A Generalized Language Model in Tensor Space

Lipeng Zhang[†], Peng Zhang[†*], Xindian Ma[†], Shuqin Gu[†], Zhan Su[†], Dawei Song[‡]
[†]College of Intelligence and Computing, Tianjin University, Tianjin, China
[‡]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
{lpzhang, pzhang, xindianma, shuqingu, suzhan}@tju.edu.cn, dawei.song2010@gmail.com

Abstract

In the literature, tensors have been effectively used for capturing the context information in language models. However, the existing methods usually adopt relatively-low order tensors, which have limited expressive power in modeling language. Developing a higher-order tensor representation is challenging, in terms of deriving an effective solution and showing its generality. In this paper, we propose a language model named Tensor Space Language Model (TSLM), by utilizing tensor networks and tensor decomposition. In TSLM, we build a high-dimensional semantic space constructed by the tensor product of word vectors. Theoretically, we prove that such tensor representation is a generalization of the n-gram language model. We further show that this high-order tensor representation can be decomposed to a recursive calculation of conditional probability for language modeling. The experimental results on Penn Tree Bank (PTB) dataset and WikiText benchmark demonstrate the effectiveness of TSLM.

Introduction

Language Modeling (LM) is a fundamental research topic that underpins a wide range of Natural Language Processing (NLP) tasks, e.g., speech recognition, machine translation, and dialog system (Yu and Deng 2014; Lopez 2008; Wang, Chung, and Seneff 2006). Statistical learning of a language model aims to approximate the probability distribution on the set of expressions in the language (Brown et al. 1992). Recently, Neural networks (NNs), e.g., Recurrent Neural Networks (RNNs) have been shown effective for modeling language (Bengio et al. 2003; Mikolov et al. 2010; Jozefowicz et al. 2016).

In the literature, a dense tensor is often used to represent a sentence or document. Cai et al. (2006) proposed TensorLSI, which considered sentences or documents as 2-order tensors (matrices) and tried to find an optimal basis for the tensor subspace in term of reconstruction error. The 2-order tensor (matrix) only reflects the local information (e.g., bi-gram information). Liu et al. (2005) proposed to model text by a multilinear algebraic tensor instead of a vector. Specifically, they represented texts using 3-order tensors to capture the context of words. However, they still adopted relatively low-order tensors, rather than high-order ones constructed by the tensor product of vectors. For a sentence with n words ($n > 3$), a n-order tensor (a high-order tensor) constructed by the tensor product of n word vectors, can consider all the combinatorial dependencies among words (not limited to two/three consecutive words in 2/3-order tensor). It turns out that low-order tensors have limited expressive power in modeling language.

It is challenging to construct a high-order tensor based language model (LM), in the sense that it will involve an exponentially increasing number of parameters. Therefore, the research problems are how to derive an effective high-order tensor based LM and how to demonstrate the generality of such a tensor-based LM. To address these problems, our motivation is to explore the expressive ability of tensor space in depth, by making use of tensor networks and tensor decomposition, in the language modeling process.

Tensor network is an elegant mathematical tool and an effective method for solving high-order tensors (e.g., tensors in quantum many-body problem), through contractions among lower-order tensors (Pellionisz and Llins 1980). Recent research shows the connections between neural networks and tensor networks, which provides a novel perspective for the interpretation of neural network (Cohen et al. (2016) and Levine et al. (2017; 2018)). With the help of tensor decomposition, the high dimensionality of parameters in tensor space can be reduced greatly. Based on tensors, a novel language representation using quantum many-body wave function was also proposed in (Zhang et al. 2018).

Inspired by the recent research, in this paper, we propose a Tensor Space Language Model (TSLM), which is based on high-dimensional semantic space constructed by the tensor product of word vectors. Tensor operations (e.g., multiplication, inner product, decomposition) can be represented intuitively in tensor networks. Then, the probability of a sentence will be obtained by the inner product of two tensors, corresponding to the input data and the global parameters, respectively.

Theoretically, we prove that TSLM is a generalization of the n-gram language model, by showing that the conditional probability of a language sequence can be calculated by the tensor representation we constructed. We further show that, after tensor decomposition, the high-order tensor representation can be decomposed to a recursive calculation of

*Corresponding author: Peng Zhang (pzhang@tju.edu.cn)

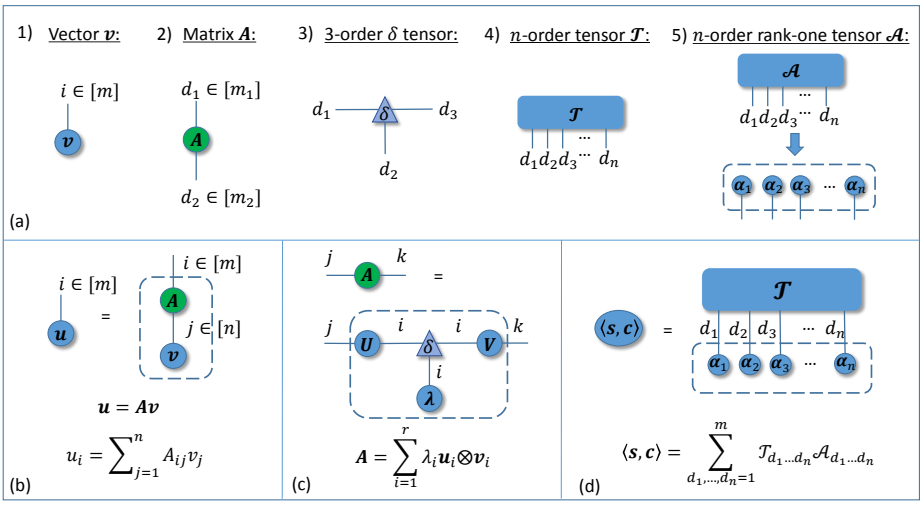

Figure 1: Introduction of Tensor Networks. (a) Tensors in the TN are represented by nodes. (b) A matrix A multiplying a vector $\boldsymbol{v}$ in TN notation. The contracted indices are denoted by j and are summed upon. The open indices are denoted by i, their number equals the order of the tensor represented by the entire network. The contraction is marked by the dashed line. (c) A TN notation illustrates the SVD of a matrix A. (d) A TN notation shows the inner product of two tensors.

conditional probability for language modeling. Finally, we evaluate our model on the PTB dataset and the WikiText benchmark, and the experimental results demonstrate the effectiveness of TSLM.

The main contributions of our work can be summarized as follows: (1) We propose a novel language model aiming to consider high-order dependencies of words via tensors and tensor networks. (2) We prove that TSLM is a generalization of the n-gram language model. (3) We can derive a recursive calculation of conditional probability for language modeling via tensor decomposition in TSLM.

