Softmax Bottleneck Makes Language Models Unable to Represent Multi-mode Word Distributions

Haw-Shiuan Chang and **Andrew McCallum**
CICS, University of Massachusetts, Amherst
140 Governors Dr., Amherst, MA, USA 01003
{hschang,mccallum}@cs.umass.edu

Abstract

Neural language models (LMs) such as GPT-2 estimate the probability distribution over the next word by a softmax over the vocabulary. The softmax layer produces the distribution based on the dot products of a single hidden state and the embeddings of words in the vocabulary. However, we discover that this single hidden state cannot produce all probability distributions regardless of the LM size or training data size because the single hidden state embedding cannot be close to the embeddings of all the possible next words simultaneously when there are other interfering word embeddings between them. In this work, we demonstrate the importance of this limitation both theoretically and practically. Our work not only deepens our understanding of *softmax bottleneck* and *mixture of softmax* (MoS) but also inspires us to propose *multi-facet softmax* (MFS) to address the limitations of MoS. Extensive empirical analyses confirm our findings and show that against MoS, the proposed MFS achieves two-fold improvements in the perplexity of GPT-2 and BERT.

"The greater the ambiguity, the greater the pleasure." — Milan Kundera

1 Introduction

Recently, researchers have found that transformer-based language models (LMs), such as GPT-2, can predict the next/masked word distribution better as their sizes grow (Radford et al., 2019; Brown et al., 2020; Kaplan et al., 2020). Compared to greedily outputting the most probable next word, sampling the next word from the predicted distribution allows a LM to generate more diverse and high-quality text sequences (Holtzman et al., 2020). By autoregressively sampling the next word according to its predicted probability, large LMs can be used to assist creative writing (Akoury et al., 2020), reduce the cost of building datasets (West et al., 2021; Liu et al., 2022), generate codes (Li et al., 2022), solve math problems (Cobbe et al., 2021), etc. As a result, one natural question arises: Do modern language modeling architectures still have restrictions in their ability to represent the appropriate distribution over next words or masked words?

In this paper, we discover that, when predicting the next word probabilities given an ambiguous context, GPT-2 is often incapable of assigning the highest probabilities to the appropriate non-synonym candidates. For example, given the input prompt *"After debating whether to bow to the* ***woman*** *or the* ***king*** *first, the jester decided on the [MASK]"*, we would expect the distribution over the *[MASK]* fillers to put high probabilities on *woman* or *king* or their synonyms. However, GPT-2 might incorrectly assign the second-highest probability to "*queen*" as in Figure 1.

In the final softmax layer of GPT-2, the log probabilities of the *woman* and *king* are computed based on the dot product between a single hidden state embedding and the global word embeddings of the *woman* and *king*, respectively. To have the highest but similar dot products for the two options, the transformer encoder in GPT-2 wants to output the hidden state that is close to the average of the *woman* embedding and the *king* embedding. However, the words *queen*, *king*, *woman*, and *man* tend to form a parallelogram in the embedding space (Mikolov et al., 2013; Ethayarajh et al., 2019; Wang et al., 2019)[1], which means the *man* and *queen* also have a similar average. Therefore, GPT-2 is forced to also output *man* or *queen* when it wants to output *woman* or *king*.

The problem not only happens to GPT-2 or the words whose embeddings form a parallelogram shape. Even though the hidden state embeddings of LMs are contextualized, the embedding of each

[1]Section 2.1 provides more background knowledge about the parallelogram shape and the softmax bottleneck.

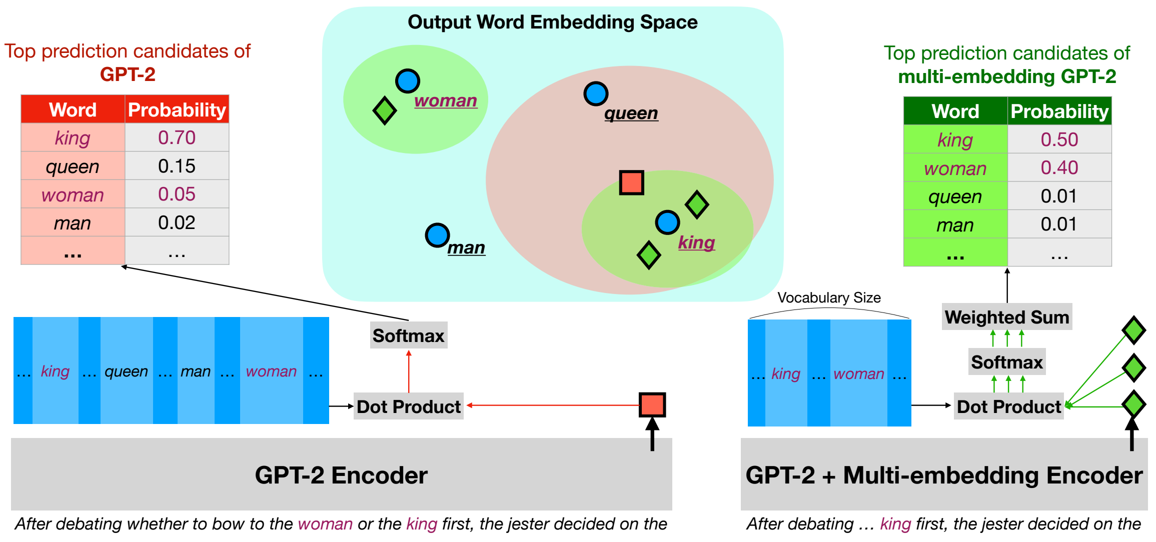

Top prediction candidates of GPT-2
Word
Probability
king
0.70
queen
0.15
woman
0.05
man
0.02
...
...
Output Word Embedding Space
woman
queen
man
king
Top prediction candidates of multi-embedding GPT-2
Word
Probability
king
0.50
woman
0.40
queen
0.01
man
0.01
...
...
Softmax
... king ... queen ... man ... woman ...
Dot Product
GPT-2 Encoder
After debating whether to bow to the woman or the king first, the jester decided on the
Vocabulary Size
Weighted Sum
Softmax
... king ... woman ...
Dot Product
GPT-2 + Multi-embedding Encoder
After debating ... king first, the jester decided on the

Figure 1: Comparison between the softmax layers using a single embedding and multiple embeddings when the next word should be either *woman* or *king*. In GPT-2 and multi-embedding GPT-2, the hidden states of the context are visualized by the single facet 🟥 and multiple facets ◆, respectively. The word embeddings are visualized using ●. GPT-2 cannot output *woman* and *king* as the top two words because ***queen*** and ***man*** are close to the midpoint of ***woman*** and ***king***. The improvement in this type of ambiguous context will be quantified in Section 5.

word in the softmax layer is global and static during the inference time. Globally dissimilar words could all become the suitable next word in a context while other interfering words might be between them, which makes the ideal next word embedding distribution have multiple modes and cannot be modeled by the single embedding representation.

In this work, we propose theorems showing that given any LM using the output softmax layer, when there are more than N word embeddings in a $N-1$ dimensional subspace/hyperplane (e.g., four embeddings in a two-dimensional plane), we can always find a set of possible next words (e.g., *woman* and *king*) such that there are some other interfering words between them (e.g., *man* or *queen*). That is, the multimodal next word distribution must exist if a few word embeddings are linearly dependent.

Recently, *mixture of softmax* (MoS) (Yang et al., 2018) regains attention as one of the few effective architecture modifications for transformer LM (Narang et al., 2021; Anonymous, 2021). In the meanwhile, Parthiban et al. (2021) show that the *softmax bottleneck* (Yang et al., 2018) theory is not sufficient to explain the improvement of MoS. As a remedy, our theorems not only provide geometrical intuitions of why and when the multiple embedding representation such as MoS would do better but also suggest that the *softmax bottleneck* might not be completely solved even if we adopt a very large hidden state size. For example, no matter how large the hidden state size is, as long as ***queen*** - ***king*** = ***woman*** - ***man*** in the embedding space, the LMs cannot output a pair of words in the longer diagonal of the parallelogram as the top two output words.

After better understanding why *mixture of softmax* (MoS) works well, we propose two enhancements over MoS. The first enhancement considers the hidden states of multiple positions and multiple transformer layers when determining the probability in each softmax; the second enhancement uses different contextualized embeddings to compute the probabilities of different subsets of words in the vocabulary.

The resulting method, *multi-facet softmax* (MFS), significantly outperforms the MoS and the softmax layer in GPT-2 on the perplexity for predicting the next word, especially in ambiguous context and non-English text in OpenWebText (Radford et al., 2019). Finally, we also show that MFS could improve the performance of GPT-2 on ProtoQA (Boratko et al., 2020), a commonsense question answering dataset where each question has multiple acceptable answers.

We summarize our theoretical, methodological, and empirical contributions as follows.

- **Theory**: We show the softmax layer using a single embedding is sometimes not able to output an appropriate rank of probabilities on a set of words with linearly dependent embeddings.

- **Method**: Addressing two weaknesses in MoS (Yang et al., 2018), we propose *multi-facet softmax* (MFS), a new alternative to the output softmax layer. MFS can replace the softmax in pre-trained LMs to better handle ambiguous contexts without re-training the LMs from scratch.
- **Analysis**: Our comprehensive empirical analyses discover and explain several phenomena, such as a) why using multiple embeddings is usually better than the single embedding with the non–linearity, b) why the improvement is larger in ambiguous contexts, less common languages, or GPT-2 compared to BERT, and c) why a LM often confuses with similar words.

2 Theoretical Limitations of the Single Embedding in the Softmax Layer

In this section, we first review the softmax layer of GPT-2 formally and explain why ***queen*** - ***king*** = ***woman*** - ***man*** still tends to hold in contextualized LMs. Next, we present our theoretical analyses, which generalize the *woman* and *king* example by showing that the candidate words in a low dimensional subspace would induce the impossibility of ranking some candidates on top of other candidates.

2.1 Background

The LMs typically use a softmax layer to predict $P_S(x|c_t)$, the probability of the next word x given the context at the tth position c_t:

$$P_S(x|c_t) = \frac{\exp(\boldsymbol{h}_{c_t}^T \boldsymbol{w}_x)}{\sum_{x'} \exp(\boldsymbol{h}_{c_t}^T \boldsymbol{w}_{x'})}, \tag{1}$$

where $\boldsymbol{h}_{c_t}$ is the tth hidden state in the context c, and $\boldsymbol{w}_x$ is the output word embedding for the word x (i.e., the linear weights that project the hidden state to the logit of the word x).[2] Yang et al. (2018) point out that the log probability distribution over all the words in the vocabulary V is $\log\left(P_S(x|c_t)\right)|_{x\in V} = \boldsymbol{h}_{c_t}^T \boldsymbol{w}_x - \log\left(\sum_{x'} \exp(\boldsymbol{h}_{c_t}^T \boldsymbol{w}_{x'})\right)|_{x\in V}$. The distribution is a linear projection from the hidden state $\boldsymbol{h}_{c_t}$ with dimension D, so the degree of freedom in the distribution is only D (i.e., there cannot be more than D linearly independent log distributions). We call this restriction *softmax bottleneck* theory.

[2]Notice that some LMs such as BERT add a bias term for each word before the softmax layer. For simplicity, our theoretical analyses focus on the LMs without the bias term such as GPT-2.

During training, the ideal output word embedding $\boldsymbol{w}_x$ should be close to the hidden states of the contexts $\boldsymbol{h}_{c_t}$ that co-occur with the word x while far away from the other hidden states. This objective is similar to the objective function of Word2Vec (Mikolov et al., 2013) except that the context embeddings are contextualized (Kong et al., 2020; Li et al., 2020).

If a context c_t has a higher chance to co-occur with *queen* compared to *king*, the context also has a higher chance to co-occur with *woman* compared to *man* to a similar degree. This is the main reason that makes ***queen*** - ***king*** = ***woman*** - ***man*** in the Word2Vec space (Ethayarajh et al., 2019). Therefore, the same linear relations tend to hold in the output word embedding space of GPT-2 as well (Wang et al., 2019).

2.2 Structural Weakness Theorems from Linear Dependency

In addition to words satisfying the analogy relations, the following theorems imply that any linear dependency among the words causes the difficulties of LM in ranking the words in an arbitrary order according to their logits (i.e., dot products between the hidden state and the word embedding). For example, ***woman*** + ***king*** = ***queen*** + ***man*** makes a LM unable to assign the highest positive logits to *woman* and *king* and output them as the top two words in Figure 1.

Theorem 1. *If the nonzero output embeddings of N words in a set W are linearly dependent and on one side of a hyperplane through the origin, the single embedding representation cannot produce positive logits for a subset of the words in W that are higher than all the logits of the other words in W.*

Here, we provide an intuitive justification: if N embeddings are in a subspace whose dimension is smaller than $N-1$ (e.g., three points in a one-dimensional line), the N embeddings are going to be linearly dependent and some set of words cannot have the top dot products due to the limited degree of freedom in the subspace. In Appendix D, we formally prove the theorem by identifying the sets of words that cannot be ranked top by the single embedding representation.

