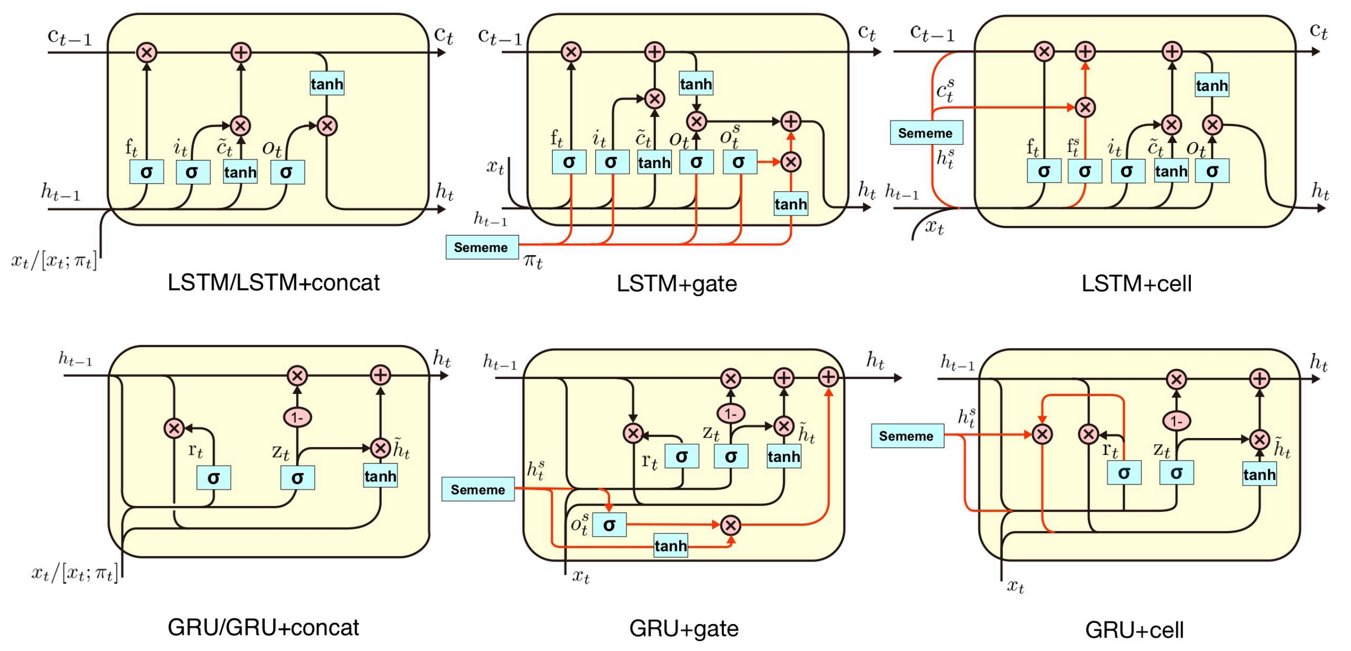

Fig. 2. The cell structures of three sememe-incorporated LSTMs and GRUs. The three figures in the first row illustrate the cell structures of LSTM (and with simple concatenation of sememe embeddings), LSTM with sememe output gate and LSTM with sememe-LSTM cell. The three figures in the second row exhibit the cell structures of GRU (and with simple concatenation of sememe embeddings), GRU with sememe output gate and GRU with sememe-GRU cell.

way, sememe knowledge is incorporated into a RNN cell via its input (word embedding). Formally, given a word x_t, its sememe set is $\mathcal{S}_t = \{s_1, \cdots, s_{|\mathcal{S}_t|}\}$, where $|\cdot|$ denotes the cardinality of a set. We concatenate its original word embedding $\mathbf{x}_t$ with sememe knowledge embedding $\boldsymbol{\pi}_t$:

$$\boldsymbol{\pi}_t = \sum_{s \in \mathcal{S}_t} \mathbf{s} \quad \tilde{\mathbf{x}}_t \quad = [\mathbf{x}_t; \boldsymbol{\pi}_t], \tag{4}$$

where $\mathbf{s}$ denotes the sememe embedding of s, and $\tilde{\mathbf{x}}_t$ is the retrofitted word embedding which incorporates sememe knowledge.