Preliminaries

We first briefly introduce tensors and tensor networks.

Tensors

1. A **tensor** can be thought as a multidimensional array. The **order** of a tensor is defined to be the number of indexing entries in the array, which are referred to as modes. The **dimension** of a tensor in a particular mode is defined as the number of values that may be taken by the index in that mode. A tensor $\mathcal{T} \in \mathbb{R}^{m_1 \times \cdots \times m_n}$ means that it is a n-order tensor with dimension m_i in each mode $i \in [n] := \{1, \ldots, n\}$. For simplicity, we also call it a m^n-dimensional tensor in the following text. A specific entry in a tensor will be referenced with subscripts, e.g. $\mathcal{T}_{d_1 \ldots d_n} \in \mathbb{R}$.

2. Tensor product is a fundamental operator in tensor analysis, denoted by $\otimes$, which can map two low-order tensors to a high-order tensor. Similarly, the tensor product of two vector spaces is a high-dimensional tensor space. For example, tensor product intakes two tensors $\mathcal{A} \in \mathbb{R}^{m_1 \times \cdots \times m_j}$ (j-order) and $\mathcal{B} \in \mathbb{R}^{m_{j+1} \times \cdots \times m_{j+k}}$ (k-order), and returns a tensor $\mathcal{A} \otimes \mathcal{B} = \mathcal{T} \in \mathbb{R}^{m_1 \times \cdots \times m_{j+k}}$ ((j+k)-order) defined by : $\mathcal{T}_{d_1 \ldots d_{j+k}} = \mathcal{A}_{d_1 \ldots d_j} \cdot \mathcal{B}_{d_{j+1} \ldots d_{j+k}}$.

3. A n-order tensor $\mathcal{A}$ is **rank-one** if it can be written as the tensor product of n vectors, i.e.,

$$\mathcal{A} = \boldsymbol{\alpha}_1 \otimes \boldsymbol{\alpha}_2 \otimes \cdots \otimes \boldsymbol{\alpha}_n \tag{1}$$

This means that each entry of the tensor is the product of the corresponding vector elements:

$$\mathcal{A}_{d_1 d_2 \ldots d_n} = \alpha_{1,d_1} \alpha_{2,d_2} \cdots \alpha_{n,d_n} \quad \forall\, i, d_i \in [m_i] \tag{2}$$

4. The **rank** of a tensor $\mathcal{T}$ is defined as the smallest number of rank-one tensors that generate $\mathcal{T}$ as their sum (Hitchcock 1927; Kolda and Bader 2009).

Tensor Networks

A **Tensor Network** (TN) is formally represented by an undirected and weighted graph. The basic building blocks of a TN are tensors, which are represented by nodes in the network. The order of a tensor is equal to the number of edges incident to it. The weight of a edge is equal to the dimension of the corresponding mode of a tensor. Fig. 1 (a) shows five examples for tensors: 1) A vector is a node with one edge. 2) A matrix is a node with two edges. 3) In particular, a triangle node represents a δ tensor, $\delta \in \mathbb{R}^{m \times \cdots \times m}$, which is defined as follow:

$$\delta_{d_1 \ldots d_n} = \begin{cases} 1, & d_1 = \cdots = d_n \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

with $d_i \in [m]\ \forall i \in [n]$, i.e. its entries are equal to one only on the super-diagonal and zero otherwise. 4) The rounded rectangle node is the same as the circle node representing an arbitrary tensor. Accordingly, a tensor of order n is represented in the TN as a node with n edges. 5) A n-order rank-one tensor $\mathcal{A}$ can be represented by the tensor product of n vectors.

Edges which connect two nodes in the TN represent a contraction operation between the two corresponding

modes. An example for a contraction is depicted in Fig. 1 (b), in which a TN corresponding to the operation of multiplying a vector $\boldsymbol{v} \in \mathbb{R}^n$ by a matrix $A \in \mathbb{R}^{m \times n}$ is performed by summing over the only contracted index j. As there is only one open index i, the result of contracting the network is a vector: $\boldsymbol{u} \in \mathbb{R}^m$ which upholds $\boldsymbol{u} = A\boldsymbol{v}$.

An important concept in the later analysis is Singular Value Decomposition (SVD), which denotes that a matrix $A \in \mathbb{R}^{m \times n}$ can be decomposed as $A = U\Lambda V$, where $\Lambda \in \mathbb{R}^{r \times r}$ represents a diagonal matrix and $U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}$ represent orthogonal matrices. In TN notation, we represent the SVD as $A = \sum_{i=1}^{r} \lambda_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i$ in Fig. 1 (c). λ_i is a singular value, $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ are the components of U, V respectively. The effect of the δ tensor is shown obviously, which can be observed as 'forcing' the i-th 'row' of any tensor to be multiplied only by the i-th 'rows' of other tensors.

Fig. 1 (d) shows the **inner product** of two tensors $\mathcal{T}$ and $\mathcal{A}$, which returns a scalar value that is the sum of the products of their entries (Kolda and Bader 2009).

Basic Formulation of Language Modeling in Tensor Space

In this section, we describe the basic formation of Tensor Space Language Model (TSLM), which is used for computing probability for the occurrence of a sequence $s = w_1^n := (w_1, \ldots, w_n)$ of length n, composed of $|\mathrm{V}|$ different tokens, and the vocabulary V containing all the words in the model, i.e. $w_i \in \mathrm{V}$. In the next sections, we will prove that when we use one-hot vectors, TSLM is a generalization of n-gram language model. If using word embedding vectors, TSLM can result in a recursive calculation of conditional probability for language modeling. In Related Work, we will discuss the relations and differences between our previous work on language representation for matching sentences (Zhang et al. 2018) and the tensor space language modeling in this paper.

Representations of Words and Sentences

First, we define the semantic space of a single word as a m-dimensional vector space $\mathbb{V}$ with the orthogonal basis $\{\boldsymbol{e}_d\}_{d=1}^m$, where each base vector $\boldsymbol{e}_d$ is corresponding to a specific semantic meaning. A word w_i in a sentence s can be written as a linear combination of the m orthogonal basis vectors as a general representation:

$$\boldsymbol{w}_i = \sum_{d_i=1}^{m} \alpha_{i,d_i} \boldsymbol{e}_{d_i} \tag{4}$$

where α_{i,d_i} is its corresponding coefficient. For the basis vectors $\{\boldsymbol{e}_d\}_{d=1}^m$, there can be two different choices, one-hot vectors or embedded vectors.