In practice, linear dependency holds approximately instead of exactly. For example, ***woman*** = ***queen*** + ***man*** - ***king*** + ε. In this practical condition, the following theorem states that the logits of the

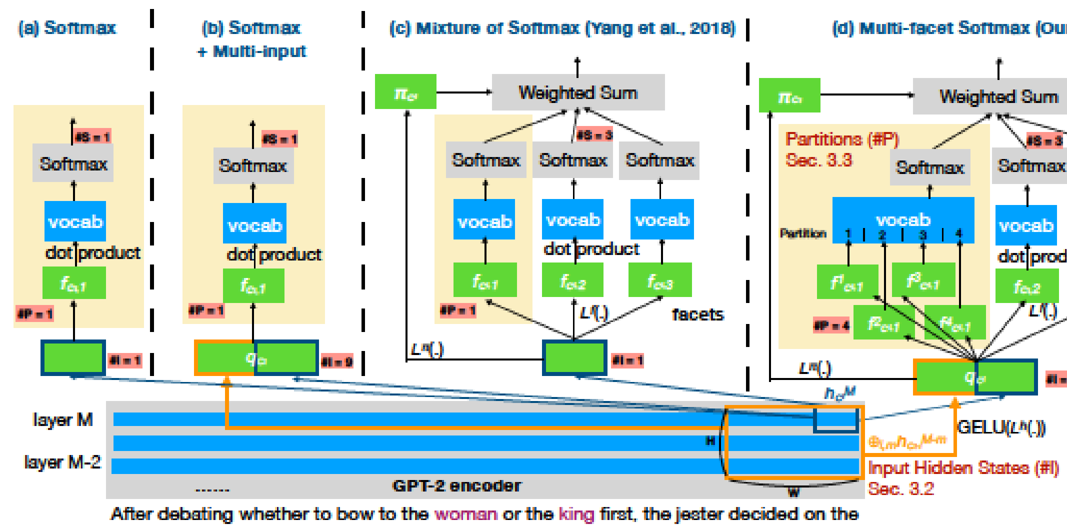

(a) Softmax
(b) Softmax + Multi-input
(c) Mixture of Softmax (Yang et al., 2018)
(d) Multi-facet Softmax (Ours)
Softmax
vocab
dot product
#S = 1
#P = 1
#I = 1
#I = 9
#S = 3
#P = 4
#S = 3
#I = 9
Weighted Sum
Partitions (#P) Sec. 3.3
Partition
facets
layer M
layer M-2
GPT-2 encoder
GELU(L^h(.))
Input Hidden States (#I) Sec. 3.2
After debating whether to bow to the woman or the king first, the jester decided on the

Figure 2: Comparison between different architectures. The #S, #I, and #P are the number of softmaxes, input hidden states, and partitions, respectively. The green boxes refer to embeddings/vectors. The vocab means the embeddings of all words in the vocabulary. $\oplus$ refers to concatenation. L^h, L^f, and L^π are linear projection layers.

interfering words (i.e., *man* and *queen*) cannot be much smaller than the logits of the candidate words (i.e., *woman* and *king*).

Theorem 2. *Let the output word embeddings in the set $W = \{\boldsymbol{w}_i \neq \boldsymbol{0} | i = 1...N\}$ satisfy $\boldsymbol{w}_1 = a_2\boldsymbol{w}_2 + ... + a_N\boldsymbol{w}_N + \boldsymbol{\varepsilon}$, where the constant $a_2, ..., a_N$ are neither all zero nor all negative and $||\boldsymbol{\varepsilon}|| < \epsilon$. Then, there must be a non-trivial partition $P = \{G, S\}$ of W such that there is no hidden state $||\boldsymbol{h}|| \leq r$ and a threshold $\tau \geq r\epsilon$ that make $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T\boldsymbol{w}_g \geq (1+\delta)\tau$ and $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T\boldsymbol{w}_s < \tau$, where $\delta = \frac{2}{1+\sum_{i=2...N}|a_i|}$.*

In the king-woman example, $(1+\delta) = (1+\frac{2}{4}) = 1.5$. Assuming $||\boldsymbol{\varepsilon}|| < \epsilon = 0.01$ and $||\boldsymbol{h}|| \leq r = 20$, we can get $\boldsymbol{h}^T\boldsymbol{\varepsilon} \leq 0.01 \times 20 = 0.2$. Then, we cannot find a hidden state $\boldsymbol{h}$ such that $\boldsymbol{h}^T\boldsymbol{w}_{king} \geq 1.5 \times 0.01 \times 20 = 0.3$ and $\boldsymbol{h}^T\boldsymbol{w}_{woman} \geq 0.3$ but $\boldsymbol{h}^T\boldsymbol{w}_{queen} < 0.2$ and $\boldsymbol{h}^T\boldsymbol{w}_{man} < 0.2$ because $\boldsymbol{h}^T\boldsymbol{w}_{king} + \boldsymbol{h}^T\boldsymbol{w}_{woman} = \boldsymbol{h}^T\boldsymbol{w}_{queen} + \boldsymbol{h}^T\boldsymbol{w}_{man} + \boldsymbol{h}^T\boldsymbol{\varepsilon}$. The formal proof of Theorem 2 can be found in Appendix D and Appendix B.1 estimates ϵ in several language models.

Even though, theoretically speaking, outputting *woman* and *king* as the top two words is possible due to the appearance of $\boldsymbol{\varepsilon}$, LMs may not successfully learn to output the optimal $\boldsymbol{h}$ and the optimal hidden state for these four words could lead to the wrong probabilities of the other words. Consequently, LMs sometimes still rank *queen* or *man* higher than *woman* or *king* in practice.

3 Multi-facet Softmax

Using multiple embeddings is a natural solution for modeling a multimodal distribution (Bishop, 1994). For instance, we can use three embeddings to capture the high probability on the *woman* and *king* but low probability on the *man* and *queen* in Figure 1.

Inspired by our geometric analysis on the limitation of the single embedding, we improve the state-of-the-art multiple embedding solution, *mixture of softmax* (MoS) (Yang et al., 2018) by two enhancements: multiple input hidden states and multiple partitions on the vocabulary.

3.1 Mixture of Softmax

Yang et al. (2018) propose *mixture of softmax* (MoS) to allow a LSTM-based (Hochreiter and Schmidhuber, 1997) LM to produce more linearly independent log probability distributions of the output words given different contexts. As in Figure 2 (c), the MoS first uses multiple linear layers L^f_k to project a hidden state $\boldsymbol{h}_{c_t}$ into multiple facet embeddings $\boldsymbol{f}_{c_t,k} = L^f_k(\boldsymbol{h}_{c_t})$.[3] The multiple facets $\boldsymbol{f}_{c_t,k}$ and softmaxes would lead to multiple probability distributions, and output probability is the weighted average of the distributions:

$$P_{MoS}(x|c_t) = \sum_{k=1}^{K} \pi_{c_t,k} \frac{\exp(\boldsymbol{f}^T_{c_t,k}\boldsymbol{w}_x)}{\sum_{x'}\exp(\boldsymbol{f}^T_{c_t,k}\boldsymbol{w}_{x'})}. \quad (2)$$

[3]We remove the tanh layer in the original MoS to improve its performance on GPT-2. See Appendix G.1 for details.

The prior weights $\pi_{c_t,k} = \frac{\exp(L^{\pi}_k(\boldsymbol{h}_{c_t}))}{\sum_{k'} \exp(L^{\pi}_{k'}(\boldsymbol{h}_{c_t}))}$, where L^{π}_k is another linear projection for dynamically generating the weights and the projection goes through a softmax to ensure $\sum_{k=1}^{K} \pi_{c_t,k} = 1$.

3.2 Multiple Input Hidden States

To model the multimodal distribution, the facets (i.e., the embeddings for different softmaxes) should be able to move freely. For example, in Figure 1, we have three facets but only have two modes, so the two embeddings are very close to the word *king*. However, when we want to output three dissimilar top words such as the *king*, *woman*, and *knight*, one of the facets should be moved to be near to the embedding of the *knight*.

Therefore, we want our solution to satisfy two properties: a) the linear transformation matrix in L^f_k should have a full rank to avoid limiting the degree of freedom in each facet, and b) the relative location of the facets should be context-dependent. MoS cannot satisfy both properties. If the first one is satisfied, the input hidden state is uniquely determined by a facet (e.g., $\boldsymbol{h}_{c_t} = (L^f_1)^{-1}(\boldsymbol{f}_{c_t,1})$). Then, there exists a global transformation between two facets (e.g., $\boldsymbol{f}_{c_t,2} = L^f_2\left((L^f_1)^{-1}(\boldsymbol{f}_{c_t,1})\right)$), which violates the second property. That is, assuming LM can move every facet freely (i.e., the facet's degree of freedom is the same as the dimension of the hidden state), LM cannot make the first two facets close to *woman* and *king* in one context but make the two facets close to *woman* and *knight* in another context. In other words, since the facet embeddings are the projection of a single hidden state, the total degree of freedom in all facet embeddings cannot exceed the dimension of the hidden state.

Our solution to this issue is using more input hidden states to construct the facets. As the orange box in Figure 2, we first concatenate a $W \times H$ block of input hidden states into $\oplus_{i=0...W-1,m=0...H-1}\boldsymbol{h}^{M-m}_{c_{t-i}}$, where $M-m$ is the transformer layer index and $t-i$ is the index of the ith to the last word in the context. The $W \times H$ is fixed as 3×3 in this paper. We make its dimension the same as the original hidden state $\boldsymbol{h}^M_{c_t}$ using a linear layer L^h plus a GELU activation function (Hendrycks and Gimpel, 2016). Then, we concatenate it with the original hidden state to form a new input hidden state

$$\boldsymbol{q}_{c_t} = \boldsymbol{h}^M_{c_t} \oplus GELU\left(L^h(\oplus_{i,m}\boldsymbol{h}^{M-m}_{c_{t-i}})\right). \quad (3)$$

The new input hidden state is passed through the linear transformation L^f_k to compute the facets $\boldsymbol{f}_{c_t,k} = L^f_k(\boldsymbol{q}_{c_t})$ and our prior weights $\pi_{c_t,k} = \frac{exp(L^{\pi}_k(\boldsymbol{q}_{c_t}))}{\sum_{k'} exp(L^{\pi}_{k'}(\boldsymbol{q}_{c_t}))}$. Since the dimension of $\boldsymbol{q}_{c_t}$ is larger than the dimension of $\boldsymbol{f}_{c_t,k}$, the inverse function $(L^f_k)^{-1}$ no longer exists.

3.3 Multiple Partitions

The next word distribution could have many modes. However, using many softmaxes significantly increases our computational burden because we need to compute the dot product between each facet and all the word embeddings in our vocabulary.

Inspired by our analysis, we propose to split all the words in the vocabulary into multiple partitions[4] and use different facets for different partitions. For example, if we can put any word from {*queen*, *man*, *woman*, *king*} into one partition and the rest of the words into another partition, we no longer have ***queen*** - ***king*** = ***woman*** - ***man*** in either of the partitions. In this method, each word only belongs to one partition, so we only need to compute one dot product for each word. Thus, the extra computational cost only comes from the extra linear projections for preparing the facets.

In many contexts c_t, the distribution of the next word has only a single mode and the global similarity between words may be useful. Using the multiple partitions alone might lose the similarity information between words in different partitions. Therefore, we propose to only replace the first softmax layer in MoS with the multiple partition method to learn the global similarity of words in different partitions using the other softmaxes. The architecture is illustrated in Figure 2 (d). Formally, we compute the probability using

$$P_{MP}(x|c_t) = \pi_{c_t,1}\frac{\exp((\boldsymbol{f}^{j_x}_{c_t,1})^T\boldsymbol{w}_x)}{\sum_{x'}\exp((\boldsymbol{f}^{j_{x'}}_{c_t,1})^T\boldsymbol{w}_{x'})} + \sum_{k=2}^{K}\pi_{c_t,k}\frac{\exp(\boldsymbol{f}^T_{c_t,k}\boldsymbol{w}_x)}{\sum_{x'}\exp(\boldsymbol{f}^T_{c_t,k}\boldsymbol{w}_{x'})}, \quad (4)$$

where j_x is the partition index that the word x belongs to and $\boldsymbol{f}^{j_x}_{c_t,1}$ is the facet for the j_xth partition.

[4]In this work, we simply put the $J \times n + j$th word into jth partition (e.g., when the number of partitions $J = 4$, the first partition includes the words with indexes $0, 4, 8, ...$). This simple global partitioning method reduces the chance of putting all the interfering words and candidates in the same partition, while minimizing the extra computational cost in our PyTorch implementation because PyTorch supports strided index slicing without copying the variable.

Multi-facet softmax (MFS) is equipped with multiple input hidden states and multiple partitions.

4 Language Modeling Experiments

We evaluate different LM architectures by comparing their capability of predicting the next word in Wikipedia 2021 and a subset of OpenWebText (Radford et al., 2019). In addition to perplexity, we also compare their mean reciprocal ranks (MRR) in Appendix C.1. The size of the training, validation, and testing set are 96%, 2%, and 2% of the whole corpus, respectively. After loading the pre-trained GPT-2 models, we train the *GPT-2 Small* for 1 epoch and *GPT-2 Medium* for 0.4 epochs. We also test our methods on BERT in Appendix B.2. Please see Appendix G for more details of our experiment setup.

4.1 Baselines

We set different numbers of softmaxes, input hidden states, and partitions in our MFS framework to construct our baselines. The configuration of different baselines could be seen in Table 1.

Softmax (GPT-2): Using a single softmax, input hidden state, and partition as in Figure 2 (a) and Equation 1. The baseline is the same as the original GPT-2 except that we add one more linear layer that converts the hidden state $\boldsymbol{h}_{c_t}^M$ to the facet embedding $\boldsymbol{f}_{c_t,1}$ as in other methods.

SigSoftmax (Kanai et al., 2018): The same as *Softmax* except when predicting the next word, Kanai et al. (2018) add some non-linearity into the softmax layer by multiplying the exponent and sigmoid of the logits.

Softmax + Multi-input: Letting *Softmax* access multiple input hidden states as in Figure 2 (b) and Equation 3. The method is similar to Tenney et al. (2019); Fan et al. (2020), and Tay et al. (2021).

MoS (Yang et al., 2018): *MoS (3)* is the *mixture of softmax* with 3 facets/softmaxes, whose probability comes from Equation 2. We also run the MoS with 4 softmaxes in *GPT-2 Small* and call the model *MoS (4)*.

