Language Modeling with Sparse Product of Sememe Experts

Yihong Gu[1,2,*] **Jun Yan**[1,3,*] **Hao Zhu**[1,2,*] **Zhiyuan Liu**[1,2,†]
Ruobing Xie[4] **Maosong Sun**[1,2] **Fen Lin**[4] **Leyu Lin**[4]
[1]Institute for Artificial Intelligence
State Key Lab on Intelligent Technology and Systems
[2]Department of CST, [3]Department of EE, Tsinghua University, Beijing, China
[4]Search Product Center, WeChat Search Application Department, Tencent
{gyh15,j-yan15,zhuhao15}@mails.tsinghua.edu.cn,
{lzy,sms}@tsinghua.edu.cn, xrbsnowing@163.com,
{felicialin,goshawklin}@tencent.com

Abstract

Most language modeling methods rely on large-scale data to statistically learn the sequential patterns of words. In this paper, we argue that words are atomic language units but not necessarily atomic semantic units. Inspired by HowNet, we use sememes, the minimum semantic units in human languages, to represent the implicit semantics behind words for language modeling, named Sememe-Driven Language Model (SDLM). More specifically, to predict the next word, SDLM first estimates the sememe distribution given textual context. Afterwards, it regards each sememe as a distinct semantic expert, and these experts jointly identify the most probable senses and the corresponding word. In this way, SDLM enables language models to work beyond word-level manipulation to fine-grained sememe-level semantics, and offers us more powerful tools to fine-tune language models and improve the interpretability as well as the robustness of language models. Experiments on language modeling and the downstream application of headline generation demonstrate the significant effectiveness of SDLM. Source code and data used in the experiments can be accessed at https://github.com/thunlp/SDLM-pytorch.

1 Introduction

Language Modeling (LM) aims to measure the probability of a word sequence, reflecting its fluency and likelihood as a feasible sentence in a human language. Language Modeling is an essential component in a wide range of natural language processing (NLP) tasks, such as Machine Translation (Brown et al., 1990; Brants et al., 2007), Speech Recognition (Katz, 1987), Information Retrieval (Berger and Lafferty, 1999; Ponte and Croft, 1998; Miller et al., 1999; Hiemstra, 1998) and Document Summarization (Rush et al., 2015; Banko et al., 2000).

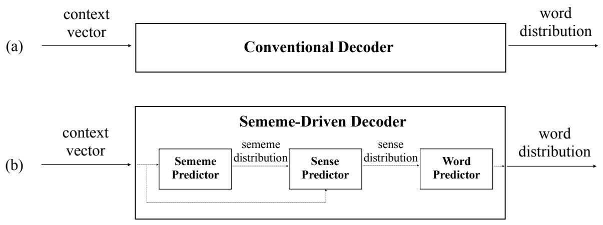

Figure 1: Decoder of (a) Conventional Language Model, (b) Sememe-Driven Language Model.

A probabilistic language model calculates the conditional probability of the next word given their contextual words, which are typically learned from large-scale text corpora. Taking the simplest language model for example, N-Gram estimates the conditional probabilities according to maximum likelihood over text corpora (Jurafsky, 2000). Recent years have also witnessed the advances of Recurrent Neural Networks (RNNs) as the state-of-the-art approach for language modeling (Mikolov et al., 2010), in which the context is represented as a low-dimensional hidden state to predict the next word.

Those conventional language models including neural models typically assume words as atomic symbols and model sequential patterns at word level. However, this assumption does not necessarily hold to some extent. Let us consider the following example sentence for which people want to predict the next word in the blank,

The U.S. trade deficit last year is initially estimated to be 40 billion ______ .

People may first realize a *unit* should be filled in, then realize it should be a *currency unit*. Based on the country this sentence is talking about, the U.S.,

∗ Equal contribution.
† Correspondence author.

one may confirm it should be an *American currency unit* and predict the word *dollars*. Here, the *unit*, *currency*, and *American* can be regarded as basic semantic units of the word *dollars*. This process, however, has not been explicitly taken into consideration by conventional language models. That is, although in most cases words are atomic language units, words are not necessarily atomic semantic units for language modeling. We argue that explicitly modeling these atomic semantic units could improve both the performance and the interpretability of language models.

Linguists assume that there is a limited close set of atomic semantic units composing the semantic meanings of an open set of concepts (i.e. word senses). These atomic semantic units are named **sememes** (Dong and Dong, 2006).[i] Since sememes are naturally implicit in human languages, linguists have devoted much effort to explicitly annotate lexical sememes for words and build linguistic common-sense knowledge bases. HowNet (Dong and Dong, 2006) is one of the representative sememe knowledge bases, which annotates each Chinese word sense with its sememes. The philosophy of HowNet regards the parts and attributes of a concept can be well represented by sememes. HowNet has been widely utilized in many NLP tasks such as word similarity computation (Liu, 2002) and sentiment analysis (Fu et al., 2013). However, less effort has been devoted to exploring its effectiveness in language models, especially neural language models.

It is non-trivial for neural language models to incorporate discrete sememe knowledge, as it is not compatible with continuous representations in neural models. In this paper, we propose a Sememe-Driven Language Model (SDLM) to leverage lexical sememe knowledge. In order to predict the next word, we design a novel sememe-sense-word generation process: (1) We first estimate sememes' distribution according to the context. (2) Regarding these sememes as experts, we propose a sparse product of experts method to select the most probable senses. (3) Finally, the distribution of words could be easily calculated by marginalizing out the distribution of senses.