As for a sentence $s = (w_1, \ldots, w_n)$ with length n, it can be represented in the tensor space:

$$\mathbb{V}^{\otimes n} := \underbrace{\mathbb{V} \otimes \mathbb{V} \otimes \cdots \otimes \mathbb{V}}_{n} \tag{5}$$

as:

$$\boldsymbol{s} = \boldsymbol{w}_1 \otimes \cdots \otimes \boldsymbol{w}_n \tag{6}$$

which is a m^n-dimensional tensor. Through the interaction of each dimension of words, the sentence tensor has a strong expressive power. Substituting Eq. 4 into Eq. 6, the representation of the sentence $\boldsymbol{s}$ can be expanded by:

$$\boldsymbol{s} = \sum_{d_1,\ldots,d_n=1}^{m} \mathcal{A}_{d_1 \ldots d_n} \boldsymbol{e}_{d_1} \otimes \cdots \otimes \boldsymbol{e}_{d_n} \tag{7}$$

where $\{\boldsymbol{e}_{d_1} \otimes \cdots \otimes \boldsymbol{e}_{d_n}\}_{d_1,\ldots,d_n=1}^m$ are the basis with m^n dimension in the tensor space $\mathbb{V}^{\otimes n}$, which denotes the high-dimensional semantic meaning. $\mathcal{A}$ is a m^n-dimensional tensor and its each entry $\mathcal{A}_{d_1 \ldots d_n}$ is the corresponding coefficient of each basis. According to the Eq. 1 and 2, we will see $\mathcal{A}$, computed as $\prod_{i=1}^{n} \alpha_{i,d_i}$, is a rank-one tensor.

A Probability Estimation Method of Sentences

A goal of language modeling is to learn the joint probability function of sequences of words in a language (Bengio et al. 2003). We assume that each sentence s_i appears with a probability p_i. Then, we can construct a mixed representation c which is a linear combination of sentences, denoted as:

$$\boldsymbol{c} := \sum p_i \boldsymbol{s}_i \tag{8}$$

We consider the mixed representation $\boldsymbol{c}$ as a high-dimensional representation of a sequence containing n words. For each sentence s_i, it can be represented by coefficient tensor $\mathcal{A}_i$ and basis vectors. Therefore, based on Eqs. 7 and 8, the mixed representation $\boldsymbol{c}$ can be formulated with a coefficient tensor $\mathcal{T}$:

$$\boldsymbol{c} = \sum_{d_1,\ldots,d_n=1}^{m} \mathcal{T}_{d_1 \ldots d_n} \boldsymbol{e}_{d_1} \otimes \ldots \otimes \boldsymbol{e}_{d_n} \tag{9}$$

The difference between $\boldsymbol{s}$ in Eq. 7 and $\boldsymbol{c}$ in Eq. 9 is mainly on two different tensors $\mathcal{A}$ and $\mathcal{T}$. According to the **Preliminaries**, $\mathcal{A}$ is essentially rank-one while $\mathcal{T}$ has a higher rank and is harder to solve. We will show that the tensor $\mathcal{T}$ encodes the parameters, and $\mathcal{A}$ is the input, in our model.

In turn, if we have estimated such a mixed representation $\boldsymbol{c}$ (in fact, its coefficient tensor $\mathcal{T}$) from our model, we can get the probability p_i of a sentence s_i via computing the inner product of $\boldsymbol{c}$ and $\boldsymbol{s}_i$:

$$p(s_i) = \langle \boldsymbol{s}_i, \boldsymbol{c} \rangle \tag{10}$$

Based on the Eq. 7, 9 and 10, we can obtain:

$$p(s_i) = \sum_{d_1,\ldots,d_n=1}^{m} \mathcal{T}_{d_1 \ldots d_n} \mathcal{A}_{d_1 \ldots d_n} \tag{11}$$

as shown in the tensor network of Fig 1(d). This is the basic formula in TSLM for estimating the sentence probability.

TSLM as a Generalization of N-Gram Language Model

The goal of the n-gram language model is to estimate the probability distribution of sentences. For a specific sentence s, its joint probability $p(s) = p(w_1^n) := p(w_1, \ldots, w_n)$ relies on the Markov Chain Rule of conditional probability:

$$p(w_1^n) = p(w_1) \prod_{i=2}^{n} p(w_i | w_1^{i-1}) \tag{12}$$

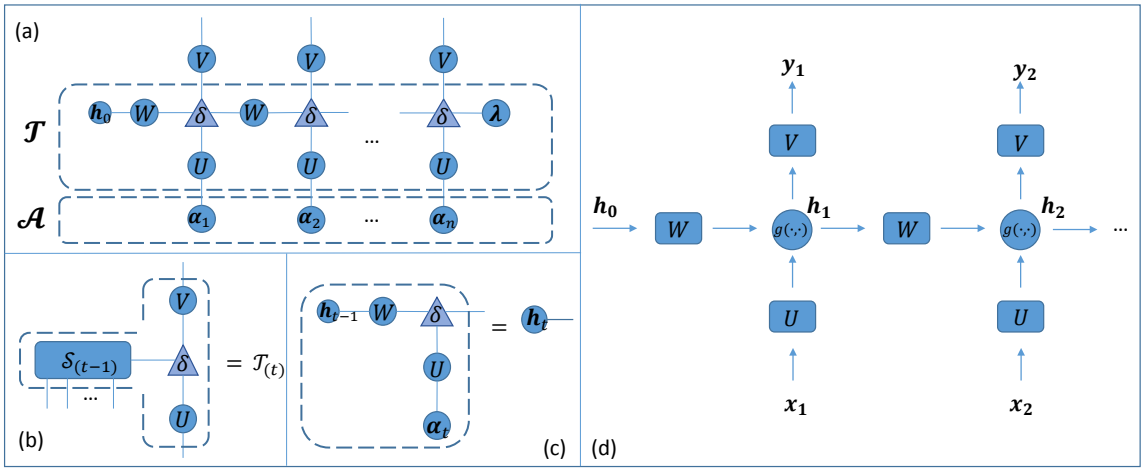

Figure 2: The TN represents the recursive calculation process of TSLM. (a) represents the inner product of two tensors $\mathcal{T}$ and $\mathcal{A}$, (b) denotes the recursive representations of $\mathcal{T}_{(t)}$ and (c) is the recursive representations of $\boldsymbol{h}_t$, respectively. (d) is a general RNN architecture.

and the conditional probability $p(w_i|w_1^{i-1})$ can be calculated as:

$$p(w_i|w_1^{i-1}) = \frac{p(w_1^i)}{p(w_1^{i-1})} \approx \frac{count(w_1^i)}{count(w_1^{i-1})} \tag{13}$$

where the $count$ denotes the frequency statistics in corpus.