DOC (Takase et al., 2018): Similar to our enhancement using multiple input hidden states, *direct output connection* (DOC) makes each of their facets coming from a different input hidden state.

Other configurations include **Softmax + Multi-partition**, which adds four partitions into the softmax, **MFS w/o Multi-partition**, which uses only one partition in *MFS* and could also be viewed as *MoS + Multi-input*, and **MFS w/o Multi-input**, which uses only one input hidden state to generate all facets.

4.2 Results

Table 1 shows that applying **MFS** to *GPT-2 Small* achieves more than 15% of the perplexity improvement between *GPT-2 Small* and *GPT-2 Medium*, while only increasing 5% of their size differences. Except for **Softmax + Multi-partition**, adding multiple input hidden states or partitions in different configurations significantly boost the performances. In Appendix B.3, we further show that the improvement of **MFS** over **Softmax** could even become 3-5 times larger in the top 5-10% of the most ambiguous contexts compared to the rest of the contexts, which suggests that some improvements indeed come from successfully modeling multimodal distribution.

MFS usually doubles the perplexity improvements between **MoS (3)** and **Softmax** but the running time of **MFS** remains similar to **MoS (3)** because **MFS** only needs a few more linear layers, which is more efficient than adding one more softmax as in **MoS (4)**. **DOC** is worse than **MoS (3)**. This may be due to a starvation problem: the facet from the last hidden state $\boldsymbol{h}_{c_t}^M$ has the prior probability close to 1 and receives most of the gradients. Finally, compared with **Softmax**, the mixed results in **SigSoftmax** suggest that adding non-linearity into the softmax layer without modeling the multimodal distribution might not always improve the models (Parthiban et al., 2021).

OpenWebText is mostly composed of English text, but some non-English text in the corpus allows us to compare the capability of different models in a multi-lingual setting. Table 2 shows that multiple embeddings improve the perplexity of the non-English text more than the perplexity of the English text. We hypothesize that the distribution of the next non-English word is more likely to be multi-mode because GPT-2 learns the global token embeddings mostly in the English contexts, which could make the embeddings of similar tokens in non-English contexts far away.

In Table 3, we present three contexts from the validation set of different datasets and compare the top three predictions of **MFS** and **Softmax** on *GPT-2 Small*. In OpenWebText and Wikipedia 2021, we can see that **Softmax** misses the correct answer in its top three predictions.

Models ↓	Configuration #S	#I	#P	GPT-2 Small Size	Time	OWT	Wiki	GPT-2 Medium Size	Time	OWT	Wiki
Softmax (GPT-2)	1	1	1	163.6M	84ms	18.72	24.06	407.3M	212ms	15.89	20.34
SigSoftmax (Kanai et al., 2018)	1	1	1	163.6M	91ms	18.63	24.06	407.3M	221ms	16.07	20.65
Softmax + Multi-input	1	9	1	169.5M	87ms	18.50	23.89	417.8M	219ms	15.76	20.29
Softmax + Multi-partition	1	1	4	165.4M	88ms	18.77	24.08	410.5M	218ms	15.89	20.30
MoS (Yang et al., 2018) (4)	4	1	1	165.4M	152ms	18.61	23.77	410.5M	299ms	15.75	20.08
MoS (Yang et al., 2018) (3)	3	1	1	164.8M	130ms	18.63	23.81	409.4M	270ms	15.79	20.11
DOC (Takase et al., 2018)	3	3	1	164.8M	130ms	18.69	24.02	409.4M	270ms	15.88	20.34
MFS w/o Multi-partition	3	9	1	171.9M	133ms	18.37	23.56	422.0M	276ms	15.65	20.06
MFS w/o Multi-input	3	1	4	166.6M	134ms	18.60	23.72	412.6M	275ms	15.71	20.08
MFS (Ours)	3	9	4	175.4M	138ms	**18.29**	**23.45**	428.3M	283ms	**15.64**	**20.02**

Table 1: Perplexity comparison between MFS (Ours) and baselines. #S, #I, #P are the number of softmaxes (i.e., K), input hidden states, and partitions, respectively. The top four baselines use a single softmax. OWT and Wiki are the test set perplexity of OpenWebText and Wikipedia 2021, respectively. The standard errors of all models are smaller than 0.02 perplexity. We also compare the number of parameters and the inference time on one batch.

Ratio in Corpus →	Non-English 14%	English 86%
Softmax	13.50 (0.0%)	19.23 (0.0%)
MoS (Yang et al., 2018) (3)	13.19 (2.3%)	19.16 (0.4%)
MFS w/o Multi-partition	12.98 (3.8%)	18.91 (1.7%)
MFS (Ours)	**12.83 (5.0%)**	**18.83 (2.1%)**

Table 2: Perplexity of the *GPT-2 Small* in OpenWebText. The percentages of the perplexity reduction compared to Softmax are presented in the parentheses.

5 Evaluation on Ambiguous Templates

We synthesize a dataset using templates (Ribeiro et al., 2020) to verify whether the softmax layer in the original GPT-2 really has difficulty in learning to output the bimodal distribution in Figure 1 and whether the multiple embedding methods could overcome the problem. First, we collect the four words with semantic analogy relations in Google analogy dataset (Mikolov et al., 2013). Next, we insert two out of the four words into our manually written templates to form the contexts and the templates we used could be found in Appendix G.3. For example, given the context "*I went to Paris and Germany before, and I love one of the places more, which is*", the GPT-2 learns to predict either *Paris* or *Germany*.

The two words can be either the diagonal words (e.g., *king* and *woman*) or the edge word (e.g., *king* and *queen*) in the parallelogram. Finally, we create a dataset with 122k training contexts, 250k validation contexts, and 122k testing contexts, where the word pairs in the testing set are unseen in the training set to see whether the model could learn to output the bimodal distribution in a general way.[5]

We load the models pre-trained on OpenWebText and continue fine-tuning the models on the last word of each sentence for 10 epochs. We report the testing performances of the best model selected by the validation loss. Since the sets of the word pairs in the training and testing set are disjoint, updating the output word embedding would make GPT-2 solve the task by memorizing/overfitting the training set quickly and lead to much worse testing performances. Thus, we freeze the output word embedding during the training.

We visualize the predictions of the Paris-Germany example in the last column of Table 3. We can see two of the softmaxes are close to *Paris* and the remaining one is close to *German*, while **Softmax** overestimates the probability of *Paris* and ranks *France* higher than the *German*. The result verifies that the correct probability distribution of the words in some ambiguous context is hard to learn using **Softmax**.

Quantitatively, Table 4 indicates that when the possible next words are the diagonal words, the **Softmax** model performs much worse compared to other multiple embedding alternatives. In the edge word dataset, the multiple embedding solutions are still better but have a much smaller gap. **MFS w/o Multi-partition** slightly improves **MoS**. We hypothesize the reason is that multiple input hidden states could help the facets to be moved more freely. Finally, multiple partitions seem to cause slight overfitting in this bimodal distribution prediction task.

[5]The setting is realistic because any related words could become the next word in some ambiguous contexts and all the words are related in a certain way (Sigman and Cecchi, 2002). We cannot expect the training corpora to contain the ambiguous contexts with so many possible next words.

Corpus →	OpenWebText	Wikipedia 2021	Analogy in Templates (Section 5)
Input Context	... The Elastic Endpoint Security and Elastic SIEM solutions mentioned in this post are now referred to as **Elastic**	... law and chance working together cannot generate CSI, either. Moreover, he claims that **CSI**	I went to Paris and Germany before, and I love one of the places more, which is **Germany**
Softmax (GPT-2)	the 0.087, E 0.043, End 0.039	the 0.174, this 0.054, if 0.038	Paris 0.893, France 0.045, **Germany** 0.033
MFS (Ours)	**Elastic** 0.220, the 0.089, EC 0.033	CSI 0.186, the 0.140, there 0.033	Paris 0.544, **Germany** 0.389, France 0.064
MFS Softmax 1	end 0.051, the 0.043, security 0.023	the 0.191, law 0.127, if 0.053	Paris 0.979, France 0.013, **Germany** 0.007
MFS Softmax 2	**Elastic** 0.652, EC 0.080, ES 0.046	the 0.191, there 0.049, this 0.047	Paris 1.000 Berlin 0.000 ##Paris 0.000
MFS Softmax 3	the 0.193, E 0.040, a 0.014	**CSI** 0.677, law 0.029, laws 0.019	**Germany** 0.852, France 0.139, China 0.004

Table 3: Prediction visualization using a context in each dataset. We show the top three words with the highest prediction probabilities of each method. In the last three rows, we visualize the outputs of the softmax grey boxes in Figure 2 (d), which model different modes of the next word distribution. The prediction target is boldfaced in the context and the predictions. ## indicates there is no space before the word.

Analogy Relation Types →	Diagonal (e.g., *king* or *woman*)					Edge (e.g., *king* or *queen*)				
Models ↓	valid	capital-common	capital-world	city-in-state	family	valid	capital-common	capital-world	city-in-state	family
Softmax (GPT-2)	2.30	3.30	2.00	2.25	2.95	2.11	2.42	1.91	2.26	2.38
MoS (Yang et al., 2018) (3)	1.75	2.18	1.60	1.85	2.82	1.87	2.26	**1.70**	2.04	2.27
MFS w/o Multi-partition	**1.72**	**2.13**	**1.59**	**1.82**	**2.52**	**1.84**	**2.23**	1.72	**1.96**	**2.16**
MFS (Ours)	1.74	2.15	**1.59**	**1.82**	2.63	1.92	2.28	1.78	2.00	2.24

Table 4: Perplexity comparison of different *GPT-2 Small* models on the words with different types of analogy relations. The validation set (valid) includes all four types of relations.

	Perplexity on Scraped	Max Answers				Max Incorrect		
Models ↓	Development Set	Top 1	Top 3	Top 5	Top 10	Top 1	Top 3	Top 5
Softmax (GPT-2)	1.5432 ± 0.0003	34.1 ± 0.8	35.2 ± 0.5	37.8 ± 0.4	45.0 ± 0.5	18.3 ± 0.4	30.7 ± 0.5	38.5 ± 0.6
MoS (Yang et al., 2018) (3)	1.5407 ± 0.0004	33.9 ± 0.8	36.0 ± 0.6	37.7 ± 0.6	44.9 ± 0.4	18.3 ± 0.4	31.7 ± 0.6	38.2 ± 0.6
MFS w/o Multi-partition	1.5411 ± 0.0003	**34.3 ± 0.7**	**36.7 ± 0.7**	38.1 ± 0.5	45.2 ± 0.4	19.4 ± 0.4	32.0 ± 0.5	38.6 ± 0.3
MFS (Ours)	**1.5402 ± 0.0005**	34.1 ± 0.6	**36.7 ± 0.5**	**38.6 ± 0.4**	**45.4 ± 0.5**	**19.7 ± 0.4**	**32.1 ± 0.4**	**39.7 ± 0.4**

Table 5: ProtoQA performances. All the numbers except perplexity are the percentages of the predictions that match the ground truth exactly on the crowdsourced development set. Max answers top k implies only evaluating the top k answers. Max incorrect top k indicates only evaluating the top answers that contain k errors. The best average performances are highlighted and the standard errors are reported as the confidence interval.

6 Answering Ambiguous Questions

ProtoQA (Boratko et al., 2020) is a question-answering dataset built for evaluating the commonsense reasoning ability of language models. Each question in ProtoQA is ambiguous and leads to a distribution of possible answers. For instance, the answer to "*Name something that people usually do before they leave for work?*" is "*Shower 0.43, Breakfast 0.30, ...*". The paper discovers that by reformulating the question-answering task as a context (e.g., "*One thing people usually do before they leave for work is ...*"), GPT-2 could generate the possible answers by sampling the next words from its word prediction distribution.

The dataset gives us a chance to directly compare the quality of the distributions generated by different LMs in Table 5. After pretraining *GPT-2 Medium* on the OpenWebText, we fine-tune them using the training data in ProtoQA for 2 epochs. We repeat the fine-tuning 5 times and compare their average perplexity in our validation set. Next, we generate 150 sentences starting from each context and compare the generated answers with the ground truth distribution. For each fine-tuned model, we repeat the generation evaluation 3 times and report the average accuracy of the resulting 15 trials.

We can see that the multiple softmaxes, input hidden states, and partitions usually improve the quality of prediction distribution, and the proposed **MFS**, which combines all modifications, achieves the best performances.

7 Related Work

Yang et al. (2018) propose the concept of *softmax bottleneck*, which points out that the dot product in the softmax layer restricts the representation power of outputting arbitrary conditional probabilities. It also proposes *MoS* to break the softmax bottleneck in an RNN-based LM. Kanai et al. (2018) and Ganea et al. (2019) add nonlinearities into the softmax layer to break the bottleneck more efficiently, but the approaches gain less improvement compared to *MoS*.

A limitation of the aforementioned previous work is that they do not tell us which kinds of sentences would be affected by the bottleneck more and whether the order of the top few next words would be affected, which are the main research questions of our work. Contrary to the previous belief that a large hidden state dimension would eliminate the softmax bottleneck, our theorems suggest that some words in a low dimensional subspace could still make the single embedding in the softmax layer become a bottleneck of arbitrarily ranking the output words. Furthermore, our geometric analyses provide an intuitive explanation about why breaking the bottleneck using multiple embeddings leads to better performances compared to only adding the non-linearity.

Demeter et al. (2020) also analyze the structural weakness of the softmax layer from a geometric perspective. They discover that the words with high prior frequencies could stop the LMs from assigning the high probabilities to rare words, which can be viewed as a special case of our theory (See Appendix E). For instance, our work shows that the softmax layer could still prevent the LMs from outputting some top words even if all the possible next words have the same prior frequency.