B. Adding Sememe Output Gate

In the first method, sememe knowledge is shallowly incorporated into RNN cells. It is essentially a kind of enhancement of word embeddings. Inspired by [16], we propose the second method which enables deep incorporation of sememes into RNN cells. More specifically, we add an additional sememe output gate $\mathbf{o}^s_t$ to the vanilla RNN cell, which decides how much sememe knowledge is absorbed into the hidden state. Meanwhile, the original gates are also affected by the sememe knowledge term. The transition equations of LSTM with the sememe output gate are as follows:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\boldsymbol{\pi}_t}] + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(\mathbf{W}_I[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\boldsymbol{\pi}_t}] + \mathbf{b}_i), \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}_c), \\
\mathbf{c}_t &= \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{c}}_t, \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\boldsymbol{\pi}_t}] + \mathbf{b}_o), \\
\underline{\mathbf{o}^s_t} &= \underline{\sigma(\mathbf{W}_{o^s}[\mathbf{x}_t; \mathbf{h}_{t-1}; \boldsymbol{\pi}_t] + \mathbf{b}_{o^s})}, \\
\mathbf{h}_t &= \mathbf{o}_t * \tanh(\mathbf{c}_t) + \underline{\mathbf{o}^s_t * \tanh(\mathbf{W}_c \boldsymbol{\pi}_t)},
\end{aligned} \tag{5}$$

where $\mathbf{W}_{o^s}$ is a weight matrix and $\mathbf{b}_{o^s}$ is a bias vector.

We can find that $\underline{\mathbf{o}^s_t * \tanh(\mathbf{W}_c \boldsymbol{\pi}_t)}$ can directly add the term of sememe knowledge to the original hidden state. By doing this, the original hidden state, which carries contextual information only, is enhanced by semantic information of sememe knowledge.

The sememe output gate can be added to GRU in a similar way, and the transition equations are as follows:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\boldsymbol{\pi}_t}] + \mathbf{b}_z), \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\boldsymbol{\pi}_t}] + \mathbf{b}_r), \\
\underline{\mathbf{o}^s_t} &= \underline{\sigma(\mathbf{W}_o[\mathbf{x}_t; \mathbf{h}_{t-1}; \boldsymbol{\pi}_t] + \mathbf{b}_o)}, \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_h[\mathbf{x}_t; \mathbf{r}_t * \mathbf{h}_{t-1}] + \mathbf{b}_h), \\
\mathbf{h}_t &= (\mathbf{1} - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}}_t + \underline{\mathbf{o}^s_t \tanh(\boldsymbol{\pi}_t)},
\end{aligned} \tag{6}$$

where $\mathbf{o}^s_t$ is the sememe output gate, $\mathbf{W}_o$ is a weight matrix and $\mathbf{b}_o$ is a bias vector.

C. Introducing Sememe-RNN Cell

In the second method, although sememe knowledge is incorporated into RNN cells directly and deeply, it can be utilized more sufficiently. Taking LSTM for example, as shown in Equation (5), the hidden state $\mathbf{h}_t$ comprises two parts. The first part is the contextual information item $\mathbf{o}_t * \tanh(\mathbf{c}_t)$. It bears the information of current word and preceding text, which has been processed by the forget gate and encoded into the cell state. The second part is the sememe knowledge item $\underline{\mathbf{o}^s_t * \tanh(\mathbf{W}_c \boldsymbol{\pi}_t)}$. It carries the information of sememe knowledge which has not processed or encoded. Therefore, the two parts are inconsistent.

To address the issue, we propose the third sememe incorporation method. We regard the sememe knowledge as another information source like the previous word and introduce an extra RNN cell to encode it. Specifically, we first feed the sememe knowledge embedding to an LSTM cell (sememe-LSTM cell) and obtain its cell and hidden states which carry sememe knowledge. Then we design a special forget gate for the sememe knowledge and use it to process the cell state of the sememe-LSTM cell, just as the previous cell state. Finally, we add the processed cell state of the sememe-LSTM to the original cell state. In addition, the hidden state of the sememe-LSTM cell is also absorbed into the input gate and output gate. Formally, the transition equations of LSTM with the sememe-LSTM cell are as follows:

$$\begin{aligned}
\underline{\mathbf{c}_t^s, \mathbf{h}_t^s} &= \underline{\mathrm{LSTM}^{(S)}(\boldsymbol{\pi}_t)},\\
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}_f),\\
\underline{\mathbf{f}_t^s} &= \underline{\sigma(\mathbf{W}_f^s[\mathbf{x}_t; \mathbf{h}_t^s] + \mathbf{b}_f^s)},\\
\mathbf{i}_t &= \sigma(\mathbf{W}_I[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\mathbf{h}_t^s}] + \mathbf{b}_i),\\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\mathbf{h}_t^s}] + \mathbf{b}_c),\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\mathbf{h}_t^s}] + \mathbf{b}_o),\\
\mathbf{c}_t &= \mathbf{f}_t * \mathbf{c}_{t-1} + \underline{\mathbf{f}_t^s * \mathbf{c}_t^s} + \mathbf{i}_t * \tilde{\mathbf{c}}_t,\\
\mathbf{h}_t &= \mathbf{o}_t * \tanh(\mathbf{c}_t),
\end{aligned} \tag{7}$$

where $\mathbf{c}_t^s$ and $\mathbf{h}_t^s$ are the cell state and hidden state of the sememe-LSTM cell, $\mathbf{f}_t^s$ is the sememe forget gate, $\mathbf{W}_{f^s}$ is a weight matrix and $\mathbf{b}_{f^s}$ is a bias vector.

Similarly, we can introduce a sememe-GRU cell to the original GRU and the transition equations are as follows:

$$\begin{aligned}
\underline{\mathbf{h}_t^s} &= \underline{GRU^{(S)}(\boldsymbol{\pi}_t)},\\
\mathbf{z}_t &= \sigma(\mathbf{W}_z[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\mathbf{h}_t^s}] + \mathbf{b}_z),\\
\mathbf{r}_t &= \sigma(\mathbf{W}_r[\mathbf{x}_t; \mathbf{h}_{t-1}; \underline{\mathbf{h}_t^s}] + \mathbf{b}_r),\\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_h[\mathbf{x}_t; \mathbf{r}_t * (\mathbf{h}_{t-1} + \underline{\mathbf{h}_t^s})] + \mathbf{b}_h),\\
\mathbf{h}_t &= (\mathbf{1} - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}}_t,
\end{aligned} \tag{8}$$

where $\mathbf{h}_t^s$ is the hidden state of the sememe-GRU cell.

IV. Language Modeling

In this section, we evaluate our sememe-incorporate RNNs on the task of language modeling (LM).

A. Dataset

We use two benchmark LM datasets, namely Penn Treebank (PTB) [17] and WikiText-2 [18]. PTB is made up of articles from the Wall Street Journal. Its vocabulary size is $10,000$. The token numbers of its training, validation and test sets are $887,521$, $70,390$ and $78,669$ respectively. WikiText-2 comprises Wikipedia articles and its vocabulary size is $33,278$. It has $2,088,628$, $217,646$ and $245,569$ tokens in its training, validation and test sets.

We choose HowNet as the source of sememes. It contains $2,186$ different sememes and $43,321$ English words with sememe annotation. We use the open-source API of HowNet, OpenHowNet [19], to obtain annotated sememes of a word. The numbers of sememe-annotated tokens in PTB and WikiText-2 are $870,520$ (83.98%) and $2,068,779$ (81.07%) respectively.

TABLE I
Perplexity Results of All the Models on the Validation and Test Sets of PTB and WikiText-2.

Dataset	PTB		WikiText-2	
Model	Validation	Test	Validation	Test
LSTM(medium)	85.14	81.74	98.87	93.69
+concat	81.79	78.90	96.05	91.41
+gate	81.15	77.73	95.27	90.19
+cell	**79.71**	**76.57**	**94.49**	**89.39**
LSTM(large)	81.88	78.34	96.86	91.07
+concat	78.35	75.25	92.72	87.51
+gate	77.02	73.90	91.02	86.16
+cell	**76.47**	**73.01**	**90.44**	**85.62**
GRU(medium)	94.84	91.05	109.11	103.07
+concat	90.68	87.29	105.11	98.89
+gate	**88.87**	**85.12**	**103.13**	**96.97**
+cell	89.56	86.49	103.53	97.65
GRU(large)	92.68	89.60	107.75	101.52
+concat	91.64	87.62	104.33	98.15
+gate	**88.08**	**84.49**	102.84	96.71
+cell	89.33	86.05	**101.22**	**95.67**