Claim 1. In our TSLM, when we set the dimension of vector space $m = |\mathrm{V}|$ and each word w as an *one-hot* vector, the probability of sentence s consist of words $d_1 \dots d_n$ in vocabulary is the entry $\mathcal{T}_{d_1 \dots d_n}$ of tensor $\mathcal{T}$.

Proof. The detailed proof can be found in Appendix. □

Intuitively, the specific sentence s can be represented as an one-hot tensor. The mixed representation $\boldsymbol{c}$ can be regarded as the total sampling distribution. The tensor inner product $\langle \boldsymbol{s}, \boldsymbol{c} \rangle$ represents statistics probability that a sentence s appears in a language.

Claim 2. In our TSLM, we define the word sequence $w_1^i = (w_1, w_2, \dots, w_i)$ with length i as:

$$\boldsymbol{w}_1^i := \boldsymbol{w}_1 \otimes \cdots \otimes \boldsymbol{w}_i \otimes \mathbf{1}_{i+1} \otimes \cdots \otimes \mathbf{1}_n \tag{14}$$

which means that the sequence w_1^i is padded via using full one vector $\mathbf{1}$. Then, the probability $p(w_1^i)$ can be computed as $p(w_1^i) = \langle \boldsymbol{w}_1^i, \boldsymbol{c} \rangle$.

Proof. It can be proved by the marginal probability of multiple discrete variables (in Appendix). □

Therefore, the conditional probability $p(w_i|w_1^{i-1})$ in n-gram language model can be calculated by Bayesian Conditional Probability Formula using tensor representations and tensor inner product as follow:

$$p(w_i|w_1^{i-1}) = \frac{\langle \boldsymbol{w}_1^i, \boldsymbol{c} \rangle}{\langle \boldsymbol{w}_1^{i-1}, \boldsymbol{c} \rangle} \tag{15}$$

This kind of representation of conditional probability is a special case of TSLM based on the one-hot basis vectors. Because of the one-hot vector representation, the n-gram language model has $O(|\mathrm{V}|^n)$ parameters. Compared with n-gram language model, our general model has $O(m^n)$ parameters ($m \ll |\mathrm{V}|$). However, the tensor space still contains exponential parameters, and in the next section, we will introduce tensor decomposition to deal with this problem.

Deriving Recursive Language Modeling Process from TSLM

We have defined the method for estimating probability of a sentence s by the inner product of two tensors $\mathcal{T}$ and $\mathcal{A}$ in Eq. 11. In this section, we describe the recursive language modeling process in Fig. 2. Firstly, our derivation is under the condition of basis vectors $\{\boldsymbol{e}_d\}_{d=1}^m$ as embedded vectors. Secondly, we recursively decompose the tensor $\mathcal{T}$ (see Fig. 3), to obtain the TN representation of tensor $\mathcal{T}$ in Fig. 2 (a). Then, we use the intermediate variables in Fig. 2 (bc) to estimate the conditional probability $p(w_t|w_1^{t-1})$. In the following, we introduce the recursive tensor decomposition, followed by the calculation of conditional probability.

Recursive Tensor Decomposition

We generalize the SVD from matrix to tensor as shown intuitively in Fig. 3, inspired by the train-style decomposition of Tensor-Train (Oseledets 2011) and Tucker Decomposition (Kolda and Bader 2009). Fig. 3 illustrates a high-order tensor recursively decomposed as several low-order tensor (vectors, matrices, etc.).

The formula of the recursive decomposition about tensor $\mathcal{T}$ is :

$$\begin{aligned} \mathcal{T} &= \sum_{i=1}^{r} \lambda_i \mathcal{S}_{(n),i} \otimes \boldsymbol{u}_i \\ \mathcal{S}_{(n),k} &= \sum_{i=1}^{r} W_{k,i} \mathcal{S}_{(n-1),i} \otimes \boldsymbol{u}_i \end{aligned} \tag{16}$$

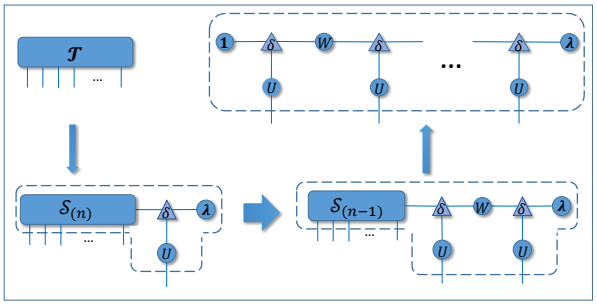

Figure 3: The recursive generalized SVD for the tensor $\mathcal{T}$.

where we define : $\mathcal{S}_{(1)} = \mathbf{1} \in \mathbb{R}^r$. It means that a n-order tensor $\mathcal{T} \in \mathbb{R}^{m \times \cdots \times m}$ can be decomposed as a n-order tensor[1] $\mathcal{S}_{(n)} \in \mathbb{R}^{m \times \cdots \times r}$, a diagonal matrix $\Lambda \in \mathbb{R}^{r \times r}$ and a matrix $U \in \mathbb{R}^{r \times m}$. One can consider this decomposition as the matrix SVD after tensor matricization, also as unfolding or flattening the tensor to matrix by one mode. Recursively, (n-1)-order tensor $\mathcal{S}_{(n),k}$, which can be seen as the k-th 'row' of the tensor $\mathcal{S}_{(n)}$, can be decomposed like the tensor $\mathcal{T}$ and W is a matrix composed by r groups of singular value vectors.

We employ this decomposition to extract the main features of the tensor $\mathcal{T}$ which is similar with the effect of SVD on matrix, then approximately represent the parameters of our model, where r ($r \leq m$) denotes the rank of tensor decomposition. This tensor decomposition method reduce the $O(m^n)$ magnitude of parameters approximatively to $O(m \times r)$.

A Recursive Calculation of Conditional Probability

In our model, we compute the conditional probability distribution as :

$$p(w_t|w_1^{t-1}) = softmax(\langle \mathcal{T}_{(t)}, \mathcal{A}_{(t-1)} \rangle) \tag{17}$$

where $\mathcal{A}_{(t-1)}$ is the input of (t-1) words, represented as $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{(t-1)}$ in Fig. 2 (a). $\langle \mathcal{T}_{(t)}, \mathcal{A}_{(t-1)} \rangle$ is denoted as $\boldsymbol{y}_t$.