Our theory is deeply connected to the mathematical work that counts the number of possible rankings of points in an embedding space (Cover, 1967; Good and Tideman, 1977). Compared to the studies, our work focuses more on analyzing the multimodal distribution in the word embedding space and its implication to language models.

An alternative to model the multimodal distribution is to use multiple embeddings to represent each output word (Athiwaratkun and Wilson, 2017; Miao et al., 2019). Compared to MoS or our approach that use multiple embeddings to represent each hidden state of the context, their method requires many extra parameters to store different senses of each output word. Another type of related model (Shazeer et al., 2017; Fedus et al., 2021) dynamically routes the signals to different experts (i.e., feed-forward networks) and Zhang et al. (2022); Mittal et al. (2022) use multiple embeddings in the attention layers. The methods are similar to MoS and our approach, but they add the multiple embeddings inside each layer of the transformer encoder while the proposed MFS is an alternative to the output softmax layer.

8 Conclusion

When the ideal distribution in the output word embedding space has multiple modes, GPT-2 cannot learn to correctly rank the words in all the modes as the top next words. This shows that the single embedding in the softmax layer, which is used nearly universally by current LMs, constitutes a performance upper bound of predicting the next/masked word. To address the systematic failure caused by these structural weaknesses, we propose *multi-facet softmax* (MFS). In our experiments, we confirm that the MFS significantly outperforms the standard softmax layer and alleviates the *softmax bottleneck* in the transformer-based LMs such as GPT-2 better than *mixture of softmax* (MoS).

9 Acknowledgement

We thank Michael Boratko, Jay Yoon Lee, Sabrina J. Mielke, Steve Cheng-Xian Li, and the anonymous reviewers for their constructive feedback. This work was supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction, in part by the IBM Research AI through the AI Horizons Network, in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative, in part by the National Science Foundation (NSF) grant numbers DMR-1534431, IIS-1763618, and IIS-1955567, and in part by the Office of Naval Research (ONR) via Contract No. N660011924032 under Subaward No. 123875727 from the University of Southern California. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

10 Ethical and Broader Impact

This work studies a general limitation of LMs and proposes solutions. The proposed theory can help us to understand that some types of hallucinations, mistakes, or biases of LMs could come from *softmax bottleneck* and their incapability of modeling the correct distribution. For example, there are 60% of male characters and 40% of female characters in our training corpus. The language generation model might be forced to assign more than 60%

probability to male characters as being much more likely to output *king* than *woman* in Figure 1.

Recently, Narang et al. (2021); Anonymous (2021) show that MoS is one of the few architecture modifications of transformer-based LM that can provide consistent improvements in downstream applications. Our work provides a fundamental reason why the multiple embedding representation is better, which could inspire more future studies that propose a better multiple-embedding architecture to improve LMs (e.g., multi-lingual BERT) or downstream applications. As examples, we list several possible future directions in Appendix H.

Finally, a better LM could lead to both positive and negative societal impacts, but they are not the focus of this paper. Generally speaking, this paper deepens our understanding of the weaknesses of modern LMs and we believe the knowledge can help us to design a better LM that increases the positive impacts and reduces the negative impacts in the future.

References

Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6470–6484, Online. Association for Computational Linguistics.

Anonymous. 2021. Scaling laws vs model architectures: How does inductive bias influence scaling? an extensive empirical study on language tasks. In *ACL ARR Blind Submission*.

Ben Athiwaratkun and Andrew Wilson. 2017. Multimodal word distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1645–1656, Vancouver, Canada. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Christopher M Bishop. 1994. Mixture density networks.

Michael Boratko, Xiang Li, Tim O'Gorman, Rajarshi Das, Dan Le, and Andrew McCallum. 2020. ProtoQA: A question answering dataset for prototypical common-sense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1122–1136, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. 2021. Isotropy in the contextual embedding space: Clusters and manifolds. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hyung Won Chung, Thibault Févry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking embedding coupling in pre-trained language models. In *ICLR*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Thomas M Cover. 1967. The number of linearly inducible orderings of points in d-space. *SIAM Journal on Applied Mathematics*, 15(2):434–439.

David Demeter, Gregory Kimmel, and Doug Downey. 2020. Stolen probability: A structural weakness of neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2191–2197, Online. Association for Computational Linguistics.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy. Association for Computational Linguistics.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. 2020. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Octavian Ganea, Sylvain Gelly, Gary Bécigneul, and Aliaksei Severyn. 2019. Breaking the softmax bottleneck via learnable monotonic pointwise non-linearities. In *Proceedings of the 36th International*

Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 2073–2082. PMLR.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019a. Representation degeneration problem in training natural language generation models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yingbo Gao, Christian Herold, Weiyue Wang, and Hermann Ney. 2019b. Exploring kernel functions in the softmax layer for contextual word classification. In *Proceedings of the 16th International Conference on Spoken Language Translation, Hong Kong*.

IJ Good and TN Tideman. 1977. Stirling numbers and a geometric, structure from voting theory. *Journal of Combinatorial Theory, Series A*, 23(1):34–45.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Sekitoshi Kanai, Yasuhiro Fujiwara, Yuki Yamanaka, and Shuichi Adachi. 2018. Sigsoftmax: Reanalysis of the softmax bottleneck. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 284–294.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Lingpeng Kong, Cyprien de Masson d'Autume, Lei Yu, Wang Ling, Zihang Dai, and Dani Yogatama. 2020. A mutual information maximization perspective of language representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online. Association for Computational Linguistics.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*.

Alisa Liu, Swabha Swayamdipta, Noah A Smith, and Yejin Choi. 2022. Wanli: Worker and ai collaboration for natural language inference dataset creation. *arXiv preprint arXiv:2201.05955*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ning Miao, Hao Zhou, Chengqi Zhao, Wenxian Shi, and Lei Li. 2019. Kernelized bayesian softmax for text generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12487–12497.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

Sarthak Mittal, Sharath Chandra Raparthy, Irina Rish, Yoshua Bengio, and Guillaume Lajoie. 2022. Compositional attention: Disentangling search and retrieval. In *International Conference on Learning Representations, ICLR*.

Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, et al. 2021. Do transformer modifications transfer across implementations and applications? *arXiv preprint arXiv:2102.11972*.

Dwarak Govind Parthiban, Yongyi Mao, and Diana Inkpen. 2021. On the softmax bottleneck of recurrent language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13640–13647.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sara Rajaee and Mohammad Taher Pilehvar. 2021. A cluster-based approach for improving isotropy in contextual embedding space. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 575–584. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 65–75, Valencia, Spain. Association for Computational Linguistics.

Mariano Sigman and Guillermo A Cecchi. 2002. Global organization of the wordnet lexicon. *Proceedings of the National Academy of Sciences*, 99(3):1742–1747.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022. A contrastive framework for neural text generation. *arXiv preprint arXiv:2202.06417*.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4599–4609, Brussels, Belgium. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Gupta, Philip Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. 2021. Omninet: Omnidirectional representations from transformers. *arXiv preprint arXiv:2103.01075*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Yile Wang, Leyang Cui, and Yue Zhang. 2019. How can bert help lexical semantics tasks? *arXiv preprint arXiv:1911.02929*.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2021. Symbolic knowledge distillation: from general language models to commonsense models. *arXiv preprint arXiv:2110.07178*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Zhilin Yang, Thang Luong, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Mixtape: Breaking the softmax bottleneck efficiently. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15922–15930.

Avishai Zagoury, Einat Minkov, Idan Szpektor, and William W Cohen. 2021. What's the best place for an ai conference, vancouver or _: Why completing comparative questions is difficult. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14292–14300.

Zhong Zhang, Nian Shao, Chongming Gao, Rui Miao, Qinli Yang, and Junming Shao. 2022. Mixhead: Breaking the low-rank bottleneck in multi-head attention language models. *Knowledge-Based Systems*, page 108075.

A Appendix Overview

To demonstrate the wide applicability of our approaches, we conduct more experiments such as applying **MFS** to BERT in Appendix B. We also show more results and conduct more analyses in Appendix C to further support our conclusions. Next, we provide technical details including the proof of our theorems in Appendix D, show that the structure weakness studied by Demeter et al. (2020) is a special case of our theory in Appendix E, the method details in Appendix F, and the experiment details in Appendix G. Finally, in Appendix H, we list several directions that could be further studied in the future.

B More Experiments

We conduct the following five extra experiments to measure the linear dependency among word embeddings in LMs, extend our multi-facet approaches to BERT, confirm the source of the improvement comes from modeling multimodal distribution, and extend our synthetic experiments to include the output candidate words that have various types of relations and to include the template that favors the single embedding representation.

B.1 Linear Dependency among Words

Theorem 2 shows that when N words are linearly dependent after moving one of the embeddings with a short distance ϵ, the output softmax layer of a LM cannot output a large logit margin between two subsets of the N words. We want to measure ϵ in the pretrained word embedding and compare the ϵ from different sets of words or from different LMs.

Given a set of N words, we form a matrix by their word embeddings and estimate the ϵ value by the minimal eigenvalue of the matrix. We first want to verify that the four analogical words used in Section 5 indeed have a smaller ϵ compared to a randomly selected four words. Thus, we define the min eigenvalue ratio as $\frac{\epsilon_S}{\epsilon_R}$, where ϵ_R is the average of minimal eigenvalues from 1,000 sampled N word sets and ϵ_S is the average of minimal eigenvalues from sets of words (e.g., analogical words from the Google analogy dataset). We analyze the ratio instead of ϵ because the average word embedding magnitudes in different LMs would affect the absolute value of ϵ.

In addition to analogical words, we also test sets of N similar words, which are composed by the nearest $N-1$ words of every query word in the vocabulary, and test the N similar stop words by finding the nearest $N-1$ words of every query word in a stop word list.[6]

We plot the min eigenvalue ratio versus N in Figure 3 and compare the curves from three GPT LMs and two T5 LMs (Raffel et al., 2020). All the ratios are below 0 and decrease as N increases, which shows the analogical words and similar words indeed have significantly smaller ϵ especially for a large N. The low minimal eigenvalues and our theory support the recent empirical finds that LMs tend to be confused by the similar words (Zagoury et al., 2021). This figure also provides a potential explanation why the candidates often include stop words when multiple embeddings outperform the single embedding in Table 3 and Table 7.

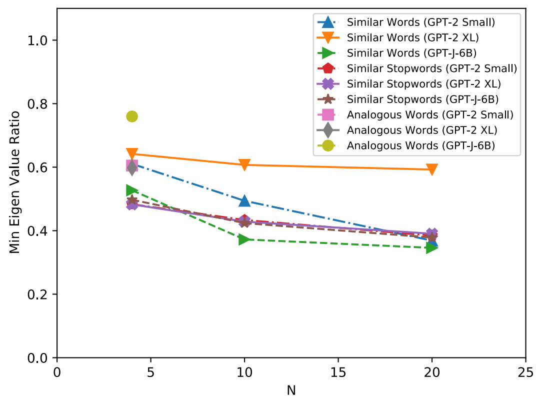
Surprisingly, we find that a larger LM does not necessarily yield a larger ratio (i.e., embeddings of related words do not become more linearly independent as dimension or the size of the LM increases). All the LMs have very similar ratios of similar stop words. Compared to *GPT-small*, although *GPT-J-6B* (Wang and Komatsuzaki, 2021) has a significantly higher ratio for analogical words, its ratio for similar words is significantly lower. Besides, *T5-11B* has significantly lower ratios compared to *T5-small*. We need further investigation to understand the reason for this empirical finding and whether a larger LM suffers less from the limitation caused by the single embedding.

B.2 Language Modeling using BERT

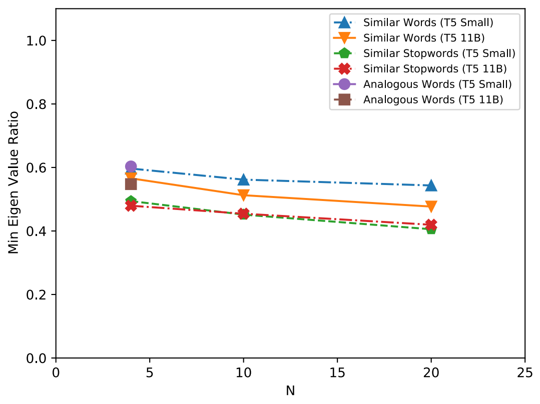
To demonstrate that our proposed method could improve the LMs other than GPT-2, we apply *multi-facet softmax*, **MFS**, to BERT. We test the model on Wikipedia 2021 and the validation size is 0.25% of the whole corpus. After loading pretrained model, we train *bert_base_cased* for 100k batches and *bert_large_cased* for 30k batches.

The results are presented in Table 6. First, **MoS** outperforms **Softmax** on BERT. The results support the finding of Narang et al. (2021) that the *softmax bottleneck* not only exists in the next word prediction tasks but also in the masked word prediction tasks. Similar to GPT-2, **MFS** at least doubles

[6]We find that some rare words or special characters might have nearly identical word embeddings due to the lack of training instances, so we exclude half of rarer word pieces in the vocabulary and exclude the word pieces whose first character is not a space. The rarity of a word piece is determined by the l2 norm of its word embedding.



