Probabilistic Transformer: A Probabilistic Dependency Model for Contextual Word Representation

Haoyi Wu and **Kewei Tu***
School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
{wuhy1, tukw}@shanghaitech.edu.cn

Abstract

Syntactic structures used to play a vital role in natural language processing (NLP), but since the deep learning revolution, NLP has been gradually dominated by neural models that do not consider syntactic structures in their design. One vastly successful class of neural models is transformers. When used as an encoder, a transformer produces contextual representation of words in the input sentence. In this work, we propose a new model of contextual word representation, not from a neural perspective, but from a purely syntactic and probabilistic perspective. Specifically, we design a conditional random field that models discrete latent representations of all words in a sentence as well as dependency arcs between them; and we use mean field variational inference for approximate inference. Strikingly, we find that the computation graph of our model resembles transformers, with correspondences between dependencies and self-attention and between distributions over latent representations and contextual embeddings of words. Experiments show that our model performs competitively to transformers on small to medium sized datasets. We hope that our work could help bridge the gap between traditional syntactic and probabilistic approaches and cutting-edge neural approaches to NLP, and inspire more linguistically-principled neural approaches in the future.[1]

1 Introduction

Once upon a time, syntactic structures were deemed essential in natural language processing (NLP). Modeling and inference about syntactic structures was an indispensable component in many NLP systems. That has all changed since the deep learning revolution started a decade ago. Modern NLP predominantly employs various neural models, most of which do not consider syntactic structures in their design.

One type of neural models that are particularly successful is transformers (Vaswani et al., 2017). Given an input text, a transformer produces a vector representation for each word that captures the meaning as well as other properties of the word in its context. Such contextual word representations can then be served into downstream neural networks for solving various NLP tasks. The power of transformers in producing high-quality contextual word representations is further unleashed with large-scale pretraining (Devlin et al., 2019; Liu et al., 2020). Nowadays, a vast majority of NLP models and systems are built on top of contextual word representations produced by some variants of pretrained transformers.

Like most other neural models, transformers were developed based on human insight and trial and error, without explicit design for incorporating syntactic structures. Nevertheless, there is evidence that contextual word representations produced by pretrained transformers encode certain syntactic structures (Hewitt and Manning, 2019; Tenney et al., 2019) and attention heads in pretrained transformers may reflect syntactic dependencies (Clark et al., 2019; Htut et al., 2019; Ravishankar et al., 2021). Because of the heuristic nature of the transformer model design, exactly how transformers acquire such syntactic capability remains unclear.

In this paper, we propose *probabilistic transformers*, a very different approach to deriving contextual word representations that is based on classic non-neural probabilistic modeling with innate syntactic components. Specifically, we design a conditional random field that models discrete latent representations of all words as well as a syntactic dependency structure of the input sentence, and we define a potential function which evaluates the compatibility of the latent representations of any pair of words

*Corresponding author.
[1]Our code is publicly available at https://github.com/whyNLP/Probabilistic-Transformer

connected by a dependency arc. We use mean field variational inference for approximate inference, producing a marginal distribution for each latent word representation, the probability vector of which can then be used as a contextual vector representation of the word.

While we propose our model from a purely syntactic and probabilistic perspective that is unrelated to transformers, we show that there is a striking resemblance between the computation graph of the inference procedure of our model and that of a transformer, with our intermediate distributions over dependency heads corresponding to self-attention scores and our intermediate distributions over latent word representations corresponding to intermediate word embeddings in a transformer. In short, we start with a probabilistic syntactic model but reach the transformer! We empirically compare our model with transformers when trained with either masked language modeling or downstream tasks. Our experimental results show that our model performs competitively to transformers on small to medium sized datasets.

We hope that probabilistic transformers, instead of being a replacement of transformers, could benefit the analysis of the syntactic capability of transformers and at the same time inspire novel extensions of transformers. Furthermore, we hope our work would promote future research of neural models that are linguistically more principled, theoretically more well-founded, and empirically no less powerful than existing models.