As shown in Fig. 2 (b), $\mathcal{T}_{(t)} \in \mathbb{R}^{m \times \cdots \times m \times |\mathrm{V}|}$ is constructed by matrix $V \in \mathbb{R}^{r \times |\mathrm{V}|}$, $\mathcal{S}_{(t-1)}$ and matrix U. The V is the weighted matrix mapping to the vocabulary:

$$\mathcal{T}_{(t),k} = \sum_{i=1}^{r} V_{k,i} \mathcal{S}_{(t-1),i} \otimes \boldsymbol{u}_i \tag{18}$$

As shown in Fig. 2 (c), when calculating the inner product of two tensors $\mathcal{T}$ and $\mathcal{A}$, we can introduce intermediate variables $\boldsymbol{h}_t$, which can be recursively calculated as:

$$\begin{aligned} \boldsymbol{h}_1 &= W\boldsymbol{h}_0 \odot U\boldsymbol{\alpha}_1 \\ &\ldots \\ \boldsymbol{h}_t &= W\boldsymbol{h}_{t-1} \odot U\boldsymbol{\alpha}_t \end{aligned} \tag{19}$$

[1]To distinguish, we use the first parenthesized subscript to indicate the order of the tensor.

where setting the $\boldsymbol{h}_0 := W^{-1}\mathbf{1}$, and the matrices W and U decomposed from Eq. 16 in the last section. The symbol $\odot$ denotes the element-wise multiplication between vectors. This multiplicative operation is derived from the tensor product (in Eq. 16) and the δ-tensor, which we have explained it 'forces' the two vectors connected with it to be multiplied by elements (in Preliminaries).

Based on the analysis above, the recursive calculation of conditional probability of our TSLM can be formulated as:

$$\begin{aligned} p(w_t|w_1^{t-1}) &= softmax(\boldsymbol{y}_t) \\ \boldsymbol{y}_t &= V\boldsymbol{h}_t \\ \boldsymbol{h}_t &= g(W\boldsymbol{h}_{t-1}, U\boldsymbol{\alpha}_t) \\ g(\boldsymbol{a}, \boldsymbol{b}) &= \boldsymbol{a} \odot \boldsymbol{b} \end{aligned} \tag{20}$$

where $\boldsymbol{\alpha}_t \in \mathbb{R}^m$ is the input at time-step $t \in [n]$, $\boldsymbol{h}_t \in \mathbb{R}^r$ is the hidden state of the network, $\boldsymbol{y}_t \in \mathbb{R}^{|\mathrm{V}|}$ denotes the output, and the trainable weight parameters $U \in \mathbb{R}^{m \times r}, W \in \mathbb{R}^{r \times r}, V \in \mathbb{R}^{r \times |\mathrm{V}|}$ are the input to hidden, hidden to hidden and hidden to output weights matrices, respectively, and g is a non-linear operation.

The operation $\odot$ stands for element-wise multiplication between vectors, for which the resultant vector upholds $(\boldsymbol{a} \odot \boldsymbol{b})_i = a_i \cdot b_i$. Differently, in the RNN and TSLM architecture, g is defined as:

$$\begin{aligned} g_{RNN}(\boldsymbol{a}, \boldsymbol{b}) &= \sigma(\boldsymbol{a} + \boldsymbol{b}) \\ g_{TSLM}(\boldsymbol{a}, \boldsymbol{b}) &= \boldsymbol{a} \odot \boldsymbol{b} \end{aligned} \tag{21}$$

where $\sigma(\cdot)$ is typically a point-wise activation function such as sigmoid, tanh etc. A bias term is usually added to Eq. 21. Since it has no effect with our analysis, we omit it for simplicity. We show a general structure of RNN in Fig. 2(d).

In fact, recurrent networks that include the element-wise multiplication operation have been shown to outperform many of the existing RNN models (Sutskever, Martens, and Hinton 2011; Wu et al. 2016). Wu et al. (2016) had given a more general formula for hidden unit in RNN, named Multiplicative Integration, and discussed the different structures of the hidden unit in RNN.

Related Works

Here, we present a brief review of related work, including some representative work in language modeling, and the more recent research on the cross fields of tensor network, neural network and language modeling.

There have been tremendous research efforts in the field of statistical language modeling. Some earlier language models are based on the Markov assumption are represented by n-gram models (Brown et al. 1992), where the prediction of the next word is often conditioned just on n preceding words. For n-gram models, Kneser and Ney (2002) proposed the most well-known KN smoothing method, and some researchers continued to improve the smoothing method, as well as introduced the low-rank model. Neural Probabilistic Language Model (Bengio et al. 2003) is to learn the joint probability function of sequence of words in a language, which shows the improvement on n-gram models. Recently, RNN (Mikolov et al. 2010) and Long Short-

	PTB			WikiText-2		
	Train	Valid	Test	Train	Valid	Test
Articles	-	-	-	600	60	60
Tokens	929,590	73,761	82,431	2,088,628	217,646	245,569
Vocab size		10,000			33,278	
OOV rate		4.8%			2.6%	

Table 1: Statistics of the PTB and WikiText-2.

	PTB				WikiText-2			
Model	**Hidden size**	**Layers**	**Valid**	**Test**	**Hidden size**	**Layers**	**Valid**	**Test**
KN-5(Mikolov and Zweig 2012)	-	-	-	141.2	-	-	-	-
RNN(Mikolov and Zweig 2012)	300	1	-	124.7	-	-	-	-
LSTM(Zaremba, Sutskever, and Vinyals 2014)	200	2	120.7	114.5	-	-	-	-
LSTM(Grave, Joulin, and Usunier 2016)	1024	1	-	82.3	1024	1	-	99.3
LSTM(Merity et al. 2017)	650	2	84.4	80.6	650	2	108.7	100.9
RNN†	256	1	130.3	124.1	512	1	126.0	120.4
LSTM†	256	1	118.6	110.3	512	1	105.6	101.4
TSLM	256	1	**117.2**	**108.1**	512	1	**104.9**	**100.4**
RNN+MoS†(Yang et al. 2018)	256	1	88.7	84.3	512	1	85.6	81.8
TSLM+MoS	256	1	**86.4**	**83.6**	512	1	**83.9**	**81.0**

Table 2: Best perplexity of models on the PTB and WikiText-2 dataset. Models tagged with † indicate that they are reimplemented by ourselves.

Term Memory (LSTM) networks (Soutner and Mller 2013) achieve promising results on language model tasks.