(a) *GPT-2 Small* (0.1B, D=768), *GPT-2 XL* (1.5B, D=1600), and *GPT-J-6B* (6B, D=4096)



(b) *T5 Small* (0.06B, D=512) and *T5 11B* (11B, D=1024)

Figure 3: Minimal eigenvalue ratios to indicate the linear dependency among different groups of N word embeddings

BERT base after training on 100k batches

Softmax (S1I1P1)	SigSoftmax (S1I1P1)	
5.8699	5.8749	
Softmax + Multi-input (S1I9P1)	Softmax + Multi-partition (S1I1P4)	
5.8520	5.8656	
MoS (Yang et al., 2018) (4) (S4I1P1)	MoS (Yang et al., 2018) (3) (S3I1P1)	DOC (Takase et al., 2018) (S3I3P1)
5.8523	5.8535	5.8547
MFS w/o Multi-partition (S3I9P1)	MFS w/o Multi-input (S3I1P4)	MFS (S3I9P4)
5.8231	5.8536	**5.8231**

BERT large after training on 30k batches

Softmax (S1I1P1)	SigSoftmax (S1I1P1)	
4.8355	4.8354	
Softmax + Multi-input (S1I9P1)	Softmax + Multi-partition (S1I1P4)	
4.8305	4.8363	
MoS (Yang et al., 2018) (4) (S4I1P1)	MoS (Yang et al., 2018) (3) (S3I1P1)	DOC (Takase et al., 2018) (S3I3P1)
4.8268	4.8291	4.8231
MFS w/o Multi-partition (S3I9P1)	MFS w/o Multi-input (S3I1P4)	MFS (S3I9P4)
4.8111	4.8287	**4.8109**

Table 6: Perplexity of models building on BERT in Wikipedia 2021.

Corpus →	OpenWebText	Wikipedia 2021	Similar Nouns in Templates
Input Context	... "Part of the Clinton inevitability strategy was to lock down the usual suspects in left-liberal policy," said Dan Nexon, a Georgetown professor who served as one of those informal Sanders advisers. **Nex**	... The projective line over the dual numbers was described by Josef Grünwald in 1906. This ring includes a nonzero nilpotent "n" satisfying. The plane of dual numbers has a **project**	There are the militia and the enemy in front of a woman, and she decides to pursue the **militia**
Softmax (GPT-2)	He 0.014, But 0.011, The 0.007	finite 0.062, hom 0.059, **project** 0.034	enemy 0.860, **militia** 0.111, Militia 0.005
MFS (Ours)	**Nex** 0.013, He 0.012, But 0.011	**project** 0.096, hom 0.049, dual 0.046	enemy 0.535, **militia** 0.433, enemies 0.029
MFS Avg	", He, But, The, In, And, (, It	hom, dual, finite, non, ", complex, unit	**militia**, enemy, Militia, enemies, militias
MFS Softmax 1	But 0.005, He 0.004, The 0.002	**project** 0.201, dual 0.075, finite 0.030	enemy 0.772, **militia** 0.189, Militia 0.017
MFS Softmax 2	**Nex** 0.260, " 0.028, He 0.023	hom 0.093, unit 0.040, non 0.037	**militia** 0.938, Militia 0.062, militias 0.000
MFS Softmax 3	He 0.025, But 0.022, The 0.014	finite 0.065, map 0.041, plane 0.030	enemy 1.000, enemies 0.000, foe 0.003

Table 7: Prediction visualization using a context in each dataset. Each row visualizes a model as in Table 3. The models are built on *GPT-2 Medium* in OpenWebText and Wikipedia and on *GPT-2 Small* in the synthesized dataset. *MFS Avg* shows the words that are closest to the average facet embedding in *MFS*. See the details in Appendix B.3. We underline the words that appear in the top predictions of both *MFS* and *MFS Avg*.

the improvement of **MoS**. The most improvement over **MoS** comes from using multiple input hidden states while adding multiple partitions yield a small or no improvement. Finally, the improvement between **MFS** and **Softmax** is around 4.5%, which is much smaller than 15% in GPT-2.

Corpus $\rightarrow$	OpenWebText					Wikipedia 2021				
Improvement Model	S3I9P4	S3I9P4	S3I9P4	S3I9P1	S3I1P1	S3I9P4	S3I9P4	S3I9P4	S3I9P1	S3I1P1
Reference Model	S3I9P1	S3I1P1	S1I1P1	S1I9P1	S1I1P1	S3I9P1	S3I1P1	S1I1P1	S1I9P1	S1I1P1
Multi-mode Percentage (%)	10.03	10.03	10.03	4.81	3.24	5.85	5.85	5.85	2.66	3.05
Multi-mode Loss Improvement	0.0248	0.0474	0.0649	0.0203	0.0110	0.0282	0.0644	0.1000	0.0472	0.0295
Other Loss Improvement	0.0035	0.0158	0.0211	0.0086	0.0064	0.0033	0.0128	0.0219	0.0136	0.0100
Improvement Ratio	7.01	3.00	3.08	2.34	1.71	8.63	5.04	4.57	3.47	2.94

Table 8: The loss improvement comparison between the *Improvement Models* and *Reference Models*. The models are named using their number of softmaxes, input hidden states, and partitions. Thus, S3I9P4 is *MFS*, S3I9P1 is *MFS w/o Multi-partition*, S1I9P1 is *Softmax + Multi-input*, S3I1P1 is *MoS (3)*, and S1I1P1 is *Softmax*. *Multi-mode Percentage* is the percentage of the contexts where the *Improvement Models* output multimodal distribution. *Multi-mode Loss Improvement* refers to the average improvement when *Improvement Models* outputs multimodal distribution and *Other Loss Improvement* refers to the improvement of the contexts where the facets of *Improvement Models* are close to each other. *Improvement Ratio* divides *Multi-mode Loss Improvement* by *Other Loss Improvement*.

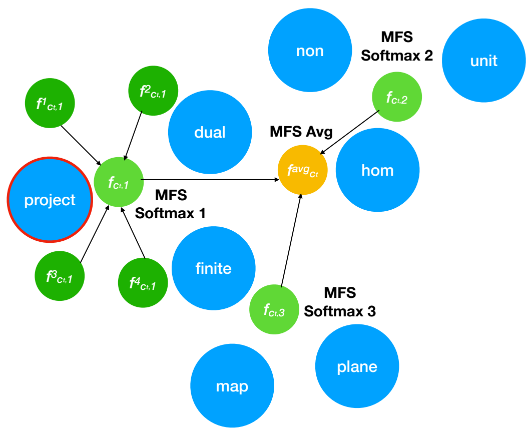


Figure 4: Illustration of the MFS predictions given the Wikipedia context in the second column of Table 7. The green circles mean the facet embeddings from MFS. The orange circle is the average of the facet embeddings (**MFS Avg**). The blue circles are the word embeddings that are close to the facet embeddings and **MFS Avg**. The word *project* is highlighted because it is the next word in our ground truth.

The smaller improvement supports the conclusion of our geometric analyses that the multi-mode ambiguity intensifies the *softmax bottleneck*. We only observe the one-directional context before the next target word in GPT-2, but we can observe the bi-directional context surrounding the masked target word in BERT. Thus, compared to next word prediction, the multi-mode ambiguity of the masked word prediction occurs less frequently when the masking probability is small (e.g., 15% in BERT). Since the masked word distribution only has a single mode most of the time but we sometimes still want the distribution to have multiple modes, multiple input hidden states can improve the performance by helping the facets to move more freely. On the other hand, multiple partitions are less useful because the distribution rarely has more than three modes.

B.3 Analysis of Improvement on Multimodal Distribution

To confirm that the perplexity improvements actually come from modeling the multimodal distribution, we define a metric to measure how multi-mode a distribution is, and then we can compare the perplexity improvement from multimodal distributions and the improvement from the distributions that are close to a single-mode distribution.

For the method with multiple embeddings, we first compute the weighted average of all the facets $\boldsymbol{f}^{avg}_{ct} = \sum_{k=1}^{K} \pi_{ct,k} \boldsymbol{f}_{ct,k}$, where we lower the influence of kth facet embedding $\boldsymbol{f}_{ct,k}$ with lower prior weight $\pi_{ct,k}$ and $\boldsymbol{f}_{ct,1} = \frac{1}{J} \sum_{j=1}^{J} \boldsymbol{f}^{j}_{ct,1}$ if J partitions are used. Figure 4 illustrates $\boldsymbol{f}^{avg}_{ct}$ and $\boldsymbol{f}_{ct,k}$ using the example in the second column of Table 7.

We visualize the new average facet using the words that are closest to the $\boldsymbol{f}^{avg}_{ct}$ in the **MFS Avg** row of Table 7. We can see that the predictions of **MFS Avg** is different from **MFS** but similar to **Softmax**. This means there are indeed some other words between the actual next word and the other possibilities, which makes the predictions of **MFS** multi-mode.

Next, to quantify the difference between **MFS** and **MFS Avg**, we define *multi-mode ratio* as $\frac{\sum_{b=1}^{T} P_M(y_b|ct)}{\sum_{b=1}^{T} P_M(x_b|ct)}$, where P_M could be either P_{MoS} from equation 2 or P_{MP} from equation 4. $\{y_1, ..., y_T\}$ is the set of words with embeddings closest to $\boldsymbol{f}^{avg}_{ct}$ and $\{x_1, ..., x_T\}$ is the set

Models ↓	Dissimilar Words			Similar Words		
	Testing	Validation	Training	Testing	Validation	Training
Softmax	1.97	1.98	1.95	2.16	2.16	2.17
MoS (3)	1.81	1.80	**1.69**	2.05	**2.05**	**1.87**
MFS w/o Multi-partition	**1.78**	**1.79**	1.70	2.04	2.06	1.88
MFS	1.79	**1.79**	**1.69**	**2.02**	**2.05**	1.89

Table 9: Perplexity comparison of different models on the similar words or dissimilar words. The models are based on *GPT-2 Small* and trained in OpenWebText.

GPT-2 Small after 1 epoch		
Softmax (S1I1P1)	SigSoftmax (S1I1P1)	
0.5494	0.5489	
Softmax + Multi-input (S1I9P1)	Softmax + Multi-partition (S1I1P4)	
0.5508	0.5492	
MoS (Yang et al., 2018) (4) (S4I1P1)	MoS (Yang et al., 2018) (3) (S3I1P1)	DOC (Takase et al., 2018) (S3I3P1)
0.5501	0.5499	0.5494
MFS w/o Multi-partition (S3I9P1)	MFS w/o Multi-input (S3I1P4)	MFS (S3I9P4)
0.5515	0.5502	**0.5519**

GPT-2 Medium after 0.4 epoch		
Softmax (S1I1P1)	SigSoftmax (S1I1P1)	
0.5665	0.5650	
Softmax + Multi-input (S1I9P1)	Softmax + Multi-partition (S1I1P4)	
0.5677	0.5665	
MoS (Yang et al., 2018) (4) (S4I1P1)	MoS (Yang et al., 2018) (3) (S3I1P1)	DOC (Takase et al., 2018) (S3I3P1)
0.5674	0.5672	0.5665
MFS w/o Multi-partition (S3I9P1)	MFS w/o Multi-input (S3I1P4)	MFS (S3I9P4)
0.5685	0.5677	**0.5685**

Table 10: MRR (mean reciprocal rank) of different models in OpenWebText. Larger is better.

of words with highest $P_M(x_b|c_t)$. Using the Wikipedia context in Table 7 as an example, the word *project* is retrieved by **MFS** but not by **MFS Avg**, so its *multi-mode ratio* for $T = 2$ is $\frac{P_{MFS}(hom|c_t)+P_{MFS}(dual|c_t)}{P_{MFS}(project|c_t)+P_{MFS}(hom|c_t)} = \frac{0.049+0.046}{0.096+0.049} \approx 0.66$. Figure 4 illustrates the relation between the **MFS Softmax k** and **MFS Avg**.

When the ratio is closer to 1, the context is less ambiguous and the prediction is closer to a single-mode distribution. We set $T = 20$ and call the prediction with *multi-mode ratio* smaller than 0.9 multimodal distribution and in Table 8,[7] we compare the loss (i.e., log of the perplexity) improvements in the multimodal distributions and the improvements in the nearly single-mode distributions.

Table 8 shows that all the multiple embedding approaches have larger loss improvements when outputting multimodal distributions. The table shows the results based on *GPT-2 Small* and the same analysis using *GPT-2 Medium* also show the same trend. As we use multiple input hidden states and partitions, the differences would be enlarged. Especially when we compare **MFS** and **MFS w/o Multi-partition**, the loss improvements of highly ambiguous context is 7 or 8 times larger than the other loss improvements, which means a large portion of the overall improvement lies on a small percentage of ambiguous contexts. For the multimodal distribution in Wikipedia, the loss improvement between **MFS** and **Softmax** could reach 0.10, which is close to the improvement between GPT-2 Small and Medium (0.16). Thus, we expect that if the corpus has more ambiguous contexts, **MFS** could achieve larger overall loss improvement.

[7]We also tried T=5 or 10 and the trends are similar. If we set the threshold smaller than 0.9, the improvement ratios (e.g., MFS over MoS) would increase but the multi-mode percentages would decrease.

B.4 Template-based Analysis on Similar or Dissimilar Nouns

To know whether the single embedding also has trouble modeling the distribution over nouns without the analogy relation, we let the different models learn to assign similarly high probabilities to two related nouns in our templates. One example in our synthesized dataset is "*I love the **banana** and the **lemon**, and my favorite is the [MASK]*". The nouns come from a hypernymy detection benchmark (Shwartz et al., 2017) containing 25,498 noun pairs. The relations between nouns in the benchmark include synonym, antonym, attribute, meronym, hypernym, coordination, event, or random. We further split the noun pairs into two datasets based on their cosine similarity in the output word embedding space of our **Softmax** baseline. The pairs with the cosine similarity higher than the medium of all cosine similarities are put into the similar word set and the other pairs are put into the dissimilar word set.

The results are presented in Table 9. In terms of the training, validation, and testing perplexity, multi-embedding approaches consistently outperform the single-embedding baselines, though the margins are smaller than those from the analogous words. Moreover, the improvement gap is larger when the nouns are dissimilar. We hypothesize that as the word embeddings of nouns become further away from each other, the next word distribution is more likely to be multi-mode and thus could be better captured by multiple embeddings.