We evaluate the performance of SDLM on the language modeling task using a Chinese newspaper corpus People's Daily [ii] (Renmin Ribao), and also on the headline generation task using the Large Scale Chinese Short Text Summarization (LCSTS) dataset (Hu et al., 2015). Experimental results show that SDLM outperforms all those data-driven baseline models. We also conduct case studies to show that our model can effectively predict relevant sememes given context, which can improve the interpretability and robustness of language models.

i Note that although sememes are defined as the minimum semantic units, there still exist several sememes for capturing syntactic information. For example, the word 和 "with" corresponds to one specific sememe 功能词 "FunctWord".

2 Background

Language models target at learning the joint probability of a sequence of words $P(w^1, w^2, \cdots, w^n)$, which is usually factorized as $P(w^1, w^2, \cdots, w^n) = \prod_{t=1}^{n} P(w^t | w^{<t})$. Bengio et al. (2003) propose the first Neural Language Model as a feed-forward neural network. Mikolov et al. (2010) use RNN and a softmax layer to model the conditional probability. To be specific, it can be divided into two parts in series. First, a context vector $\mathbf{g}^t$ is derived from a deep recurrent neural network. Then, the probability $P(w^{t+1}|w^{\leq t}) = P(w^{t+1}; \mathbf{g}^t)$ is derived from a linear layer followed by a softmax layer based on $\mathbf{g}^t$. Let $\mathrm{RNN}(\cdot, \cdot; \boldsymbol{\theta}_{\mathrm{NN}})$ denote the deep recurrent neural network, where $\boldsymbol{\theta}_{\mathrm{NN}}$ denotes the parameters. The first part can be formulated as

$$\mathbf{g}^t = \mathrm{RNN}(\mathbf{x}_{w^t}, \{\mathbf{h}_l^{t-1}\}_{l=1}^{L}; \boldsymbol{\theta}_{\mathrm{NN}}). \quad (1)$$

Here we use subscripts to denote layers and superscripts to denote timesteps. Thus $\mathbf{h}_l^t$ represents the hidden state of the L-th layer at timestep t. $\mathbf{x}_{w^t} \in \mathbb{R}^{H_0}$ is the input embedding of word w^t where H_0 is the input embedding size. We also have $\mathbf{g}^t \in \mathbb{R}^{H_1}$, where H_1 is the dimension of the context vector.

Supposing that there are N words in the language we want to model, the second part can be written as

$$P(w^{t+1}; \mathbf{g}^t) = \frac{\exp({\mathbf{g}^t}^{\mathrm{T}} \mathbf{w}_{w^{t+1}})}{\sum_{w'} \exp({\mathbf{g}^t}^{\mathrm{T}} \mathbf{w}_{w'})}, \quad (2)$$

where $\mathbf{w}_w$ is the output embedding of word w and $\mathbf{w}_1, \mathbf{w}_2, \cdots \mathbf{w}_N \in \mathbb{R}^{H_2}$. Here H_2 is the output embedding size. For a conventional neural language model, H_2 always equals to H_1.

ii http://paper.people.com.cn/rmrb/

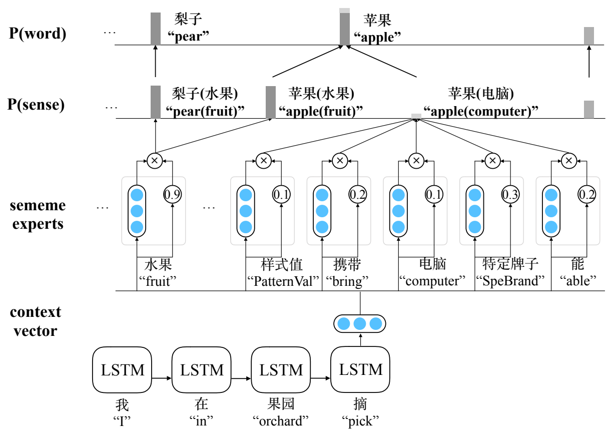

Figure 2: An example of the architecture of our model.

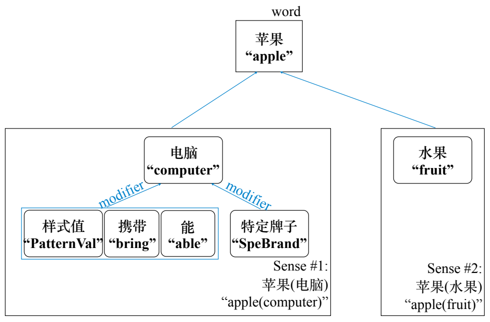

Figure 3: An example of the word-sense-sememe hierarchy.

Given the corpus $\{w^t\}_{t=1}^n$, the loss function is defined by the negative log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{t=1}^{n} \log P(w^t | w^{<t}; \boldsymbol{\theta}), \qquad (3)$$

where $\boldsymbol{\theta} = \{\{\mathbf{x}_i\}_{i=1}^N, \{\mathbf{w}_i\}_{i=1}^N, \boldsymbol{\theta}_{\text{NN}}\}$ is the set of parameters that are needed to be trained.

3 Methodology

In this section, we present our SDLM which utilizes sememe information to predict the probability of the next word. SDLM is composed of three modules in series: Sememe Predictor, Sense Predictor and Word Predictor. The Sememe Predictor first takes the context vector as input and assigns a weight to each sememe. Then each sememe is regarded as an expert and makes predictions about the probability distribution over a set of senses in the Sense Predictor. Finally, the probability of each word is obtained in the Word Predictor.

Here we use an example shown in Figure 2 to illustrate our architecture. Given context 我在果园摘 "In the orchard, I pick", the actual next word could be 苹果 "apples". From the context, especially the word 果园 "orchard" and 摘 "pick", we can infer that the next word probably represents a kind of fruit. So the Sememe Predictor assigns a higher weight to the sememe 水果 "fruit" (0.9) and lower weights to irrelevant sememes like 电脑 "computer" (0.1). Therefore in the Sense Predictor, the sense 苹果 (水果) "apple (fruit)" is assigned a much higher probability than the sense 苹果 (电脑) "apple (computer)". Finally, the probability of the word 苹果 "apple" is calculated as the sum of the probabilities of its senses 苹果 (水果) "apple(fruit)" and 苹果 (电脑) "apple (computer)".