B. Experimental Settings

a) Baseline Methods: We choose the vanilla LSTM and GRU as the baseline methods. Notice that bidirectional RNNs are generally not used in the LM task because they are not allowed to know the whole sentence.

b) Hyper-parameters: Following previous work, we try the models on two sets of hyper-parameters, namely "medium" and "large". For "medium", the dimension of hidden states and word/sememe embeddings is set to 650, the batch size is 20, and the dropout rate is 0.5. For "large", the dimension of vectors is 1500, the dropout rate is 0.65, and other hyper-parameters are the same as "medium". All the word and sememe embeddings are randomly initialized as real-valued vectors using a normal distribution with mean 0 and variance 0.05. The above-mentioned hyper-parameter settings are applied to all the models.

c) Training Strategy: We also adopt the same training strategy for all the models. We choose stochastic gradient descent (SGD) as the optimizer, whose initial learning rate is 20 for LSTM and 10 for GRU. The learning rate would be divided by 4 if no improvement is observed on the validation set. The maximum training epoch number is 40 and the gradient norm clip boundary is 0.25.

C. Experimental Results

Table I shows the perplexity results on both validation and test sets of the two datasets. From the table, we can observe that:

(1) All the sememe-incorporated RNNs, including the simplest "+concat" models, achieve lower perplexity as compared

to corresponding vanilla RNNs, which demonstrates the usefulness of sememes for enhancing sequence modeling and the effectiveness of our sememe incorporation methods;

(2) Among the three different methods, "+cell" performs best for LSTM and "+gate" performs best for GRU at both "medium" and "large" hyper-parameter settings. The possible explanation is that "+cell" incorporates limited sememe knowledge into GRU as compared to LSTM. In fact, GRU+cell is much less affected by sememe knowledge than GRU+gate, as shown in Equation 8.

(3) By comparing the results of the "large" vanilla RNNs and the "medium" sememe-incorporated RNNs, we can exclude the possibility that better performance of the sememe-incorporated RNNs is brought by more parameters. For example, the perplexity of the "large" vanilla LSTM and the "medium" LSTM+cell on the four sets is 81.88/79.71, 78.34/76.57, 96.86/94.49 and 91.07/89.39 respectively. Although the "large" vanilla LSTM has much more parameters than "medium" LSTM+cell (76M vs. 24M), it is still outperformed by the latter.

V. Natural Language Inference

Natural language inference (NLI), also known as recognizing textual entailment, is a classic sentence pair classification task.In this section, we evaluate our models on the NLI task.

A. Dataset

We choose the most famous benchmark dataset, Stanford Natural Language Inference (SNLI) [20] dataset for evaluation. It contains 570k sentence pairs, each of which comprises a premise and a hypothesis. The relations of the sentence pairs are manually classified into 3 categories, namely "entailment", "contradiction" and "neutral". In addition, the coverage of its sememe-annotated tokens is 82.31%.

B. Experimental Settings

a) Baseline Methods: In addition to vanilla LSTM, GRU and their bidirectional variants, we design two extra baseline methods. For the first one, we randomly assign each word some meaningless labels in the same quantity as its sememes and substitute sememes by these labels, where the total number of different labels is also equal to that of sememes. For the second one, we use WordNet as the source of external knowledge and substitute the sememes of a word by its synonyms with same POS tag.

b) Hyper-parameters: We use 300-dimensional word embeddings pre-trained by GloVe [21], which are frozen during training. Embeddings of sememes and meaningless labels are randomly initialized as 300-dimensional real-valued vectors using a normal distribution with mean 0 and variance 0.05. The dropout rate for input word embedding is 0.2. The dimension of hidden states is 2048. In addition, other hyper-parameters are the same as those of the LM experiment.

c) Training Strategy: We still choose the SGD optimizer, whose initial learning rate is 0.1 and weight factor is 0.99. We divide the learning rate by 5 if no improvement is observed on the validation dataset.

