**Compare with the SOTA.** We carry out further experiments on Transformer and list the results in Table 1. We find that KerBS can also bring performance gains to Transformer models.

Table 1: Experimental results on Transformer.

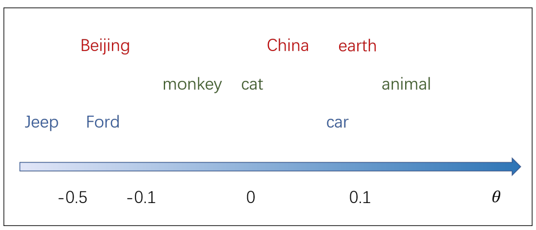
| Method | Transformer | Transformer+MoS | Transformer+KerBS |
|---|---|---|---|
| MT | 29.6 | 28.5 | 30.9 |
| Dialog | 10.61 | 9.81 | 10.90 |

**More analysis.** Following are some examples illustrating the effectiveness of KerBS in learning word properties.

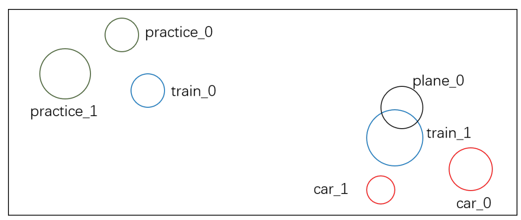
Table 2: Randomly selected words with different numbers of senses after training.

| Sense | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| word | Redwood<br>heal<br>structural<br>theoretical<br>rotate | particular<br>figure<br>during<br>known<br>size | open<br>order<br>amazing<br>sound<br>base | they<br>work<br>body<br>power<br>change |

Firstly, KerBS can learn the multisense property. From Table 2, we find that words with single meaning, including some proper nouns, are allocated with only one sense. But for words with more complex meanings, such as pronouns, more senses are necessary to represent them. (In our experiment, we restrict each word's sense number between 1 and 4, in order to keep the training stable.) In addition, we find that words with 4 senses have several distinct meanings. For instance, 'change' means transformation as well as small currency.




(a) Words with different $\theta$.




(b) A visualization of learned word embeddings. The radius of each sense embedding is decided by an order-preserving transformation of $\theta$.

Secondly, $\theta$ in KerBS is an indicator for words' semantic scopes. In figure (a) we compare the $\theta$ of 3 sets of nouns. For each set of them, we find words denoting bigger concepts (such as car, animal and earth) have larger $\theta$. In the pre-experiment, we build an oracle to generate word embeddings with different variances. It turns out that KerBS can accurately recover the order relation of variances between words. Please notice that in Figure (a), we use the largest $\theta$ of each sense as the $\theta$ of the word.

Figure (b) is a simple example of KerBS embedding. We find that the word 'train' has 2 senses (train_0 and train_1). While train_0 is close to the embeddings of 'practice', train_1 falls in the region of vehicles.

**Compare with MoS** Conditional text generation is more widely used than language models, but there is no experiments except LM in the paper of MoS. So we have to compare KerBS against MoS in other tasks. But the structure of MoS is kept the same, except the embedding size and the number of mixture of components. The speeds of MoS and KerBS with same number of senses are nearly equal in our experiments.

**Ablation study of dynamic allocation** We further perform a supplementary experiment on MT task. We find that with fixed sense allocation, KerBS still performs better than MoS(+0.55BLEU) and single sense KerBS(+0.20BLEU), but slightly worse than KerBS with dynamic allocation(-0.28BLEU). Moreover, the major advantage of dynamic allocation is that we can actually use a small total sense number to save computation, because only a few critical words need more than one senses. We will add a thorough analysis in the next version.

**Confusion in the Algorithm.** We are sorry for the confusion in Algorithm part and we will make it clear in the next version. (a) The computation of $P(y_{t-1} = S_i^j | x_{[0:t-1]})$ in Eq.7 is exactly the same as in Eq.5. (b) Figure 2 is a sketch map of kernel behaviours, where x-axis and y-axis is the coordinate frame of the embedding manifold, which takes a sense embedding $w_i^j$ as the origin. When the embedding of another sense is closer to the origin, their similarity gets higher. As a result, the kernel maximize at the origin. (c) The maximization in the loop is a MLE step, which maximizes the log-likelihood of generating the correct words. (d) S and T should be extracted from a corpus.