In the following subsections, we first introduce the word-sense-sememe hierarchy in HowNet, and then give details about our SDLM.

3.1 Word-Sense-Sememe Hierarchy

We also use the example of "apple" to illustrate the word-sense-sememe hierarchy. As shown in Figure 3, the word 苹果 "apple" has two senses, one is the Apple brand, the other is a kind of fruit. Each sense is annotated with several sememes organized in a hierarchical structure. More specifically, in HowNet, sememes "PatternVal", "bring", "SpeBrand", "computer" and "able" are annotated with the word "apple" and organized in a tree structure. In this paper, we ignore the structural relationship between sememes. For each word, we group all its sememes as an unordered set.

We present the notations that we use in the following subsections as follows. We define the overall sememe, sense, and word set as $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{W}$. And we suppose the corpus contains $K = |\mathcal{E}|$ sememes, $M = |\mathcal{S}|$ senses and $N = |\mathcal{W}|$ words. For word $w \in \mathcal{W}$, we denote its corresponding sense set as $\mathcal{S}^{(w)}$. For sense $s \in \mathcal{S}^{(w)}$, we denote its corresponding sememes as an unordered set $\mathcal{E}^{(s)} = \{e_{n_1}, e_{n_2}, \cdots, e_{n_k}\} \subset \mathcal{E} = \{e_k\}_{k=1}^K$.

3.2 Sememe Predictor

The Sememe Predictor takes the context vector $\mathbf{g} \in \mathbb{R}^{H_1}$ as input and assigns a weight to each sememe. We assume that given the context $w^1, w^2, \cdots, w^{t-1}$, the events that word w^t contains sememe e_k ($k \in \{1, 2, \cdots, K\}$) are independent, since the sememe is the minimum semantic unit and there is no semantic overlap between any two different sememes. For simplicity, we ignore the superscript t. We design the Sememe Predictor as a linear decoder with the sigmoid activation function. Therefore, q_k, the probability that the next word contains sememe e_k, is formulated as

$$q_k = P(e_k|\mathbf{g}) = \sigma(\mathbf{g}^{\mathrm{T}}\mathbf{v}_k + b_k), \tag{4}$$

where $\mathbf{v}_k \in \mathbb{R}^{H_1}$, $b_k \in \mathbb{R}$ are trainable parameters, and $\sigma(\cdot)$ denotes the sigmoid activation function.

3.3 Sense Predictor and Word Predictor

The architecture of the Sense Predictor is motivated by Product of Experts (PoE) (Hinton, 1999). We regard each sememe as an expert that only makes predictions on the senses connected with it. Let $\mathcal{D}^{(e_k)}$ denote the set of senses that contain sememe e_k, the k-th expert. Different from conventional neural language models, which directly use the inner product of the context vector $\mathbf{g} \in \mathbb{R}^{H_1}$ and the output embedding $\mathbf{w}_w \in \mathbb{R}^{H_2}$ for word w to generate the score for each word, we use $\phi^{(k)}(\mathbf{g}, \mathbf{w})$ to calculate the score given by expert e_k. And we choose a bilinear function parameterized with a matrix $\mathbf{U}_k \in \mathbb{R}^{H_1 \times H_2}$ as a straight implementation of $\phi^{(k)}(\cdot, \cdot)$:

$$\phi^{(k)}(\mathbf{g}, \mathbf{w}) = \mathbf{g}^{\mathrm{T}}\mathbf{U}_k\mathbf{w}. \tag{5}$$

Let $\mathbf{w}_s$ denote the output embedding of sense s. The score of sense s provided by sememe expert e_k can be written as $\phi^{(k)}(\mathbf{g}, \mathbf{w}_s)$. Therefore, $P^{(e_k)}(s|\mathbf{g})$, the probability of sense s given by expert e_k, is formulated as

$$P^{(e_k)}(s|\mathbf{g}) = \frac{\exp(q_k C_{k,s} \phi^{(k)}(\mathbf{g}, \mathbf{w}_s))}{\sum_{s' \in \mathcal{D}^{(e_k)}} \exp(q_k C_{k,s'} \phi^{(k)}(\mathbf{g}, \mathbf{w}_{s'}))}, \tag{6}$$

where $C_{k,s}$ is a normalization constant because sense s is not connected to all experts (the connections are sparse with approximately λN edges, $\lambda < 5$). Here we can choose either $C_{k,s} = 1/|\mathcal{E}^{(s)}|$ (*left normalization*) or $C_{k,s} = 1/\sqrt{|\mathcal{E}^{(s)}||\mathcal{D}^{(e_k)}|}$ (*symmetric normalization*).

In the Sense Predictor, q_k can be viewed as a gate which controls the magnitude of the term $C_{k,s}\phi^{(k)}(\mathbf{g}, \mathbf{w}_{w_s})$, thus control the flatness of the sense distribution provided by sememe expert e_k. Consider the extreme case when $q_k \to 0$, the prediction will converge to the discrete uniform distribution. Intuitively, it means that the sememe expert will refuse to provide any useful information when it is not likely to be related to the next word.