TABLE II
Accuracy Results of all the Models on the Test Set of SNLI.

Knowledge	Model	LSTM	GRU	BiLSTM	BiGRU
None	vanilla	80.73	81.91	81.96	81.92
Meaningless Label	+concat	80.81	81.37	82.13	82.16
	+gate	79.37	80.93	80.84	79.35
	+cell	78.92	81.52	81.78	81.24
WordNet	+concat	80.19	81.36	82.14	82.37
	+gate	80.97	81.78	82.45	81.68
	+cell	81.42	81.75	82.33	81.79
Sememe	+concat	**81.72**	81.87	82.39	82.70
	+gate	81.61	82.33	83.06	83.18
	+cell	81.63	**82.76**	**83.30**	**83.87**

d) Classifier: Following previous work [22], [23], we employ a three-layer perceptron plus a three-way softmax layer as the classifier, whose input is a feature vector constructed from the embeddings of a pair of sentences. Specifically, we use any RNN model to process the two sentences of a premise-hypothesis pair and obtain their embeddings $\mathbf{h}_{pre}$ and $\mathbf{h}_{hyp}$. Then we construct the feature vector $\mathbf{v}$ as follows:

$$\mathbf{v} = \begin{bmatrix} \mathbf{h}_{pre} \\ \mathbf{h}_{hyp} \\ |\mathbf{h}_{pre} - \mathbf{h}_{hyp}| \\ \mathbf{h}_{pre} * \mathbf{h}_{hyp} \end{bmatrix}. \tag{9}$$

C. Experimental Results

Table II lists the results of all the models on the test set of SNLI. From this table, we can see that:

(1) All the sememe-incorporated models achieve marked performance enhancement compared with corresponding vanilla models, which proves the usefulness of sememes in improving the sentence representation ability of RNNs and the effectiveness of our sememe incorporation methods;

(2) By comparing the results of the models with the same knowledge incorporation method but different external knowledge, the superiority of sememes is further confirmed. When sememes are substituted by meaningless labels or WordNet synonyms, the performance of any knowledge-incorporated model declines and becomes even worse than that of vanilla RNNs.

(3) Among the three sememe incorporation methods, "+cell" achieves the best overall performance, which manifests the great efficiency of "+cell" in utilizing sememe knowledge. This is also consistent with the conclusion of the LM experiment basically.

D. Case Study

In this subsection, we use two examples to illustrate how sememes are beneficial to handling the task of NLI. Table III exhibits two premise-hypothesis pairs in the SNLI dataset, where the sememes of some important words are appended to corresponding words.

TABLE III
Two Examples of Premise-hypothesis Pairs in the SNLI dataset. The Words In Italic Type are Important Words and their Sememes are Appended.

Entailment Example **Premise**: Four men stand in a circle facing each other *playing* [`perform`, `reaction`, `MusicTool`] brass *instruments* [`MusicTool`, `implement`] which people watch them. **Hypothesis**: The men are playing *music* [`music`].
Contradict Example **Premise**: A group of women playing volleyball *indoors* [`location`, `house`, `internal`]. **Hypothesis**: People are *outside* [`location`, `external`] tossing a ball.

For the first premise-hypothesis pair, its relation type is annotated as "entailment" in SNLI. Our sememe-incorporated models yield the correct result while all the baseline methods do not. We notice that there are several important words whose sememes provide useful information. In the premise, both the words "playing" and "instruments" have the sememe `MusicTool`, which is semantically related to the sememe `music` of the word "music" in the hypothesis. We speculate that the semantic relatedness given by sememes assists our models in coping with this sentence pair.

The second premise-hypothesis pair, whose true relation is "contradict", is also classified correctly by our models but wrongly by the baseline methods. We find that the word "indoors" in the premise has the sememe `internal`, while the word "outside" in the hypothesis has the sememe `external`. The two sememes are a pair of antonyms, which may explain why our models make the right judgment.