Recently, Cohen et al. (2016) and Levine et al. (2017; 2018) use tensor analysis to explore the expressive power and interpretability of neural networks, including convolutional neural network (CNN) and RNN. Levine et al. (2018) even explored the connection between quantum entanglement and deep learning. Inspired by their work, Zhang et al. (2018) proposed a Quantum Many-body Wave Function inspired Language Modeling (QMWF-LM) approach. However, in QMWF-LM and TSLM, the term *language modeling* has different meanings and application tasks. Specifically, QMWF-LM is basically a *language representation* which encodes language features extracted by a CNN, and performs the semantic matching in Question Answer (QA) task as an *extrinsic evaluation*.

Different from QMWF-LM, in this paper, TSLM focuses on the Markov process of conditional probabilities in language modeling task with an *intrinsic evaluation*. Based on the tensor representation and tensor networks, we propose the tensor space language model. We have established the connection between TSLM and neural language models (e.g., RNN based LMs) and proved that TSLM is a more general language model.

Experiments

Datasets

PTB Penn Tree Bank dataset (Marcus, Marcinkiewicz, and Santorini 1993) is often used to evaluate language models. It consists of 929k training words, 73k validation words, 82k test words, and has 10k words in its vocabulary.

WikiText-2 (WT2) dataset (Merity et al. 2017). Compared with the preprocessed version of PTB, WikiText-2 is larger. It also features a larger vocabulary and retains the original case, punctuation and numbers, all of which are removed in PTB. It is composed of full articles.

Table 1 shows statistics of these two datasets. The out of vocabulary (OOV) rate denotes the percentage of tokens have been replaced by an $\langle unk \rangle$ token. The token count includes newlines which add to the WikiText-2 dataset.

Evaluation Metrics

Perplexity is the typical measure used for reporting progress in language modeling. It is the average per-word log-probability on the holdout data set.

$$\mathrm{PPL} = e^{(-\frac{1}{n}\sum_i \ln p(w_i))}$$

The lower the perplexity, the more effective the model is. We follow the standard procedure and sum over all the words.

Comparative Models and Experimental Settings

In order to demonstrate the effectiveness of TSLM, we compare our model with several baseline models, including Kneser-Ney 5-gram (KN-5) (Mikolov and Zweig 2012), RNN based language model (Mikolov et al. 2010), Long Short-Term Memory network (LSTM) based language model (Zaremba, Sutskever, and Vinyals 2014), and RNN added Matrix of Softmax (MoS) language model (Yang et al. 2018). Models tagged with † indicate that they are reimplemented by ourselves.

Kneser-Ney 5-gram (KN-5): It uses Kneser-Ney Smoothing on the n-gram language model (n=5) (Chen and Goodman 1996; Mikolov and Zweig 2012). It is also the most representative statistical language model, and we consider it as a low-order tensor language model.

RNN: Recurrent neural network based language models (Mikolov et al. 2010), use a recurrent hidden layer to represent longer and variable length histories, instead of using fixed number of words to represent the context.

LSTM: LSTM neural network is an another variant of RNN structure. It allows to discover both long and short patterns in data and eliminates the problem of vanishing gradient by training RNN. LSTM approved themselves in various applications and it seems to be very promising course also for the field of language modeling (Soutner and Mller 2013).

RNN+MoS: A high-rank RNN language model (Yang et al. 2018) breaking the softmax bottleneck, formulates the next-token probability distribution as a Matrix of Softmax (MoS), and improves the perplexities on the PTB and WikiText-2. It is the state-of-the-art softmax technique for solving probability distributions.

Since our focus is to show the effectiveness of language modeling in tensor space, we choose to set a relatively small scale network structure. Specifically, we set the same scale parameters for comparison experiments, i.e. 256/512 hidden size, 1 hidden layer, 20 batch size and 30/40 sequence length. Among them, the hidden size is equivalent to the tensor decomposition rank r, and sequence length means the tensor order n in our model.

Experimental Results and Analysis

Table 2 shows the results on PTB and WT2, respectively. From Table 2, we could observe that our proposed TSLM achieves the lower perplexity, which reflects that TSLM outperform others. It is obvious that KN-5 method gets the worst performance. We choose KN-5 as a baseline, since it is a typical n-gram model and we prove that TSLM is a generalization of n-gram. Although the most advanced Kneser-Ney Smoothing is used, it still has not achieved good performance. The reason could be that statistic language model is based on the word frequency statistics and does not have the semantic advantages that word vectors in continuous space can satisfy.

The LSTM based language modeling can theoretically model arbitrarily long dependencies. The experimental results show that our model has achieved relatively better results than LSTM reimplemented by ourselves. Based on the MoS, RNN based language models achieve state-of-the-art results. To prove the effectiveness of our model using MoS, we compared our model to RNN+MoS model on the same parameters. The empirical results of our model have also been improved, which also illustrates the effectiveness of our model structure.

We have shown that TSLM is a generalized language model. In practice, its performance is better than RNN based language model. The reason could be that the non-linear operation g in Eq. 20 is an element-wise multiplication, which is capable of expressing more complex semantic information than a standard RNN structure with activation function using an addition operation.

Note that, the parameters n, m and r are crucial factors for modeling language in TSLM. The order n of the tensor reflects the maximum sentence length. With the increase of the maximum sentence length, the performance of the model gradually increases. However, after a certain degree of growth, the expressive ability of the TSLM will reach an upper bound. The reason can be summarized as: The size of the corpus determines the size of the tensor space we need to

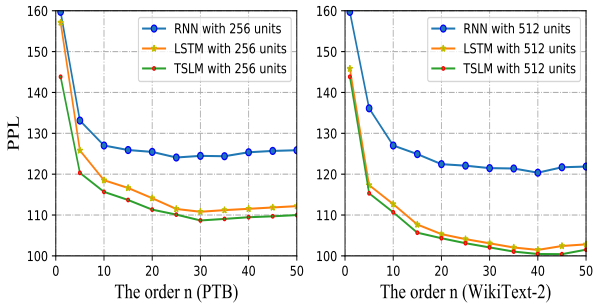

Figure 4: Perplexity (PPL) with different max length of sentences in corpus.

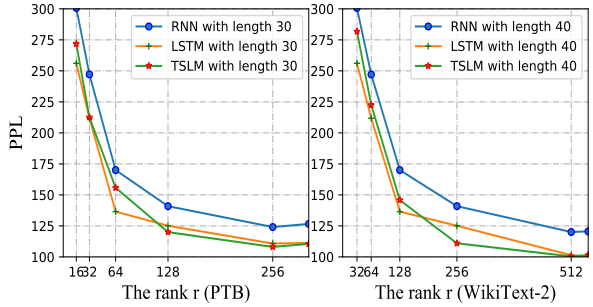

Figure 5: Perplexity (PPL) with different hidden sizes.

model. The larger the order is, the larger the semantic space that TSLM can model and the more semantic information it can contain. As shown in Fig. 4, we can see that the model is optimal when n equals 30 and 40 on PTB and WikiText datasets, respectively. There are similar experimental phenomena in RNN and LSTM.