Finally, we summarize the predictions on sense s by taking the product of the probabilities given by relevant experts and then normalize the result; that is to say, $P(s|\mathbf{g})$, the probability of sense s, satisfies

$$P(s|\mathbf{g}) \propto \prod_{e_k \in \mathcal{E}^{(s)}} P^{(e_k)}(s|\mathbf{g}). \tag{7}$$

Using Equation 5 and 6, we can formulate $P(s|\mathbf{g})$ as

$$P(s|\mathbf{g}) = \frac{\exp(\sum_{e_k \in \mathcal{E}^{(s)}} q_k C_{k,s} \mathbf{g}^{\mathrm{T}}\mathbf{U}_k\mathbf{w}_s)}{\sum_{s'} \exp(\sum_{e_k \in \mathcal{E}^{(s')}} q_k C_{k,s'} \mathbf{g}^{\mathrm{T}}\mathbf{U}_k\mathbf{w}_{s'})}. \tag{8}$$

It should be emphasized that all the supervision information provided by HowNet is embodied in the connections between the sememe experts and the senses. If the model wants to assign a high probability to sense s, it must assign a high probability to some of its relevant sememes. If the model wants to assign a low probability to sense s, it can assign a low probability to its relevant sememes. Moreover, the prediction made by sememe expert e_k has its own tendency because of its own $\phi^{(k)}(\cdot, \cdot)$. Besides, the sparsity of connections between experts and senses is also determined by HowNet itself. For our dataset, on average, a word is connected with 3.4 sememe experts and each sememe expert will make predictions about 22 senses.

As illustrated in Figure 2, in the Word Predictor, we get $P(w|\mathbf{g})$, the probability of word w, by summing up probabilities of corresponding s given by the Sense Predictor, that is

$$P(w|\mathbf{g}) = \sum_{s \in \mathcal{S}^{(w)}} P(s|\mathbf{g}). \tag{9}$$

3.4 Implementation Details

Basis Matrix Actually, HowNet contains $K \approx 2000$ sememes. In practice, we cannot directly introduce $K \times H_1 \times H_2$ parameters, which might be computationally infeasible and lead to overfitting. To address this problem, we apply a weight-sharing trick called the basis matrix. We use R basis matrices and their weighted sum to estimate $\mathbf{U}_k$:

$$\mathbf{U}_k = \sum_{r=1}^{R} \alpha_{k,r} \mathbf{Q}_r, \tag{10}$$

where $\mathbf{Q}_r \in \mathbb{R}^{H_1 \times H_2}$, $\alpha_{k,r} > 0$ are trainable parameters, and $\sum_{r=1}^{R} \alpha_{k,r} = 1$.

Weight Tying To incorporate the weight tying strategy (Inan et al., 2017; Press and Wolf, 2017), we use the same output embedding for multiple senses of a word. To be specific, the sense output embedding $\mathbf{w}_s$ for each $s \in S^{(w)}$ is the same as the word input embedding $\mathbf{x}_w$.

4 Experiments

We evaluate our SDLM on a Chinese language modeling dataset, namely People's Daily based on perplexity.[iii] Furthermore, to show that our SDLM structure can be a generic Chinese word-level decoder for sequence-to-sequence learning, we conduct a Chinese headline generation experiment on the LCSTS dataset. Finally, we explore the interpretability of our model with cases, showing the effectiveness of utilizing sememe knowledge.

4.1 Language Modeling

Dataset

We choose the People's Daily Corpus, which is widely used for Chinese NLP tasks, as the resource. It contains one month's news text from People's Daily (Renmin Ribao). Taking Penn Treebank (PTB) (Marcus et al., 1993) as a reference, we build a dataset for Chinese language modeling based on the People's Daily Corpus with 734k, 10k and 19k words in the training, validation and test set. After the preprocessing similar to (Mikolov et al., 2010) (see Appendix A), we get our dataset and the final vocabulary size is 13,476.

Baseline

As for baselines, we consider three kinds of neural language modeling architectures with LSTM cells: simple LSTM, Tied LSTM and AWD-LSTM.

LSTM and Tied LSTM Zaremba et al. (2014) use the dropout strategy to prevent overfitting for neural language models and adopt it to two-layer LSTMs with different embedding and hidden size: 650 for medium LSTM, and 1500 for large LSTM. Employing the weight tying strategy, we get Tied LSTM with better performance. We set LSTM and Tied LSTM of medium and large size as our baseline models and use the code from PyTorch examples[iv] as their implementations.

AWD-LSTM Based on several strategies for regularizing and optimizing LSTM-based language models, Merity et al. (2018) propose AWD-LSTM as a three-layer neural network, which serves as a very strong baseline for word-level language modeling. We build it with the code released by the authors[v].

Variants of Softmax Meanwhile, to compare our SDLM with other language modeling decoders, we set cHSM (Class-based Hierarchical Softmax) (Goodman, 2001), tHSM (Tree-based Hierarchical Softmax) (Mikolov et al., 2013) and MoS (Mixture of Softmaxes) (Yang et al., 2018) as the baseline add-on structures to the architectures above.

Experimental Settings

We apply our SDLM and other variants of softmax structures to the architectures mentioned above: LSTM (medium / large), Tied LSTM (medium / large) and AWD-LSTM. MoS and SDLM are only applied on the models that incorporate weight tying, while tHSM is only applied on the models without weight tying, since it is not compatible with this strategy.

For a fair comparison, we train these models with same experimental settings and conduct a hyper-parameter search for baselines as well as our models (the search setting and the optimal hyper-parameters can be found in Appendix C.1). We keep using these hyper-parameters in our SDLM for all architectures. It should be emphasized that we use the SGD optimizer for all architectures, and we decrease the learning rate by a factor of 2 if no improvement is observed on the validation set. We uniformly initialize the word embeddings, the class embeddings for cHSM and the non-leaf embeddings for tHSM in $[-0.1, 0.1]$. In addition, we set R, the number of basis matrices, to 5 in Tied LSTM architecture and to 10 in AWD-LSTM architecture. We choose the *left normalization* strategy because it performs better.