E. Adversarial Attack Experiment

Adversarial attack and defense have attracted considerable research attention recently [24], [25], because they can disclose and fix the vulnerability of neural networks that small perturbations of input can cause significant changes of output. Adversarial attack is aimed at generating adversarial examples to fool a neural model (victim model) and adversarial defense is targeted at improving the robustness of the model against attack. In the field of NLP, all kinds of adversarial attack methods have been proposed [26] but few efforts are made in adversarial defense [27].

We intuitively believe that incorporating sememes can improve the robustness of neural networks, because sememes are general linguistic knowledge and complementary to text corpora on which neural networks heavily rely. Therefore, we evaluate the robustness of our sememe-incorporated RNNs against adversarial attack.

a) Attack Method: We use a genetic algorithm-based attack model [28], which is a typical gradient-free black-box attack method. It generates adversarial examples by substituting words of model input iteratively and achieves impressive attack performance on both sentiment analysis and NLI.

b) Baseline Methods: Besides the WordNet-incorporated RNNs, we choose adversarial training [29] as the baseline methods. Adversarial training is believed to be an effective defense method. It adds some generated adversarial examples to the training set, aiming to generalize the victim model to the adversarial attack.

c) Experimental Settings: For the attack method, we use all the recommended hyper-parameters of its original work [28]. For the victim models including vanilla RNNs together with their sememe and WordNet-incorporated versions, the hyper-parameters and training strategy are the same as those in the previous experiment. For adversarial training, we add 57k (10% of the number of total sentence pairs in SNLI) generated adversarial examples (premise-hypothesis pairs) to the training set. We use the attack success rate (%) as the evaluation metric. The lower the attack success rate is, the more robust a model is.

d) Experimental Results: Table IV lists the success rates of adversarial attack against all the victim models. We can observe that:

(1) The attack success rates of our sememe-incorporated RNNs are consistently lower than those of the vanilla RNNs, which indicates the sememe-incorporated RNNs have greater robustness against adversarial attack. Furthermore, the experimental results also verify the effectiveness of sememes in improving robustness of neural networks.

(2) Among the three sememe incorporation methods, "+cell" beats the other two once again. It demonstrates the advantage of "+cell" in properly incorporating sememes into RNNs.

(3) Some models incorporated with WordNet have lower attack access rates than corresponding vanilla RNNs. However, from an overall perspective, incorporating WordNet into RNNs only brings negligible decrease of the attack access rate. It indicates WordNet has very limited capability to improve the robustness of neural networks, especially compared with sememes.

(4) Adversarial training increases the attack success rates of all the models rather than decrease them, which is consistent with the findings of previous work [28]. It shows adversarial training is not an effective defense method, at least for the attack we use. In fact, there are few effective adversarial defense methods, which makes the superiority of sememes in defending adversarial attack and improving robustness of models more valuable.

VI. Related Work

A. HowNet and Its Applications

HowNet [15] is one of the most famous sememe KBs, whose construction takes several linguistic experts more than two decades. After HowNet is published, it has been employed in diverse NLP tasks including word similarity computation [3], word sense disambiguation [30], word representation learning [5], sentiment analysis [4], semantic composition [31], adversarial attack [32], reverse dictionary [33], etc. There are also some studies trying to expand HowNet [34], [35] and transfer its sememe knowledge to other languages [36], [37].

[7] exploit sememes in a sequence modeling task (language modeling) for the first time. They add a sememe predictor

TABLE IV
SUCCESS RATES (%) OF ADVERSARIAL ATTACK AGAINST ALL THE DEFENSE MODELS ON THE TEST SET OF SNLI. "VANILLA+AT" REPRESENTS ADVERSARIAL TRAINING.

Knowledge	Model	LSTM	GRU	BiLSTM	BiGRU
None	vanilla	64.8	65.4	63.9	64.7
None	vanilla+at	66.2	67.4	65.9	66.3
WordNet	+concat	63.2	66.4	63.6	64.8
	+gate	64.7	65.6	63.8	64.2
	+cell	64.3	66.1	64.5	63.9
Sememe	+concat	64.5	65.3	63.3	64.2
	+gate	62.9	65.1	62.5	63.6
	+cell	63.9	64.9	62.3	63.2

to an LSTM language model, which is aimed at predicting sememes of the next word using preceding text. Then they use the predicted sememes to predict the next word. Nevertheless, their method uses sememes in the decoder step of the LSTM language model and does not improve the LSTM's ability to encode sequences. Therefore, it cannot be applied to other sequence modeling tasks. As far as we know, we are the first to utilize sememes to improve general sequence modeling ability of neural networks.