Other key factors that affect the capability of TSLM are the dimension of the orthogonal basis m and the rank of tensor decomposition r, where each orthogonal basis denote the basic semantic meaning. The tensor decomposition is enough to extract the main features of the tensor $\mathcal{T}$ when $r = m$. They correspond to the word embedding size and hidden size in RNN or LSTM, and are usually set as the same value. We try to set the value of them as $[16, 32, 64, 128, 256, 512, ...]$. As shown in Fig. 5, we can see that the model is optimal when the decomposition rank r equals 256 and 512 on PTB and WikiText datasets, respectively. These phenomena is mainly due to the saturation of semantic information in tensor space, which means that the number of the basic semantic meaning is finite with respect to the specific corpus.

Conclusions and Future work

In this paper, we have proposed a language model based on tensor space, named Tensor Space Language Model (TSLM). We use the tensor networks to represent the high-order tensor for modeling language, and calculate the probability of a specific sentence by the inner product of tensors. Moreover, we have proved that TSLM is a generalization of n-gram language model and we can derive the recursive

language model process from TSLM. Experimental results demonstrate the effectiveness of our model, compared with the standard RNN-based language model and LSTM-based language model on two typical datasets to evaluate language modeling. In the future, we will further explore the potential of tensor network for modeling language in both theoretical and empirical directions.

Acknowledgement

This work is supported in part by the state key development program of China (grant No. 2017YFE0111900), Natural Science Foundation of China (grant No. U1636203, 61772363), and the European Unions Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 721321.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Janvin, C. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Brown, P. F.; Desouza, P. V.; Mercer, R. L.; Pietra, V. J. D.; and Lai, J. C. 1992. Class-based n -gram models of natural language. *Computational Linguistics* 18(4):467–479.

Cai, D.; He, X.; and Han, J. 2006. Tensor space model for document analysis. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 625–626. ACM.

Chen, S. F., and Goodman, J. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, 310–318. Association for Computational Linguistics.

Cohen, N.; Sharir, O.; and Shashua, A. 2016. On the expressive power of deep learning: A tensor analysis. *Computer Science*.

Grave, E.; Joulin, A.; and Usunier, N. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Hitchcock, F. L. 1927. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics* 6(1-4):164189.

Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Kneser, R., and Ney, H. 2002. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing*, 181–184 vol.1.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *Siam Review* 51(3):455–500.

Levine, Y.; Yakira, D.; Cohen, N.; and Shashua, A. 2018. Deep learning and quantum entanglement: Fundamental connections with implications to network design. In *International Conference on Learning Representations*.

Levine, Y.; Sharir, O.; and Shashua, A. 2017. Benefits of depth for long-term memory of recurrent networks. *arXiv preprint arXiv:1710.09431*.

Liu, N.; Zhang, B.; Yan, J.; and Chen, Z. 2005. Text representation: from vector to tensor. In *IEEE International Conference on Data Mining*, 725–728.

Lopez, A. 2008. Statistical machine translation. *Acm Computing Surveys* 40(3):1–49.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. *Building a large annotated corpus of English: the penn treebank*. MIT Press.

Merity, S.; Xiong, C.; Bradbury, J.; Socher, R.; Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer sentinel mixture models. In *ICLR*.

Mikolov, T., and Zweig, G. 2012. Context dependent recurrent neural network language model. *SLT* 12:234–239.

Mikolov, T.; Karafit, M.; Burget, L.; Cernock, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September*, 1045–1048.

Oseledets, I. V. 2011. Tensor-train decomposition. *Siam Journal on Scientific Computing* 33(5):2295–2317.

Pellionisz, A., and Llins, R. 1980. Tensorial approach to the geometry of brain function: Cerebellar coordination via a metric tensor. *Neuroscience* 5(7):1125–1136.

Soutner, D., and Mller, L. 2013. Application of lstm neural networks in language modelling. In *International Conference on Text, Speech and Dialogue*, 105–112.

Sutskever, I.; Martens, J.; and Hinton, G. E. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1017–1024.

Wang, C.; Chung, G.; and Seneff, S. 2006. Automatic induction of language model data for a spoken dialogue system. *Language Resources and Evaluation* 40(1):25–46.

Wu, Y.; Zhang, S.; Zhang, Y.; Bengio, Y.; and Salakhutdinov, R. R. 2016. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2856–2864.

Yang, Z.; Dai, Z.; Salakhutdinov, R.; and Cohen, W. W. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Yu, D., and Deng, L. 2014. *Automatic Speech Recognition: A Deep Learning Approach*. Springer.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhang, P.; Su, Z.; Zhang, L.; Wang, B.; and Song, D. 2018. A quantum many-body wave function inspired language modeling approach. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1303–1312. ACM.

Appendix

Claim 1

In our TSLM, when we set the dimension of vector space $m = |\mathrm{V}|$ and each word w as an *one-hot* vector, the probability of sentence s consist of words $d_1, \ldots, d_n$ in vocabulary is the entry $\mathcal{T}_{d_1 \ldots d_n}$ of tensor $\mathcal{T}$.

Proof of Claim 1

Proof. In our TSLM, when we set the dimension of vector space $m = |\mathrm{V}|$ and each word w as an one-hot vector, the specific sentence s will be represented as an one-hot tensor. The mixed representation $\boldsymbol{c}$ can be regarded as the total sampling distribution. The tensor inner product $\langle \boldsymbol{s}, \boldsymbol{c} \rangle$ represents statistics probability that a sentence $\boldsymbol{s}$ appears in a language. Specifically, the word w_i is an one-hot vector $\boldsymbol{w}_i = (0, \ldots, 1, \ldots, 0)$, and the vectors of any two different words are orthogonal (word vector itself is basis vector):

$$\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle = \langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{22}$$