Experimental Results

Table 1 shows the perplexity on the validation and test set of our models and the baseline models. From Table 1, 2, and 3, we can observe that:

1. Our models outperform the corresponding baseline models of all structures, which indicates the effectiveness of our SDLM. Moreover, our SDLM not only consistently outperforms state-of-the-art MoS model, but also offers much better interpretability (as described in Sect. 4.3), which

iii Although we only conduct experiments on Chinese corpora, we argue that this model has the potential to be applied to other languages in the light of works on construction sememe knowledge bases for other languages, such as (Qi et al., 2018).

iv `https://github.com/pytorch/examples/tree/master/word_language_model`

v `https://github.com/salesforce/awd-lstm-lm` makes it possible to interpret the prediction process of the language model. Note that under a fair comparison, we do not see MoS's improvement over AWD-LSTM while our SDLM outperforms it by 1.20 with respect to perplexity on the test set.

2. To further locate the performance improvement of our SDLM, we study the perplexity of the single-sense words and multi-sense words separately on Tied LSTM (medium) and Tied LSTM (medium) + SDLM. Improvements with respect to perplexity are presented in Table 2. The performance on both single-sense words and multi-sense words gets improved while multi-sense words benefit more from SDLM structure because they have richer sememe information.

3. In Table 3 we study the perplexity of words with different mean number of sememes. We can see that our model outperforms baselines in all cases and is expected to benefit more as the mean number of sememes increases.

Model	#Paras	Validation	Test
LSTM (medium)	24M	116.46	115.51
+ cHSM	24M	129.12	128.12
+ tHSM	24M	151.00	150.87
Tied LSTM (medium)	15M	105.35	104.67
+ cHSM	15M	116.78	115.66
+ MoS	17M	98.47	98.12
+ SDLM	17M	**97.75**	**97.32**
LSTM (large)	76M	112.39	111.66
+ cHSM	76M	120.07	119.45
+ tHSM	76M	140.41	139.61
Tied LSTM (large)	56M	101.46	100.71
+ cHSM	56M	108.28	107.52
+ MoS	67M	94.91	94.40
+ SDLM	67M	**94.24**	**93.60**
AWD-LSTM[iv]	26M	89.35	88.86
+ MoS	26M	92.98	92.76
+ SDLM	27M	**88.16**	**87.66**

Table 1: Single model perplexity on validation and test sets on the People's Daily dataset.

	#senses = 1	#senses > 1
Baseline ppl	93.21	121.18
SDLM ppl	87.22	111.88
Δppl	5.99	9.29
Δppl/Baseline ppl	6.4%	7.8%

Table 2: Perplexity of words with different number of senses on the test set.

We also test the robustness of our model by randomly removing 10% sememe-sense connections in HowNet. The test perplexity for Tied LSTM (medium) + SDLM slightly goes up to 97.67, compared to 97.32 with a complete HowNet, which shows that our model is robust to tiny incompleteness of annotations. However, the performance of out model is still largely dependent upon the accuracy of sememe annotations. As HowNet is continuously updated, we expect our model to perform better with sememe knowledge of higher quality.

iv We find that multi-layer AWD-LSTM has problems converging when adopting cHSM, so we skip that result.

	[1, 2)	[2, 4)	[4, 7)	[7, 14)
Baseline ppl	71.56	161.32	557.26	623.71
SDLM ppl	68.47	114.95	465.29	476.45
Δppl	3.09	16.36	91.98	147.25
Δppl/Baseline ppl	4.3%	10.1%	16.5%	23.61%

Table 3: Perplexity of words with different mean number of sememes on the test set.

4.2 Headline Generation

Dataset

We use the LCSTS dataset to evaluate our SDLM structure as the decoder of the sequence-to-sequence model. As its author suggests, we divide the dataset into the training set, the validation set and the test set, whose sizes are 2.4M, 8.7k and 725 respectively. Details can be found in Appendix B.

Models

For this task, we consider two models for comparison.

RNN-context As described in (Bahdanau et al., 2015), RNN-context is a basic sequence-to-sequence model with a bi-LSTM encoder, an LSTM decoder and attention mechanism adopted. The context vector is concatenated with the word embedding at each timestep when decoding. It's widely used for sequence-to-sequence learning, so we set it as the baseline model.

RNN-context-SDLM Based on RNN-context, we substitute the decoder with our proposed SDLM and name it RNN-context-SDLM.

Experimental Settings

We implement our models with PyTorch, on top of the OpenNMT libraries[v]. For both models, we set the word embedding size to 250, the hidden unit size to 250, the vocabulary size to 40000, and the beam size of the decoder to 5. For RNN-context-SDLM, we set the number of basis matrices to 3. We conduct a hyper-parameter search for both

v http://opennmt.net

models (see Appendix C.2 for settings and optimal hyper-parameters).

Experimental Results

Following previous works, we report the F1-score of ROUGE-1, ROUGE-2, and ROUGE-L on the test set. Table 4 shows that our model outperforms the baseline model on all metrics. We attribute the improvement to the use of SDLM structure.

Words in headlines do not always appear in the corresponding articles. However, words with the same sememes have a high probability to appear in the articles intuitively. Therefore, a probable reason for the improvement is that our model could predict sememes highly relevant to the article, thus generate more accurate headlines. This could be corroborated by our case study.

Model	Rouge-1	Rouge-2	Rouge-L
RNN-context	37.5	25.0	34.9
RNN-context-SDLM	**38.9**	**26.2**	**36.2**

Table 4: ROUGE scores of both models on the LCSTS test set.