B.5 Adversarial Template Analysis

To test whether the proposed methods still can effectively utilize the information from the global word embeddings, we design an adversarial template to create the contexts that can only be completed by averaging the global word embeddings. For example, *"**Miami** is not in **Wisconsin** but is in [MASK]=**Florida**"*.

In this task, the validation perplexity of **Softmax**, **MoS**, **MFS w/o Multi-partition**, and **MFS** are 2.50, 2.59, 2.54, and 2.88, respectively. Since multiple embeddings are not required, it is not surprising that **Softmax** performs the best. Nevertheless, the differences are smaller than the differences in Table 4. We believe that the similar losses are because multiple embeddings are a generalization of the single embedding, so GPT-2 could learn to generate the same embedding for all facets to mimic the behavior of single embedding if required.

The significantly worse performance of **MFS** here is caused by the multiple partition technique. This result supports our motivation of combining multiple partitions with multiple softmaxes and shows that multiple partitions handle ambiguous contexts better (as shown in Table 8) by sacrificing some global word embedding structures. Nevertheless, a corpus usually has more ambiguous contexts than the adversarial context tested here, so using multiple embeddings and multiple partitions performs better in Wikipedia and OpenWebText overall.

C More Results

We provide more numbers and analyses of our experiments.

C.1 Ranking Metric in Language Modeling Experiments

We would like to verify that our perplexity improvements come from not only the slight probability differences of each candidate but also the better ranks of the candidates. Thus, in Table 10, we evaluate different models using mean reciprocal rank (MRR). Similar to the perplexity, the MRR improvement from **Softmax** to **MFS** is around 15% of the MRR improvement from *GPT-2 Small* to *GPT-2 Medium*, which is similar to the percentage of perplexity improvement. This suggests that **MFS** could lead to not only a better probability prediction but also a better candidate rank prediction.

C.2 Perplexity Curves in Language Modeling Experiments

In Table 1, we only show the testing perplexity at the end of our training. In Figure 6, we plot the validation perplexity decay curves during the training on OpenWebText. We can see that the performance ranking of each model is stable during the training, while the improvement of each enhancement may vary. For example, in *GPT-2 Medium*, the improvement of **MFS** over **MFS w/o Multi-partition** is more obvious in epoch 0.25 compared to epoch 0.4.

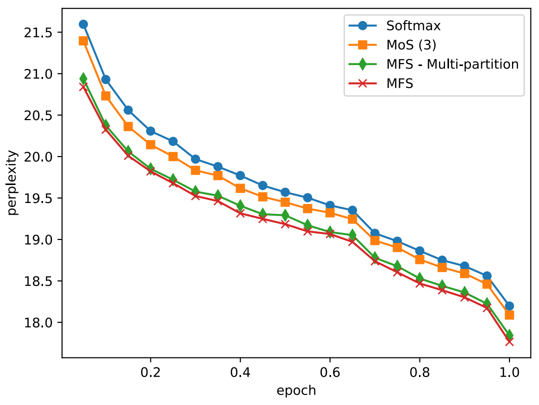
C.3 Perplexity Curves in Template Analysis

In Table 4, we only show the lowest validation perplexity after each of the ten epochs. In Figure 5, we plot the training and validation perplexity decay curves.

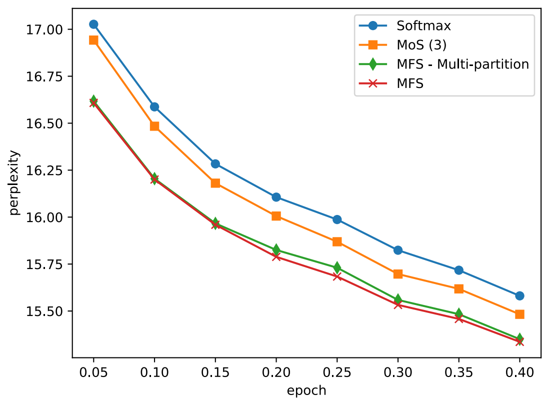
The curves tell us that the multi-embedding models perform better in both training and validation perplexity. As we train the single-embedding models longer, the validation perplexity increases quickly, which implies that using a single embedding to model multimodal distribution could cause severe overfitting when we predict the next word given an ambiguous context.

C.4 Stability in Language Modeling Experiments

In our case, training our model requires a huge amount of GPU resources for us, so it is not very

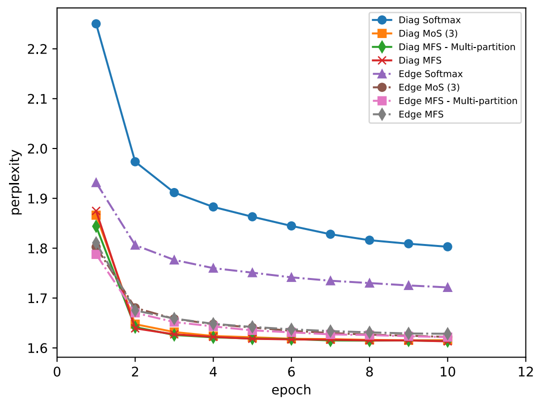


(a) Curves on *GPT-2 Small*

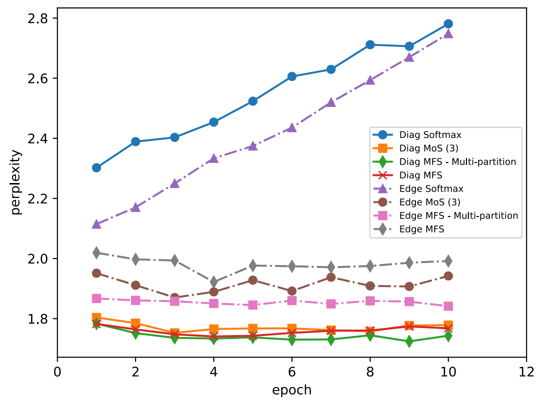


(b) Curves on *GPT-2 Medium*

Figure 5: The perplexity curves for the language modeling tasks using the validation set of OpenWebText.



(a) Perplexity in the training data



(b) Perplexity in the validation data

Figure 6: The perplexity curves from different models for the ambiguous template analysis

	Max Answers				Max Incorrect		
Models ↓	Top 1	Top 3	Top 5	Top 10	Top 1	Top 3	Top 5
Softmax (GPT-2)	36.5 $\pm$ 0.7	39.7 $\pm$ 0.5	43.5 $\pm$ 0.4	52.2 $\pm$ 0.6	20.9 $\pm$ 0.4	37.7 $\pm$ 0.6	46.7 $\pm$ 0.6
MoS (Yang et al., 2018) (3)	36.6 $\pm$ 0.8	40.2 $\pm$ 0.6	43.2 $\pm$ 0.6	52.1 $\pm$ 0.4	21.3 $\pm$ 0.6	38.4 $\pm$ 0.5	45.9 $\pm$ 0.6
MFS w/o Multi-partition	**37.7 $\pm$ 0.7**	**42.0 $\pm$ 0.6**	**44.6 $\pm$ 0.5**	**52.6 $\pm$ 0.3**	22.9 $\pm$ 0.4	39.5 $\pm$ 0.5	**47.4 $\pm$ 0.4**
MFS	36.9 $\pm$ 0.7	41.6 $\pm$ 0.7	44.4 $\pm$ 0.6	52.3 $\pm$ 0.6	**23.1 $\pm$ 0.5**	**39.7 $\pm$ 0.6**	46.9 $\pm$ 0.6

Table 11: ProtoQA performances on the crowdsourced development sets. The matching between prediction and ground truth is done by WordNet. All the numbers are percentages. Max answers top k implies only evaluating the top k answers from different LMs. Max incorrect top k indicates only evaluating the top answers that contain k errors. The highest average performances are highlighted and the standard errors are reported as the confidence interval.

feasible to train multiple times using multiple random seeds. We indeed try to use different random seeds for a few models and we confirm that the validation loss difference is at least ten times smaller than the improvement of different models.

To verify that our testing dataset is large enough to provide stable perplexity, we randomly split the testing dataset into 10 subsets and compute the standard error of the average testing perplexity of the 10 subsets. We find that the standard error is less than 0.02 perplexity in all models and datasets in Table 1. The standard error is much smaller than most of the improvements, which means our testing dataset is large enough to make the reported perplexity stable. The consistent improvements during the whole training process in Figure 5 further sup-

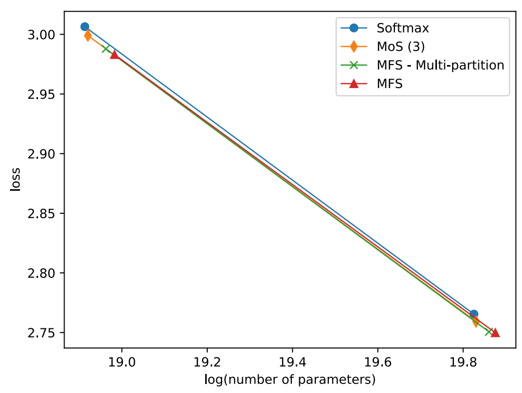


Figure 7: The log of model size versus the log of perplexity in the text set of OpenWebText. The group of points on the left comes from the models based on *GPT-2 Small*. The group of points on the right comes from the models based on *GPT-2 Medium*. The models are trained for 0.4 epoch.

port the stability of our experiments.

C.5 ProtoQA Results using WordNet

In Table 5, we report the metrics using exact matching. In Table 11, we report the metrics that match the prediction with the ground truth using WordNet (Miller, 1992) and find the scores show a similar trend.

C.6 Perplexity Improvement versus Model Size

Kaplan et al. (2020) empirically demonstrate that increasing the model size would decrease the loss and their relation follows a scaling law. That is, we can plot the log of model size (i.e., parameter number) versus its loss as in Figure 7, and if a new LM model could result in lines that are closer to the origin than the baselines, the new model is better in terms of the loss than only increasing the model size of the baselines.

From Figure 7, we can see that the approaches using multiple embedding are better than the **Softmax** baseline using single embedding. Although the lines formed by **MFS w/o Multi-partition** and **MFS** are not always closer to the origin than **MoS**, our perplexity improvement from adding multiple input hidden states or multiple partitions cannot be solely explained by their extra parameters for several reasons:

- Compared to **MoS**, the line formed by **MFS w/o Multi-partition** becomes slightly closer to the origin when the model size is close to *GPT-2 Medium*.
- The improvement of **MFS w/o Multi-partitions (S3I9P1)** is larger than the improvement of **Softmax + Multi-input (S1I9P1)** plus the improvement of **MoS (S3I1P1)** in BERT and GPT-2. For example, in BERT base, the perplexity improvement of **Softmax + Multi-input**, **MoS (3)**, and **MFS w/o Multi-partitions** are 0.018, 0.016, and 0.047, respectively.
- Our multi-mode analyses in Appendix B.3 indicate that our enhancements, especially using multiple partitions, capture the multimodal distribution better. We expect that the overall perplexity improvement would be larger if the corpus contains more ambiguous contexts. We also conduct a preliminary experiment to confirm the claim. We add more ambiguous contexts into Wikipedia 2016 by mapping all the uppercased words into the $[UNK]$ token. That is, we add another mode corresponding to the $[UNK]$ token in many context positions. Then, we train and test the uncased BERT in this synthesized dataset. We found that the improvement of **MFS w/o Multi-partition** in this case can do significantly better than simply increasing the model size.
- Our enhancements only require some extra linear layers, which are usually more efficient than increasing the model size (e.g., by adding another transformer layer).
- Unlike increasing the model size, keep increasing the number of input hidden states or the number of partitions would lead to a smaller improvement. This suggests that **MFS** cannot keep storing more and more knowledge into its extra linear layers as in the architecture using a larger hidden state size or a deeper transformer encoder.

C.7 More Visualization

In Table 3, we compare the prediction of **MFS** and **Softmax** on *GPT-2 Small*. In the first two columns of Table 7, we present the examples from the models built on *GPT-2 Medium* in OpenWebText and Wikipedia 2021. We can see a similar pattern. The embedding of the correct answer is different from the embeddings of other possibilities, so **Softmax** assigns lower probabilities to the correct answer, while **MFS** does much better. This suggests that a larger model such as *GPT-2 Medium* suffers from the *softmax bottleneck* in a similar way.

In the last column of Table 7, we visualize an example in another synthetic experiment described in Appendix B.4. We can see that although there may not be any words between the appropriate candidates, the prediction of **Softmax** may still be biased toward one option much more than the other, while the prediction of **MFS** is much closer to the equally likely bimodal distribution we created in the training data.

D Proof of Theorems

To prove Theorem 1, we first introduce a lemma. Assuming in the word embedding of GPT-2, $\underline{\textbf{\textit{woman}}}$ + $\underline{\textbf{\textit{king}}}$ = $\underline{\textbf{\textit{queen}}}$ + $\underline{\textbf{\textit{man}}}$, we want to show that GPT-2 cannot output *woman* and *king* as the top two words in this lemma. This means we cannot find a hidden state $\boldsymbol{h}$ and a threshold $\tau > 0$ such that $\boldsymbol{h}^T\underline{\textbf{\textit{woman}}} \geq \tau$ and $\boldsymbol{h}^T\underline{\textbf{\textit{king}}} \geq \tau$ but $\boldsymbol{h}^T\underline{\textbf{\textit{queen}}} < \tau$ and $\boldsymbol{h}^T\underline{\textbf{\textit{man}}} < \tau$. This example could be generalized into the following Lemma and Theorems. We can generalize the example as follows:

Lemma 1. *Let the output word embeddings in the set* $W = \{\boldsymbol{w}_{l_j} \neq \boldsymbol{0} | j = 1...L\} \cup \{\boldsymbol{w}_{r_j} \neq \boldsymbol{0} | j = 1...R\}$ *satisfy* $-a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R}$*, where their coefficient* $-a_{l_1}, ..., -a_{l_L}, a_{r_1}, ..., a_{r_R}$ *are all positive constants and* $-a_{l_1} - ... - a_{l_L} \geq a_{r_1} + ... + a_{r_R}$*. Then, there is no hidden state* $\boldsymbol{h}$ *and a threshold* $\tau > 0$ *that make* $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T\boldsymbol{w}_g \geq \tau$ *and* $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T\boldsymbol{w}_s < \tau$*, where* $G = \{\boldsymbol{w}_{l_j} | j = 1...L\}$ *and* $S = \{\boldsymbol{w}_{r_j} | j = 1...R\}$.