B. Recurrent Neural Networks

The recurrent neural network (RNN) [38] and its representative extentions including LSTM [8] and GRU [39], have been widely employed in various NLP tasks, e.g., language modeling [40], sentiment analysis [41], semantic role labelling [42], dependency parsing [43] and natural language inference [44].

To improve the sequence modeling ability of RNNs, some researches try to reform the frameworks of RNNs, e.g., integrating the attention mechanism [45], adding hierarchical structures [46] and introducing bidirectional modeling [47].

In addition, some work focuses on incorporating different kinds of external knowledge into RNNs. General linguistic knowledge from famous KBs such as WordNet [13] and ConceptNet [14] attracts considerable attention. These KBs usually comprise relations between words, which are hard to be incorporated into the internal structures of RNNs. Therefore, most existing knowledge-incorporated methods employ external linguistic knowledge on the hidden layers of RNNs rather than the internal structures of RNN cells [9], [10], [11], [12].

Since sememe knowledge is very different from the word-level relational knowledge, it cannot be incorporated into RNNs with the same methods. As far as we know, no previous work tries to incorporate such knowledge as sememes into RNNs.

VII. CONCLUSION AND FUTURE WORK

In this paper, we make the first attempt to incorporate sememes into RNNs to enhance their sequence modeling ability, which is beneficial to many downstream NLP tasks. We preliminarily propose three highly adaptable sememe incorporation methods, and employ them in typical RNNs including LSTM, GRU and their bidirectional versions. In experiments, we evaluate our methods on the benchmark datasets of language modeling and natural language inference. Experimental results show that sememe-incorporated RNNs achieve obvious performance improvement, which demonstrates the usefulness of sememes and effectiveness of our methods.

In the future, we will explore following directions including: (1) considering the hierarchical structures of sememes, which contain more semantic information of words; (2) using attention mechanism to adjust the weights of sememes in different context to take better advantage of sememes; (3) evaluating our sememe-incorporated RNNs on other sequence modeling tasks; and (4) incorporating sememe knowledge into other neural models such as the transformer.

REFERENCES

[1] L. Bloomfield, "A set of postulates for the science of language," *Language*, vol. 2, no. 3, pp. 153–164, 1926.

[2] A. Wierzbicka, *Semantics: Primes and universals: Primes and universals*. Oxford University Press, UK, 1996.

[3] Q. Liu and S. Li, "Word similarity computing based on hownet," *International Journal of Computational Linguistics & Chinese Language Processing*, vol. 7, no. 2, pp. 59–76, 2002.

[4] X. Fu, G. Liu, Y. Guo, and Z. Wang, "Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon," *Knowledge-Based Systems*, vol. 37, pp. 186–195, 2013.

[5] Y. Niu, R. Xie, Z. Liu, and M. Sun, "Improved word representation learning with sememes," in *Proceedings of ACL*, 2017.

[6] X. Zeng, C. Yang, C. Tu, Z. Liu, and M. Sun, "Chinese liwc lexicon expansion via hierarchical classification of word embeddings with sememe attention," in *Proceedings of AAAI*, 2018.

[7] Y. Gu, J. Yan, H. Zhu, Z. Liu, R. Xie, M. Sun, F. Lin, and L. Lin, "Language modeling with sparse product of sememe experts," in *Proceedings of EMNLP*, 2018.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio, "A neural knowledge language model," *arXiv preprint arXiv:1608.00318*, 2016.

[10] B. Yang and T. Mitchell, "Leveraging knowledge bases in lstms for improving machine reading," in *Proceedings of ACL*, 2017.

[11] P. Parthasarathi and J. Pineau, "Extending neural generative conversational model using external knowledge sources," in *Proceedings of EMNLP*, 2018.

[12] T. Young, E. Cambria, I. Chaturvedi, H. Zhou, S. Biswas, and M. Huang, "Augmenting end-to-end dialogue systems with commonsense knowledge," in *Proceedings of AAAI*, 2018.

[13] G. Miller, *WordNet: An electronic lexical database*. MIT press, 1998.