4.3 Case Study

The above experiments demonstrate the effectiveness of our SDLM. Here we present some samples from the test set of the People's Daily Corpus in Table 5 as well as the LCSTS dataset in Table 6 and conduct further analysis.

For each example of language modeling, given the context of previous words, we list the Top 5 words and Top 5 sememes predicted by our SDLM. The target words and the sememes annotated with them in HowNet are blackened. Note that if the target word is an out-of-vocabulary (OOV) word, helpful sememes that are related to the target meaning are blackened.

Example (1)		
去年 美国 贸易逆差 初步 估计 为 <N> ______ 。		
The U.S. trade deficit last year is initially estimated to be <N> ______ .		
Top 5 word prediction		
美元 "dollar"	， ","	。 "."
日元 "yen"	和 "and"	
Top 5 sememe prediction		
商业 "commerce"	**金融 "finance"**	**单位 "unit"**
多少 "amount"	专 "proper name"	
Example (2)		
阿 总理 ______ 已 签署 了 一 项 命令 。		
Albanian Prime Minister ______ has signed an order.		
Top 5 word prediction		
内 "inside"	**<unk>**	在 "at"
塔 "tower"	和 "and"	
Top 5 sememe prediction		
政 "politics"	**人 "person"**	花草 "flowers"
担任 "undertake"	水域 "waters"	

Table 5: Some examples of word and sememe predictions on the test set of the People's Daily Corpus.

Sememes annotated with the corresponding sense of the target word 美元 "dollar" are 单位 "unit", 商业 "commerce", 金融 "finance", 货币 "money" and 美国 "US". In Example (1), the target word "dollar" is predicted correctly and most of its sememes are activated in the predicting process. It indicates that our SDLM has learned the word-sense-sememe hierarchy and used sememe knowledge to improve language modeling.

Example (2) shows that our SDLM can provide interpretable results on OOV word prediction with sememe information associated with it. The target word here should be the name of the Albanian prime minister, which is out of vocabulary. But with our model, one can still conclude that this word is probably relevant to the sememe "politics", "person", "flowers", "undertake" and "waters", most of which characterize the meaning of this OOV word – the name of a politician. This feature can be helpful when the vocabulary size is limited or there are many terminologies and names in the corpus.

For the example of headline generation, given the article and previous words, when generating the word 生 "student", except the sememe 预料 "predict", all other Top 5 predicted sememes have high relevance to either the predicted word or the context. To be specific, the sememe 学习 "study" is annotated with 生 "student" in HowNet. 考试 "exam" indicates "college entrance exam". 特定牌子 "brand" indicates "BMW". And 高等 "higher" indicates "higher education", which is the next step after this exam. We can conclude that with sememe knowledge, our SDLM structure can extract critical information from both the given article and generated words explicitly and produce better summarization based on it.

5 Related Work

Neural Language Modeling. RNNs have achieved state-of-the-art performance in the language modeling task since Mikolov et al. (2010) first apply RNNs for language modeling. Much work has been done to improve RNN-based language modeling. For example, a variety of work (Zaremba et al., 2014; Gal and Ghahramani, 2016; Merity et al., 2017, 2018) introduces many regularization and optimization methods for RNNs. Based on the observation that the word

Article
8 日 ， 阜 新 一 开 宝马 轿车 参加 高考 的 男 考生 考场 作弊 被 抓 ， 因 不满 监考 老师 没收 作弊 手机 ， 从 背后 一 脚 将 女 监考 老师 从 最后 一 排 踹 到 讲台 。 并 口 出 狂 言 ： “ 你 知道 我 爸 是 谁 啊 ， 你 就 查 我 ？ ” 目前 ， 打人 考生 已 被拘留 。
On the 8th in Fuxin, a male student drove a BMW to take the college entrance exam and was caught cheating. Because the teacher confiscated his mobile phone, he kicked the teacher from the last row to the podium and shouted: ”Do you know who my dad is? How dare you catch me!” Currently, this student has been detained.
Gold
男生 高考 作弊 追打 监考 老师 ： 你 知道 我 爸 是 谁 ？
In the college entrance exam, a male student caught cheating hit the teacher: Do you know who my dad is?
RNN-context-SDLM
高考 生 作弊 被 抓 ： 你 知道 我 爸 是 谁 啊 ？
In the college entrance exam, a student was caught cheating: Do you know who my dad is?
Top 5 sememe prediction
考试 “exam” **学习 “study”** **特定牌子 “brand”** 预料 “predict” **高等 “higher”**

Table 6: An example of generated headlines on the LCSTS test set.

appearing in the previous context is more likely to appear again, some work (Grave et al., 2017a,b) proposes to use cache for improvements. In this paper, we mainly focus on the output decoder, the module between the context vector and the predicted probability distribution. Similar to our SDLM, Yang et al. (2018) propose a high-rank model which adopts a Mixture of Softmaxes structure for the output decoder. However, our model is sememe-driven with each expert corresponding to an interpretable sememe.

Hierarchical Decoder Since softmax computation on large vocabulary is time-consuming, therefore being a dominant part of the model's complexity, various hierarchical softmax models have been proposed to address this issue. These models can be categorized to class-based models and tree-based models according to their hierarchical structure. Goodman (2001) first proposes the class-based model which divides the whole vocabulary into different classes and uses a hierarchical softmax decoder to model the probability as $\mathbb{P}(\text{word}) = \mathbb{P}(\text{word}|\text{class})\mathbb{P}(\text{class})$, which is similar to our model. For the tree-based models, all words are organized in a tree structure and the word probability is calculated as the probability of always choosing the correct child along the path from the root node to the word node. While Morin and Bengio (2005) utilize knowledge from WordNet to build the tree, Mnih and Hinton (2008) build it in a bootstrapping way and Mikolov et al. (2013) construct a Huffman Tree based on word frequencies. Recently, Jiang et al. (2017) reform the tree-based structure to make it more efficient on GPUs. The major differences between our model and theirs are the purpose and the motivation. Our model targets at improving the performance and interpretability of language modeling using external knowledge in HowNet. Therefore, we take its philosophy of the word-sense-sememe hierarchy to design our hierarchical decoder. Meanwhile, the class-based and tree-based models are mainly designed to speed up the softmax computation in the training process.