Proof. To prove by contradiction, we assume there is a $\boldsymbol{h}$ such that $\forall \boldsymbol{w}_{l_j} \in G, \boldsymbol{h}^T\boldsymbol{w}_{l_j} \geq \tau$ and $\forall \boldsymbol{w}_{r_j} \in S, \boldsymbol{h}^T\boldsymbol{w}_{r_j} < \tau$. Thus, we can get $-a_{l_1}\boldsymbol{h}^T\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{h}^T\boldsymbol{w}_{l_L} \geq -a_{l_1}\tau - ... - a_{l_L}\tau \geq (a_{r_1} + ... + a_{r_R})\tau > a_{r_1}\boldsymbol{h}^T\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{h}^T\boldsymbol{w}_{r_R}$, which contradicts to $-a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R}$. □

We can rephrase the condition and the conclusion to have our Theorem 1.

Theorem 1. *If the nonzero output embeddings of* N *words in a set* W *are linearly dependent and on one side of a hyperplane through the origin, the single embedding representation cannot produce positive logits to a subset of the words in* W *that are higher than all the logits of the other words in* W.[8]

Proof. The set $W = \{\boldsymbol{w}_i \neq \boldsymbol{0} | i = 1...N\}$ contain the embeddings of the N words. Based on the premise, we can write $\boldsymbol{0} = a_1\boldsymbol{w}_1 + ... + a_N\boldsymbol{w}_N$ and $\min_{\boldsymbol{w}_i \in W} \boldsymbol{h}_0^T\boldsymbol{w}_i > 0$, where $\boldsymbol{h}_0$ is a normal vector of the hyperplane. At least one of the a_i is negative. Otherwise, we will get the contradiction $0 = \boldsymbol{h}_0^T\boldsymbol{0} = a_1\boldsymbol{h}_0^T\boldsymbol{w}_1 + ... + a_N\boldsymbol{h}_0^T\boldsymbol{w}_N \geq (a_1 + ... + a_N)\min_{\boldsymbol{w}_i \in W} \boldsymbol{h}_0^T\boldsymbol{w}_i > 0$. Similarly, at least one of a_i is positive. We can move all the terms in $\boldsymbol{0} = a_1\boldsymbol{w}_1 + ... + a_N\boldsymbol{w}_N$ with negative a_i to the left as $-a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R}$. If $-a_{l_1} - ... - a_{l_L} \geq a_{r_1} + ... + a_{r_R}$, we choose $G = \{\boldsymbol{w}_{l_j} | j = 1...L\}$. Otherwise, we choose $G = \{\boldsymbol{w}_{r_j} | j = 1...R\}$

If we can have a hidden state such that the positive logits of words in G are always larger than the logits of the other words in W (let's call the complementary set S), there must exist $\tau > 0$ that can make $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T\boldsymbol{w}_g \geq \tau$ and $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T\boldsymbol{w}_s < \tau$, which violates our Lemma 1. □

Next, we would like to generalize our Theorem 1 by using a more practical condition where the word embeddings are almost linearly dependent. Notice that the theorem needs to assume the magnitude of the hidden state is limited. Otherwise, the margin could be arbitrarily magnified. In practice, the magnitude is not arbitrarily large in GPT-2 and BERT because a too large magnitude of hidden state could magnify the gradients too much to have a stable training process.

Theorem 2. *Let the output word embeddings in the set* $W = \{\boldsymbol{w}_i \neq \boldsymbol{0} | i = 1...N\}$ *satisfy* $\boldsymbol{w}_1 = a_2\boldsymbol{w}_2 + ... + a_N\boldsymbol{w}_N + \boldsymbol{\varepsilon}$*, where the constant* $a_2, ..., a_N$ *are neither all zero nor all neg-*

[8]Notice that Theorem 1 does not cover the situations where the target top words have negative logits (i.e., some logits of the words in G are negative). In the single softmax model, we believe the situations rarely happen in the LMs empirically.

If some logits of the target top words are still positive, the words that are somehow similar to those words are very likely to also be positive, which would be ranked higher than the target top words with the negative logits.

If the logits of all the target top words are negative in some contexts, the logits of all the words would be negative. Then, the word embeddings with smaller magnitudes tend to have the logits closer to 0, so having the larger logits than the other negative logits. This means the prior probability of the words would be inversed when the hidden states sometimes produce all negative logits.

If a LM always uses negative logits to compute probability in all the contexts, Lemma 1 and Theorem 1 still hold if we set $\tau < 0$ and switch the choices of G and S.

ative and $||\varepsilon|| < \epsilon$. Then, there must be a non-trivial partition $P = \{G, S\}$ of W such that there is no hidden state $||\boldsymbol{h}|| \leq r$ and a threshold $\tau \geq r\epsilon$ that makes $\min_{\boldsymbol{w}_g \in G} \boldsymbol{h}^T \boldsymbol{w}_g \geq (1+\delta)\tau$ and $\max_{\boldsymbol{w}_s \in S} \boldsymbol{h}^T \boldsymbol{w}_s < \tau$, where $\delta = \frac{2}{1+\sum_{i=2...N} |a_i|}$.

Proof. We can first move all the terms with negative a_i to the left as $\boldsymbol{w}_1 - a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R} + \varepsilon$. We perform proof by contradiction, so we assume the logits of the words in G can always be larger than $(1+\delta)\tau$ and the logits of the words in S can always be smaller than τ.

Case 1: $1 - a_{l_1} - ... - a_{l_L} \geq a_{r_1} + ... + a_{r_R}$, so $1 - a_{l_1} - ... - a_{l_L} \geq \frac{1+\sum_{i=2...N}|a_i|}{2}$. We choose $G = \{\boldsymbol{w}_1, \boldsymbol{w}_{l_1}, ..., \boldsymbol{w}_{l_L}\}$ and $S = \{\boldsymbol{w}_{r_1}, ..., \boldsymbol{w}_{r_R}\}$. Thus, we can get $\boldsymbol{h}^T\varepsilon \leq ||\boldsymbol{h}||||\varepsilon|| \leq r\epsilon \leq \tau$ and

$$\boldsymbol{h}^T\boldsymbol{w}_1 - a_{l_1}\boldsymbol{h}^T\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{h}^T\boldsymbol{w}_{l_L} \tag{5}$$
$$\geq (1 - a_{l_1} - ... - a_{l_L})(1+\delta)\tau \tag{6}$$
$$= (1 - a_{l_1} - ... - a_{l_L})(1 + \frac{2}{1+\sum_{i=2...N}|a_i|})\tau \tag{7}$$
$$\geq (1 - a_{l_1} - ... - a_{l_L})(1 + \frac{1}{1 - a_{l_1} - ... - a_{l_L}})\tau \tag{8}$$
$$= (1 - a_{l_1} - ... - a_{l_L} + 1)\tau \tag{9}$$
$$\geq (a_{r_1} + ... + a_{r_R} + 1)\tau \tag{10}$$
$$> a_{r_1}\boldsymbol{h}^T\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{h}^T\boldsymbol{w}_{r_R} + \boldsymbol{h}^T\varepsilon, \tag{11}$$

which contradict with $\boldsymbol{w}_1 - a_{l_1}\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{w}_{l_L} = a_{r_1}\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{w}_{r_R} + \varepsilon$.

Case 2: $1 - a_{l_1} - ... - a_{l_L} < a_{r_1} + ... + a_{r_R}$. We choose $G = \{\boldsymbol{w}_{r_1}, ..., \boldsymbol{w}_{r_R}\}$ and $S = \{\boldsymbol{w}_1, \boldsymbol{w}_{l_1}, ..., \boldsymbol{w}_{l_L}\}$. Therefore,

$$a_{r_1}\boldsymbol{h}^T\boldsymbol{w}_{r_1} + ... + a_{r_R}\boldsymbol{h}^T\boldsymbol{w}_{r_R} \tag{12}$$
$$\geq (a_{r_1} + ... + a_{r_R})(1 + \frac{2}{1+\sum_{i=2...N}|a_i|})\tau \tag{13}$$
$$> (a_{r_1} + ... + a_{r_R})(1 + \frac{1}{a_{r_1} + ... + a_{r_R}})\tau \tag{14}$$
$$= (a_{r_1} + ... + a_{r_R} + 1)\tau \tag{15}$$
$$> (1 - a_{l_1} - ... - a_{l_L} + 1)\tau \tag{16}$$
$$> \boldsymbol{h}^T\boldsymbol{w}_1 - a_{l_1}\boldsymbol{h}^T\boldsymbol{w}_{l_1} - ... - a_{l_L}\boldsymbol{h}^T\boldsymbol{w}_{l_L} - \boldsymbol{h}^T\varepsilon. \tag{17}$$

□

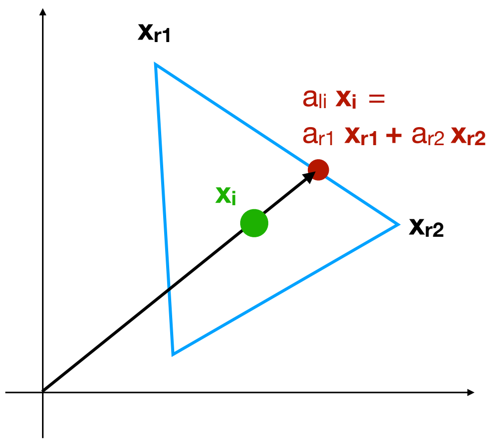


Figure 8: An example for explaining the connection between our Theorem 1 and the theorem from Demeter et al. (2020).

E Theoretical Connection to Demeter et al. (2020)

The theory in Demeter et al. (2020) is as follows: "*Let C be the convex hull of the embeddings $\{\boldsymbol{x}_i\}$ of a vocabulary V. If an embedding $\boldsymbol{x}_i$ for word $\boldsymbol{w}_i \in V$ is interior to C, then the maximum probability $P(\boldsymbol{w}_i)$ assigned to $\boldsymbol{w}_i$ using a dot-product softmax is bounded by the probability assigned to at least one word $\boldsymbol{w}_i$ whose embedding is on the convex hull*"

The theory is a special case of our Lemma 1 if we only consider the hidden states that would lead to the positive logit of the interior word $\boldsymbol{w}_i$. To see that, we first find a constant $a_{l_i} > 1$ such that $a_{l_i}\boldsymbol{x}_i$ intersects with one supporting hyperplane of the convex hull. This intersection point could be expressed by $\sum_j a_{r_j}\boldsymbol{x}_{r_j}$, where the word embeddings $\boldsymbol{x}_{r_j}$ are vertexes of C and $\sum_j a_{r_j} = 1$. As a result, we satisfy the condition of our Lemma 1: $a_{l_i}\boldsymbol{x}_i = \sum_j a_{r_j}\boldsymbol{x}_{r_j}$ and $a_{l_i} > \sum_j a_{r_j}$. Please see an illustration in Figure 8 for an example. Then, Lemma 1 suggests that the logit $\boldsymbol{h}^T\boldsymbol{x}_i$ cannot be larger than the logits of all the word embeddings $\boldsymbol{h}^T\boldsymbol{x}_{r_j}$. This means at least one of the $\boldsymbol{h}^T\boldsymbol{x}_{r_j}$ on the convex hull would lead to a larger prediction probability, which is also the conclusion of the theory in Demeter et al. (2020).

F Method Details

When replacing the softmax layer in the pretrained LMs, we found that the initialization of the extra linear layers should make the initial prediction of LMs close to the prediction using a softmax layer, which is the architecture used in the pretraining. Otherwise, the performance would drop significantly. The initialization is especially important for BERT. To achieve the goal, we initialize the weights of the linear layers such that different facets are almost identical at the beginning and let the LMs gradually learn to output diverse facets during the training. Specifically, we can write the linear layer on the new hidden state $L^f_k(\boldsymbol{q}_{c_t})$ as

$$\begin{aligned}\boldsymbol{f}_{c_t,k} &= L^f_k(\boldsymbol{q}_{c_t}) \\ &= \mathbf{L^I_k} \boldsymbol{h}^M_{c_t} + \mathbf{L^B_k} GELU\left(L^h(\oplus_{i,m} \boldsymbol{h}^{M-m}_{c_{t-i}})\right) + \boldsymbol{b}.\end{aligned} \tag{18}$$

We initialize $\mathbf{L^I_k}$ as an identity matrix, $\boldsymbol{b} \leftarrow \mathbf{0}$, and $\mathbf{L^B_k} \leftarrow \mathcal{U}(-\epsilon, \epsilon)$, where $\mathcal{U}$ is the uniform distribution and $\epsilon = 0.00005$ if $k \neq K$. Otherwise, $\epsilon = 0$. Consequently, all the facets $\boldsymbol{f}_{c_t,k}$ are initially close to the last hidden state of the original GPT-2 $\boldsymbol{h}^M_{c_t}$. Our baselines (e.g., **Softmax**, **MoS**, and **DOC**) also adopt the same way to initialize their weights.

We implement our models based on huggingface[9] (Wolf et al., 2020). Please see our codes for more details.

F.1 Architecture Differences in BERT

The architecture of **MFS** for BERT is mostly the same as the one for GPT-2 and the differences are described in this subsection.

In GPT-2 the block of input hidden state is right-aligned with the last word to prevent seeing the ground truth. On the other hand, the block in BERT is centered at the masked word.