[14] R. Speer and C. Havasi, "ConceptNet 5: A large semantic network for relational knowledge," in *The People's Web Meets NLP*. Springer, 2013, pp. 161–176.

[15] Z. Dong and Q. Dong, "Hownet-a hybrid language and knowledge resource," in *Proceedings of NLP-KE*, 2003.

[16] Y. Ma, H. Peng, and E. Cambria, "Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm," in *Proceedings of AAAI*, 2018.

[17] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, no. 2, 1993.

[18] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *Proceedings of ICLR*, 2017.

[19] F. Qi, C. Yang, Z. Liu, Q. Dong, M. Sun, and Z. Dong, "Openhownet: An open sememe-based lexical knowledge base," *arXiv preprint arXiv:1901.09957*, 2019.

[20] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of EMNLP*, 2015.

[21] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of EMNLP*, 2014.

[22] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts, "A fast unified model for parsing and sentence understanding," in *Proceedings of ACL*, 2016.

[23] L. Mou, R. Men, G. Li, Y. Xu, L. Zhang, R. Yan, and Z. Jin, "Natural language inference by tree-based convolution and heuristic matching," in *Proceedings of ACL*, 2016.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proceedings of ICLR*, 2014.

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of ICLR*, 2015.

[26] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and L. Chenliang, "Adversarial attacks on deep learning models in natural language processing: A survey," *arXiv preprint arXiv:1901.06796*, 2019.

[27] W. Wang, L. Wang, B. Tang, R. Wang, and A. Ye, "A survey: Towards a robust deep neural network in text domain," *arXiv preprint arXiv:1902.07285*, 2019.

[28] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," in *Proceedings of EMNLP*, 2018.

[29] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proceedings of ICLR*, 2015.

[30] X. Duan, J. Zhao, and B. Xu, "Word sense disambiguation through sememe labeling." in *Proceedings of IJCAI*, 2007.

[31] F. Qi, J. Huang, C. Yang, Z. Liu, X. Chen, Q. Liu, and M. Sun, "Modeling semantic compositionality with sememe knowledge," in *Proceedings of ACL*, 2019.

[32] Y. Zang, C. Yang, F. Qi, Z. Liu, M. Zhang, Q. Liu, and M. Sun, "Textual adversarial attack as combinatorial optimization," *arXiv preprint arXiv:1910.12196*, 2019.

[33] L. Zhang, F. Qi, Z. Liu, Y. Wang, Q. Liu, and M. Sun, "Multi-channel reverse dictionary model," in *Proceedings of AAAI*, 2020.

[34] R. Xie, X. Yuan, Z. Liu, and M. Sun, "Lexical sememe prediction via word embeddings and matrix factorization," in *Proceedings of AAAI*, 2017.

[35] H. Jin, H. Zhu, Z. Liu, R. Xie, M. Sun, F. Lin, and L. Lin, "Incorporating chinese characters of words for lexical sememe prediction," in *Proceedings of ACL*, 2018.

[36] F. Qi, Y. Lin, M. Sun, H. Zhu, R. Xie, and Z. Liu, "Cross-lingual lexical sememe prediction," in *Proceedings of EMNLP*, 2018.

[37] F. Qi, L. Chang, M. Sun, O. Sicong, and Z. Liu, "Towards building a multilingual sememe knowledge base: Predicting sememes for BabelNet synsets," in *Proceedings of AAAI*, 2020.

[38] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[39] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of EMNLP*, 2014.

[40] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[41] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov, "Semeval-2016 task 4: Sentiment analysis in twitter," in *Proceedings of SemEval*, 2016.

[42] L. He, K. Lee, O. Levy, and L. Zettlemoyer, "Jointly predicting predicates and arguments in neural semantic role labeling," in *Proceedings of ACL*, 2018.

[43] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional lstm feature representations," *TACL*, vol. 4, pp. 313–327, 2016.

[44] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proceedings of EMNLP*, 2016.

[45] P. Bachman and D. Precup, "Variational generative stochastic networks with collaborative shaping." in *Proceedings of ICML*, 2015.

[46] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.

[47] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *Proceedings of International Conference on Artificial Neural Networks*, 2005.