Sememe. Recently, there are a lot of works concentrating on utilizing sememe knowledge in traditional natural language processing tasks. For example, Niu et al. (2017) use sememe knowledge to improve the quality of word embeddings and cope with the problem of word sense disambiguation. Xie et al. (2017) apply matrix factorization to predict sememes for words. Jin et al. (2018) improve their work by incorporating character-level information. Our work extends the previous works and tries to combine word-sense-sememe hierarchy with the sequential model. To be specific, this is the first work to improve the performance and interpretability of Neural Language Modeling with sememe knowledge.

Product of Experts. As Hinton (1999, 2002) propose, the final probability can be calculated as the product of probabilities given by experts.Gales and Airey (2006) apply PoE to the speech recognition where each expert is a Gaussian mixture model. Unlike their work, in our SDLM, each expert is mapped to a sememe with better interpretability. Moreover, as the final distribution is a categorical distribution, each expert is only responsible for making predictions on a subset of the categories (usually less than 10), so we call it Sparse Product of Experts.

Headline Generation. Headline generation is a kind of text summarization tasks. In recent years, with the advances of RNNs, a lot of works have been done in this domain. The encoder-decoder models (Sutskever et al., 2014; Cho et al., 2014) have achieved great success in sequence-to-sequence learning. Rush et al. (2015) propose a local attention-based model for abstractive sentence summarization. Gu et al. (2016) introduce the copying mechanism which is close to the rote memorization of the human being. Ayana et al. (2016) employ the minimum risk training strategy to optimize model parameters. Different from these works, we focus on the decoder of the sequence-to-sequence model, and adopt SDLM to utilize sememe knowledge for sentence generation.

6 Conclusion and Further Work

In this paper, we propose an interpretable Sememe-Driven Language Model with a hierarchical sememe-sense-word decoder. Besides interpretability, our model also achieves state-of-the-art performance in the Chinese Language Modeling task and shows improvement in the Headline Generation task. These results indicate that SDLM can successfully take advantages of sememe knowledge.

As for future work, we plan the following research directions: (1) In language modeling, given a sequence of words, a sequence of corresponding sememes can also be obtained. We will utilize the context sememe information for better sememe and word prediction. (2) Structural information about sememes in HowNet is ignored in our work. We will extend our model with the hierarchical sememe tree for more accurate relations between words and their sememes. (3) It is imaginable that the performance of SDLM will be significantly influenced by the annotation quality of sememe knowledge. We will also devote to further enrich the sememe knowledge for new words and phrases, and investigate its effect on SDLM.

Acknowledgement

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273) and the research fund of Tsinghua University-Tencent Joint Laboratory for Internet Innovation Technology. This work is also funded by China Association for Science and Technology (2016QNRC001). Hao Zhu and Jun Yan are supported by Tsinghua University Initiative Scientific Research Program. We thank all members of Tsinghua NLP lab. We also thank anonymous reviewers for their careful reading and their insightful comments.

References

Shiqi Shen Ayana, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *arXiv preprint arXiv:1604.01904*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Michele Banko, Vibhu O Mittal, and Michael J Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of ACL*, pages 318–325. Association for Computational Linguistics.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of SIGIR*, pages 222–229. ACM.

Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP*.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Zhendong Dong and Qiang Dong. 2006. *Hownet and the computation of meaning (with Cd-rom)*. World Scientific.

Xianghua Fu, Guo Liu, Yanyan Guo, and Zhiqiang Wang. 2013. Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon. *Knowledge-Based Systems*, 37:186–195.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*.

M. J. F. Gales and S. S. Airey. 2006. Product of gaussians for speech recognition. *Computer Speech and Language*, 20(1):22–40.

J Goodman. 2001. Classes for fast maximum entropy training. In *Proceedings of ICASSP*, pages 561–564 vol.1.

Edouard Grave, Moustapha Cisse, and Armand Joulin. 2017a. Unbounded cache model for online language modeling with open vocabulary. In *Proceedings of NIPS*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017b. Improving neural language models with a continuous cache. In *Proceedings of ICLR*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*, pages 1631–1640.

Djoerd Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of TPDL*, pages 569–584. Springer.

G. E Hinton. 1999. Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on*, pages 1–6 vol.1.

G. E. Hinton. 2002. *Training products of experts by minimizing contrastive divergence.* MIT Press.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of EMNLP*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of ICLR*.

Nan Jiang, Wenge Rong, Min Gao, Yikang Shen, Zhang Xiong, Nan Jiang, Wenge Rong, Min Gao, Yikang Shen, and Zhang Xiong. 2017. Exploration of tree-based hierarchical softmax for recurrent language models. In *Proceedings of IJCAI*.

Huiming Jin, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Incorporating chinese characters of words for lexical sememe prediction. In *Proceedings of ACL*, pages 2439–2449. Association for Computational Linguistics.

Dan Jurafsky. 2000. Speech & language processing. chapter 4. Pearson Education India.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Qun Liu. 2002. Word similarity computing based on hownet. *Computational linguistics and Chinese language processing*, 7(2):59–76.

Mitchell P. Marcus, Beatrice Santorini, and Ann Marcinkiewicz, Mary. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *Proceedings of ICLR*.

