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.