The softmax layer of BERT is slightly different from that of GPT-2. For example, BERT adds a bias term after the dot product between the hidden state and the output word embedding. We keep the bias term in our experiments. Besides, the pretrained BERT has a language modeling head including a linear layer, a GELU (Gaussian Error Linear Unit) layer (Hendrycks and Gimpel, 2016), and a layernorm layer (Ba et al., 2016), so instead of adding an extra linear layer as in GPT-2, we just use different language modeling heads to create different facets in BERT. All the heads are initialized using the weights in the pretrained BERT except that the linear layer is initialized as in Equation 18 when the multiple input hidden states are used and the corresponding linear weights $\mathbf{L^B_k} \leftarrow \mathcal{U}(-\epsilon, \epsilon)$, where $\epsilon = 0.05$ if $k \neq K$. Otherwise, $\epsilon = 0$.

G Experimental Details

In this section, we describe some details of our experimental setup.

G.1 Baselines

The **MoS** (Yang et al., 2018) and **DOC** (Takase et al., 2018) are originally designed for RNN-based LM. To improve their methods on pretrained Transformer-based LM and make their results more comparable to **MFS**, we change some of their implementation details.

MoS originally has a tanh layer before the softmax layers. However, we found that adding tanh hurts the performances of all methods we tested, especially the **Softmax** and **MoS** baselines. For example, after adding tanh and training *GPT-2 Small* for 0.4 epoch on Wikipedia, the validation perplexity degradation of **Softmax** is from 25.70 to 26.15, the degradation of **MoS** is from 25.42 to 25.83, and the degeneration of **MFS** is from 25.06 to 25.12. We suspect this is because GPT-2 is pretrained without the tanh layer and the tanh limits the magnitude of facets $||\boldsymbol{f}_{c_t,k}||$, which could be viewed as the inverse of the temperature in the softmax layer. Therefore, we remove the tanh layer in all of our experiments. From the theoretical perspective, adding tanh does not invalidate our motivation because adding tanh does not change the total degree of freedom in all facet embeddings and the dimension of the hidden state.

In **DOC**, we use the hidden states of the last three transformer layers to compute the three facets and we set $\lambda_\beta = 0$. Each facet is only determined by one layer of hidden state, so the first two facets cannot access the last hidden state. We found that the model quickly learns to only use the last facet because only the last hidden state is trained to perform the LM task in the pretrained models. This prevents the first two facets from getting any gradients and causes a starvation problem.

We tried an aggressive dropout trick to solve the starvation problem in **DOC**. If one of the softmaxes does not assign the highest probability to any of the correct next words in a batch, we consider that the corresponding facet starves, so we drop the other facets with some probability to ensure this starved facet receives some gradients and gradually gets back on track. However, our preliminary experi-

[9] https://huggingface.co/

ment suggests that the dropout trick cannot improve the perplexity of **DOC**. The dropout probability is either too low to solve the starvation problem or too high to preserve the knowledge learned from pretraining. Thus, we do not adopt this trick in our final experiment.

G.2 Language Modeling

We download Wikipedia using `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor` and OpenWebText using `https://github.com/jcpeterson/openwebtext`. For Wikipedia, we preprocess the text using `https://github.com/attardi/wikiextractor`. For OpenWebText, we download the pre-filtered URLs in 2017 and 2018 and scrape the text on April 2021. When splitting the corpus into training, validation, and testing sets, we do not shuffle the data. Instead, we use the text near the end of the corpus as the validation and test set to reduce information leakage. To ensure every model is trained on the same data and accelerate the training in our machines, we split the training data into 20 consecutive partitions and load only one partition at a time during training. When training *GPT-2 Medium*, we only use the first 8 partitions to let the training be finished within a week. For BERT, we perform the sentence segmentation using SpaCy[10] and input one sentence into BERT at a time.

We set our hyperparameters (e.g., facet number $K = 3$ and $W \times H = 3 \times 3$ when using multiple input hidden states) based on the validation performance in Wikipedia 2016, the resulting model size, and the memory constraint in GPUs. To explore the limitation of the softmax layer, we untie the input word embeddings and output word embeddings in all of our experiments. The untying allows the LMs to arrange the output word embeddings more freely and allows us to observe if the resulting output word embeddings still cause multi-mode distribution. This is also the main reason the model size of our *GPT-2* baseline is larger than the size of pretrained *GPT-2* (Radford et al., 2019). We use AdamW (Loshchilov and Hutter, 2019) optimizer and set the learning rate as 1e-5 and do not use the warm-up because the training starts from the pretrained models. The sequence length (i.e., bptt) is set as 200 for GPT-2 and 256 for BERT. The batch sizes are set as 4 for *GPT-2 Small*, 16 for *GPT-2 Large*, 120 for *BERT base*, and 128 for *BERT large*.

The analyses in Table 2 and Table 8 use the first 4000 sequences in the validation dataset and all the methods are based on *GPT-2 Small*. We use PYCLD2[11] to distinguish between English and non-English text.

We use NVIDIA GeForce RTX 2080 for training *GPT-2 Small* and *BERT base*, GeForce RTX 8000 for training *GPT-2 Medium*, Tesla M40 for training *BERT large*. Since we start from the pretrained LM, we can finish training each LM within 2 weeks using 1 GPU for *GPT-2 Small*, *BERT base*, and *GPT-2 Medium*, and using 4 GPUs for training *BERT large*.

When testing the inference time in Table 1, we average the time of running NVIDIA TITAN X on 10,000 batches, where each batch contains 4 sequences whose length are 200.

When visualizing the prediction in Table 3, we exclude the non-ASCII symbol prediction from the top word list of all models.

G.3 Ambiguous Templates Analysis

Among the semantic relations in Google analogy dataset, we choose three different relations between locations: *capital-common-countries*, *capital-world*, *city-in-state*, and one relation between people: *family*. We exclude the *currency* category because their instance often does not form a parallelogram in the word embedding space (Ethayarajh et al., 2019). The templates we use are listed in Table 12. For the *family* category, our templates assume the words are not pronouns, so we exclude the set of four words that include *he* or *she*.

For each of the four words in an analogy instance (e.g., $\underline{\textbf{\textit{queen}}} : \underline{\textbf{\textit{king}}} = \underline{\textbf{\textit{woman}}} : \underline{\textbf{\textit{man}}}$), we would create 32 training or testing sequences[12] based on the diagonal words such as *king* or *woman*. Similarly, we would create 64 sequences in the edge datasets. Some words contain multiple word pieces and we average the losses of all word pieces during training and testing.

We split the synthesized sequences based on their word pair overlapping. First, we randomly sample half of the word pairs (e.g., *king* and *queen*) in each category as our training pairs. If both of the word pairs in an analogy instance are training

[10]`https://spacy.io/`

[11]`https://github.com/aboSamoor/pycld2`)

[12]2 (diagonal words) $\times$ 4 (templates) $\times$ 2 (word orders in the template) $\times$ 2 (possible next words)

Dataset ↓	Templates
Anology (Person or Person)	Between the *$ARG1* and the *$ARG2*, I decided to first talk to the [MASK]
	The *$ARG1* and the *$ARG2* are my favorites, and I especially love the [MASK]
	The *$ARG1* and the *$ARG2* happily live together. One day, bad luck happens to the [MASK]
	The *$ARG1* and the *$ARG2* stay at my house, and I need to take care of the [MASK]
Anology (Location or Location)	I went to *$ARG1* and *$ARG2* before, and I love one of the places more, which is [MASK]
	$ARG1 and *$ARG2* are my favorites, and I especially love [MASK]
	My uncle used to live in *$ARG1* and *$ARG2* but now, he is selling his house in [MASK]
	The traveler plans to visit *$ARG1* and *$ARG2*, and the traveler first arrives in [MASK]
Similarity (Noun or Noun)	I love the *$ARG1* and the *$ARG2*, and my favorite is the [MASK]
	Yesterday, a man encountered the *$ARG1* and the *$ARG2*. Today, he again saw the [MASK]
	There are the *$ARG1* and the *$ARG2* in front of a woman, and she decides to pursue the [MASK]
	If you can choose the *$ARG1* or the *$ARG2*, would you choose the [MASK]

Table 12: The templates used in the analysis. The first four templates are for the analogy relations from the *capital-common-countries*, *capital-world*, and *city-in-state* categories. The next four templates are for the analogy relations from the *family* category. The final four templates are for similar or dissimilar nouns.

pairs, the instance is put into our training set. If only one of the word pairs is a training pair, the instance would belong to our validation set. The rest of the instances form our testing set. During the fine-tuning, we evaluate a model using the validation set after each epoch and select the model based on its best validation perplexity.

G.4 ProtoQA Evaluation

In our experiments, we use the scraped development set as our validation set and the crowdsourced development set as our test set. We do not test our methods on the test set of ProtoQA because the result of every submission would show up in their leaderboard and we do not want to overwhelm the leaderboard with our 15 trials.

Due to our limited GPU resources, we compare the methods built on *GPT-2 Medium* rather than *GPT-2 Large*. To maximize the perplexity of the *GPT-2 Medium* model using Softmax on the scraped development set, we fine-tune our models using learning rate 3e-5 and warmup step 500.

The original paper (Boratko et al., 2020) does not consider the frequency of the answer during the fine-tuning (i.e., the most possible answer and the least possible answer of each question appear in the training data with the same chance). In terms of the performance of the scraped development set, we find that weighting each answer based on the square root of its frequency is better than weighting each answer uniformly as in the original paper or weighting each answer based on its frequency, so we use the square root weighting to finetuning all our models.

During testing time, each model generates the answers using Nucleus Sampling (Holtzman et al., 2020) with p = 0.9 and temperature = 1. Then, we collect all the words before the first period as an answer and drop the generated sentences without a period.

H Future Work

Capturing the next word distribution well given an ambiguous context could be important in some downstream applications. A next step could be investigating whether multiple facets lead to a better language generation model for the applications. For example, we would like to know whether breaking the *softmax bottleneck* could reduce the hallucination of LMs (e.g., outputting *queen* when the reasonable next words should be *king* or *woman*) and increase the coherence of the generated text. We also want to more systematically investigate whether modeling multi-mode distribution could help LMs to reduce the undesired bias and to better distinguish similar words (Zagoury et al., 2021) as in Appendix B.4.

Narang et al. (2021); Anonymous (2021) find that *MoS* can significantly improve the BERT-like LMs on natural language understanding (NLU) tasks when the LMs are trained from scratch. Although we find that the perplexity improvement of multi-embedding BERT is not as large as multi-embedding GPT-2, pretraining using multiple embeddings does not decrease the inference speed of the BERT encoder on NLU tasks. This motivates the future studies that test if *MFS* also provides a larger improvement than *MoS* in NLU tasks.

Table 2 suggests that multiple embeddings improve more in a non-English context. We wonder whether multiple embeddings are more beneficial to the LMs that are trained on a non-English dominating corpus. Chung et al. (2021) discover that using a larger output embedding dimension improves the multilingual BERT. An interesting research question is whether the improvement comes from alleviating the *softmax bottleneck* and whether *MFS* could also lead to similar improvements in multilingual benchmarks.

The hidden state size of GPT-3 175B (Brown et al., 2020) is huge (12,288). An interesting question is whether some sets of output word embeddings in GPT-3 are still in a low-dimensional subspace and whether the *softmax bottleneck* is still a prominent problem on the road of pursuing general intelligence when such a large hidden state dimension is used. We also would like to know if models using multiple facets could reach a similar performance by a smaller hidden state size.

Recently, Gao et al. (2019a); Rajaee and Pilehvar (2021); Cai et al. (2021); Su et al. (2022) point out the structure in the contextual embedding space prevents it from having an isotropic property. Our study and Demeter et al. (2020) show that the structure in the word embedding space only models the global similarity between words and prevents the LM from outputting arbitrary context-dependent word distributions. We would like to know if we can discover a new LM architecture with a better contextual/word embedding space that could better model context-dependent word similarities and balance it with the global word similarities. In addition, our finding might be one of the reasons that we can improve the language generation quality by encouraging word embedding to be more isotropic (Su et al., 2022).

Gao et al. (2019b) show that a mixture of kernel functions outperforms MoS. *Mixtape* (Yang et al., 2019) is another efficient solution to the *softmax bottleneck*, whose hidden state for each word is the weighted average of the facets where the weights are dynamically predicted. If only using one softmax (i.e., $K = 1$), our multiple partition method could be viewed as a special case of *Mixtape* that uses a global and binarized weight to prevent complications of predicting the weights of each word. Our results indicate that multiple partitions need to be combined with multiple softmax layers in order to gain consistent performance improvement. A potential future direction is to compare *MFS* with a *mixture of kernel functions* and *Mixtape* on the transformer-based LMs or combine *MFS* with a *mixture of kernel functions* and *Mixtape* to gain further improvements.

The results in Kong et al. (2020) suggest that predicting n-grams could be better than predicting individual words in BERT in some applications. The total number of possible n-grams is several orders of magnitude higher than the number of individual tokens in the vocabulary. In addition, the linear dependency among n-grams might be common. For example, the embedding of the ***brown color*** + ***a dog*** may be similar to the embedding of the ***brown dog***. The problem would be more serious as the length of the prediction sequence (n) increases, so predicting the next sentence using a single embedding might suffer from the *softmax bottleneck* even more. Therefore, our solutions to *softmax bottleneck* may lead to a better phrase representation or sentence representation in this type of self-supervised pretraining.

Finally, language modeling is only an example of extreme classification. The nearly ubiquitous usage of single embedding representation in the classification, self-supervised models (e.g., contrastive learning models), or recommendation problems provides many research opportunities. We believe that our theoretical results could guide researchers to identify the potential applications where the *softmax bottleneck* is serious and multi-embedding representation is accordingly helpful.