Firstly, for the sentence $s = (w_1, \ldots, w_n)$ with length n is represented as $\boldsymbol{s} = \boldsymbol{w}_1 \otimes \cdots \otimes \boldsymbol{w}_n$. It is an one-hot tensor, and the tensors of any two different sentences are orthogonal (when being viewed as the flatten vectors):

$$\begin{aligned} \boldsymbol{s}_i &= \boldsymbol{w}_{i,1} \otimes \cdots \otimes \boldsymbol{w}_{i,n} \\ \boldsymbol{s}_j &= \boldsymbol{w}_{j,1} \otimes \cdots \otimes \boldsymbol{w}_{j,n} \\ \Rightarrow \langle \boldsymbol{s}_i, \boldsymbol{s}_j \rangle &= \prod_{k=1}^{n} \langle \boldsymbol{w}_{i,k}, \boldsymbol{w}_{j,k} \rangle \\ &= \begin{cases} 1, & \boldsymbol{w}_{i,k} = \boldsymbol{w}_{j,k},\ \forall\, k \in [n] \\ 0, & \text{otherwise} \end{cases} \\ \Rightarrow \langle \boldsymbol{s}_i, \boldsymbol{c} \rangle &= \langle \boldsymbol{s}_i, \sum_j p_j \boldsymbol{s}_j \rangle = p_i \end{aligned} \tag{23}$$

Secondly, for the representations in TSLM, the orthogonal basis can be composed by $\{\boldsymbol{w}_d\}_{d=1}^{|\mathrm{V}|}$, and the sentence s will be represented as:

$$\boldsymbol{s} = \sum_{d_1, \ldots, d_n = 1}^{|\mathrm{V}|} \mathcal{A}_{d_1 \ldots d_n} \boldsymbol{w}_{d_1} \otimes \cdots \otimes \boldsymbol{w}_{d_n} \tag{24}$$

where

$$\mathcal{A}_{d_1 \ldots d_n} = \begin{cases} 1, & d_k = \mathrm{index}(w_k, \mathrm{V}), \forall\, k \in [n] \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

which means tensor $\mathcal{A}$ is an one-hot tensor, and $\mathrm{index}(w,\mathrm{V})$ means the index of w in vocabulary V. We have defined $\boldsymbol{c} := \sum p_i \boldsymbol{s}_i$ as :

$$\boldsymbol{c} = \sum_{d_1, \ldots, d_n = 1}^{|\mathrm{V}|} \mathcal{T}_{d_1 \ldots d_n} \boldsymbol{w}_{d_1} \otimes \cdots \otimes \boldsymbol{w}_{d_n} \tag{26}$$

Then, the probability of sentence s_i is:

$$\begin{aligned} p_i &= \langle \boldsymbol{s}_i, \boldsymbol{c} \rangle \\ &= \sum_{d_1, \ldots, d_n = 1}^{|\mathrm{V}|} \mathcal{T}_{d_1 \ldots d_n} \mathcal{A}_{d_1 \ldots d_n} \\ &= \mathcal{T}_{d_1 \ldots d_n},\ \ d_k = \mathrm{index}(w_k, \mathrm{V}), \forall\, k \in [n] \end{aligned} \tag{27}$$

Therefore, the probability of sentence s consist of words $d_1, \ldots, d_n$ is $p(w_1 \ldots w_n = d_1 \ldots d_n) = \mathcal{T}_{d_1 \ldots d_n}$. □

Claim 2

In our TSLM, we define the word sequence $w_1^i = (w_1, w_2, \ldots, w_i)$ with length i as:

$$\boldsymbol{w}_1^i := \boldsymbol{w}_1 \otimes \cdots \otimes \boldsymbol{w}_i \otimes \mathbf{1}_{i+1} \otimes \cdots \otimes \mathbf{1}_n \tag{28}$$

Then, the probability $p(w_1^i)$ can be computed as $p(w_1^i) = \langle \boldsymbol{w}_1^i, \boldsymbol{c} \rangle$.

Proof of Claim 2

Proof. According to the marginal distribution in probability theory and statistics, for two discrete random variables, the marginal probability function can be written as $p(X = x)$:

$$p(X = x) = \sum_y p(X = x, Y = y) \tag{29}$$

where $p(X = x, Y = y)$ is the joint distribution of two variables X and Y.

Firstly, in our TSLM, we can define the marginal distribution using the word variable w as:

$$\begin{aligned} p(w_i) &= \sum_{w_j \in \mathrm{V}} p(w_i, w_j) \\ p(w_1, \ldots, w_{n-1}) &= \sum_{w_n \in \mathrm{V}} p(w_1, \ldots, w_{n-1}, w_n) \end{aligned} \tag{30}$$

Secondly, the probability of the word sequence w_1^i can be written as:

$$\begin{aligned} & p(w_1^i) \\ =& p(w_1, \ldots, w_i) \\ =& \sum_{w_{i+1}, \cdots, w_n \in \mathrm{V}} p(w_1, \ldots, w_i, w_{i+1}, \ldots, w_n) \\ =& \sum_{d_{i+1}, \cdots, d_n = 1}^{|\mathrm{V}|} \mathcal{T}_{d_1 \ldots d_n},\ \ d_k = \mathrm{index}(w_k, \mathrm{V}), \forall\, k \in [i] \end{aligned} \tag{31}$$

Then, the inner product $\langle \boldsymbol{w}_1^i, \boldsymbol{c} \rangle$ can be written as:

$$\begin{aligned} & \langle \boldsymbol{w}_1^i, \boldsymbol{c} \rangle \\ =& \langle \boldsymbol{w}_1 \otimes \cdots \otimes \mathbf{1}, \sum_{d_1, \cdots, d_n = 1}^{|\mathrm{V}|} \mathcal{T}_{d_1 \ldots d_n} \boldsymbol{w}_{d_1} \otimes \cdots \otimes \boldsymbol{w}_{d_n} \rangle \\ =& \sum_{d_1, \cdots, d_n = 1}^{|\mathrm{V}|} \mathcal{T}_{d_1 \ldots d_n} \langle \boldsymbol{w}_1 \otimes \cdots \otimes \mathbf{1}, \boldsymbol{w}_{d_1} \otimes \cdots \otimes \boldsymbol{w}_{d_n} \rangle \\ =& \sum_{d_1, \cdots, d_n = 1}^{|\mathrm{V}|} \mathcal{T}_{d_1 \ldots d_n} \prod_{j=1}^{i} \langle \boldsymbol{w}_j, \boldsymbol{w}_{d_j} \rangle \prod_{j=i+1}^{n} \langle \mathbf{1}, \boldsymbol{w}_{d_j} \rangle \\ =& \sum_{d_{i+1}, \cdots, d_n = 1}^{|\mathrm{V}|} \mathcal{T}_{d_1 \ldots d_n},\ \ d_k = \mathrm{index}(w_k, \mathrm{V}), \forall\, k \in [i] \end{aligned} \tag{32}$$

Therefore, we derive that the probability $p(w_1^i)$ can be computed as $p(w_1^i) = \langle \boldsymbol{w}_1^i, \boldsymbol{c} \rangle$ according to Eq. 31 and 32. □