Stephen Merity, Bryan Mccann, and Richard Socher. 2017. Revisiting activation regularization for language rnns. *arXiv preprint arXiv:1708.01009*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

David RH Miller, Tim Leek, and Richard M Schwartz. 1999. A hidden markov model information retrieval system. In *Proceedings of SIGIR*, pages 214–221. ACM.

Andriy Mnih and Geoffrey Hinton. 2008. A scalable hierarchical distributed language model. In *Proceedings of NIPS*, pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proceedings of ACL*, volume 1, pages 2049–2058.

Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281. ACM.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of EACL*.

Fanchao Qi, Yankai Lin, Maosong Sun, Hao Zhu, Ruobing Xie, and Zhiyuan Liu. 2018. Cross-lingual lexical sememe prediction. In *Proceedings of EMNLP*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Ruobing Xie, Xingchi Yuan, Zhiyuan Liu, and Maosong Sun. 2017. Lexical sememe prediction via word embeddings and matrix factorization. In *Proceedings of IJCAI*, pages 4200–4206. AAAI Press.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank rnn language model. In *Proceedings of ICLR*.

Wojciech Zaremba, Ilya. Sutskever, and Oriol. Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

A Details about Preprocessing of the People's Daily Dataset

In this section, we describe the details about preprocessing of the People's Daily dataset.

Firstly, we treat sentence, which is segmented by particular punctuations, as the minimum unit and then shuffle the corpus. We split the corpus into the training set, the validation set and the test set, which contain 734k, 10k, 19k words respectively. Similar to the preprocessing performed in (Mikolov et al., 2010), we replace the number with <N>, the specific date with <date>, the year with <year>, and the time with <time>. Different from the preprocessing of the English language modeling dataset, we keep the punctuations and therefore do not append <eos> at the end of each sentence. Those words that occur less than 5 times are replaced with <unk>.

Since our model requires that every word should be included in the dictionary of HowNet, we segment each non-annotated word into annotated words with the forward maximum matching algorithm.

B Details about Preprocessing of the LCSTS Dataset

In this section, we describe the details about preprocessing of the LCSTS dataset.

The dataset consists of over 2 million article-headline pairs collected from Sina Weibo, the most popular social media network in China. It's composed of three parts. Each pair from PART-II and PART-III is labeled with a score which indicates the relevance between the article and its headline. As its author suggests, we take pairs from a subset of PART-II as the validation set and a subset of PART-III as the test set. Only pairs with score 3, 4 and 5, which means high relevance, are taken into account. We take pairs from PART-I that do not occur in the validation set as the training set.

Similar to what we do for preprocessing the People's Daily dataset, the word segmentation is carried out with jieba[vi] based on the dictionary of HowNet to alleviate the OOV problems.

C Details about Experiments Setting

In this section we describe the strategy we adopt to choose hyper-parameters and the optimal hyper-parameters used in the experiment.

[vi] https://pypi.python.org/pypi/jieba

Hyper-parameter	Baseline
Learning rate	30
Batch size	15
Embedding size	400
RNN hidden size	[1150, 1150, 400]
Word-level V-dropout	0.1
Embedding V-dropout	0.5
Hidden state V-dropout	0.2
Recurrent weight dropout	0.5
Context vector dropout	0.4

Table 7: Hyper-parameters used for AWD-LSTM and its variants

C.1 Language Modeling

The hyper-parameters are chosen according to the performance on the validation set. For medium (Tied) LSTM and its cHSM, tHSM variants, we search the dropout rate from $\{0.45, 0.5, 0.55, 0.6, 0.65, 0.7\}$. For large (Tied) LSTM and its cHSM, tHSM variants, we search the dropout rate from $\{0.6, 0.65, 0.7, 0.75, 0.8\}$. For AWD-LSTM and its variants, we follow most of the hyper-parameters described in (Merity et al., 2018) and only search the dropout rates (embedding V-dropout from $\{0.35, 0.4, 0.45, 0.5\}$, hidden state V-dropout from $\{0.2, 0.25, 0.3\}$, word level V-dropout from $\{0.05, 0.1, 0.15\}$ and context vector dropout from $\{0.4, 0.5\}$). For our SDLM and MoS, we fix all other hyper-parameters and only search the dropout rates of the last two layers respectively from {0.35, 0.4, 0.45} and {0.25, 0.3}. The initial learning rate for MoS on the top of AWD-LSTM is set to 20 to avoid diverging.

For (Tied) LSTM, we set the hidden unit and word embedding size to 650 (medium) / 1500 (large), batch size to 20, bptt to 35, dropout rate to 0.6 (medium) / 0.7 (large) and initial learning rate to 20. The optimal dropout rates for cHSM and tHSM are 0.55 (cHSM, medium), 0.5 (cHSM, medium, tied), 0.7 (cHSM, large), 0.65 (cHSM, large, tied), 0.55 (tHSM, medium) and 0.7 (tHSM, large). For AWD-LSTM and its variants, the hyper-parameters for the baseline are summarized in Table 7.

C.2 Headline Generation

The hyper-parameters are chosen according to the performance on the validation set. For RNN-context, we search the dropout rate from $\{0.1, 0.15, 0.2, 0.25, 0.3, 0.35\}$, the batch size from $\{32, 64\}$ and try SGD and Adam optimizers. For RNN-context-SDLM, we search the dropout rate from $\{0.15, 0.2, 0.25\}$, the batch size from $\{32, 64\}$ and try SGD and Adam optimizers.

For RNN-context, we use SGD optimizer with starting learning rate 0.001. We decay the learning rate by 0.5 every epoch after 7 epochs. The batch size is 32 and the dropout rate is 0.15. For RNN-context-SDLM, we use Adam optimizer with starting learning rate 0.001. The batch size is 64 and the dropout rate is 0.2.