2 Probabilistic Transformers

We will first introduce the basic model, a conditional random field (CRF) as illustrated in Figure 1, then show the inference procedure, and finally introduce some variants to the basic model.

2.1 The CRF Model

Given a sentence (a sequence of words), denote n as the sequence length. For the i-th word, we define Z_i as a discrete latent label that represents the syntactic (and possibly semantic) property of the word in the sentence (i.e., it is a contextual representation) with a label set of size d. Such a discrete representation deviates from the common practice of representing a word with a continuous vector, but it is sufficient at least for syntactic processing (Kitaev et al., 2022) and it greatly simplifies our probabilistic model. For the i-th word, we also

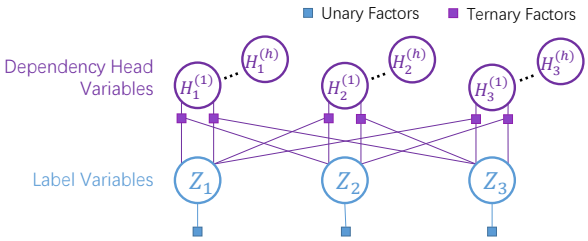


Figure 1: The factor graph for our CRF model with $n = 3$. For clarity, ternary factors that connect to $H_i^{(c)}$ with $c > 1$ are not shown in the figure.

define $H_i \in \{1, 2, \cdots, n\}$ representing the syntactic dependency head of the word. So the set of variables $\{H_i\}_{i=1}^n$ specifies a dependency structure. We may also allow H_i to point to a dummy root node, which will be discussed in Section 2.3.5. We follow the head-selection paradigm of dependency parsing and do not enforce the tree constraint, which again simplifies our model design.

Next, we define two types of potential functions. For the i-th word w_i, we define a unary potential function (corresponding to the unary factors in Figure 1) evaluating the compatibility of the word and its label Z_i:

$$\phi_u(Z_i) = \exp\left(\mathbf{S}_{w_i, Z_i}\right) \tag{1}$$

where $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is a score matrix, $|\mathcal{V}|$ is the size of the vocabulary. For simplicity, we do not exploit any morphological or contextual features for computing the scores. For every pair of words w_i and w_j ($i \neq j$), we define a ternary potential function (corresponding to the ternary factors in Figure 1) over Z_i, Z_j and H_i, which evaluates the compatibility between the labels of the two words if w_j is the dependency head of w_i:

$$\phi_t(H_i, Z_i, Z_j) = \begin{cases} \exp\left(\mathbf{T}_{Z_i, Z_j}\right) & H_i = j \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{T} \in \mathbb{R}^{d \times d}$ is a score matrix.

Inspired by the multi-head structure in transformers, we allow multiple dependency structures for the same sentence, which may represent different flavors of dependencies. Each dependency structure resides in a different *channel* with its own dependency head variables and ternary potential functions. For the c-th channel, we denote the set of dependency head variables by $\{H_i^{(c)}\}_{i=1}^n$ and the score matrix of the ternary potential function

by $\mathbf{T}^{(c)}$. Let h denote the total number of channels. We may stack all the score matrices $\mathbf{T}^{(c)}$ for $c = 1, \cdots, h$ to form a score tensor $\mathbf{T} \in \mathbb{R}^{d \times d \times h}$. Note that all the channels share the same set of latent label variables $\{Z_i\}_{i=1}^n$.

2.2 Inference

Following Wang and Tu (2020), we use Mean Field Variational Inference (MFVI) to perform approximate inference. Different from the previous work, however, we need to run inference over latent labels in addition to dependency heads.

MFVI iteratively passes messages between random variables and computes an approximate posterior marginal distribution for each random variable (denoted by $Q(\cdot)$). Let $\mathcal{F}_{ic}^{(t)}$ denote the message received by variable $H_i^{(c)}$ at time step t from ternary factors, and $\mathcal{G}_i^{(t)}$ denote the message received by variable Z_i at time step t from ternary factors. We have

$$\mathcal{F}_{ic}^{(t)}(j) = \sum_a \sum_b \left(Q_i^{(t)}(a) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)} \right) \tag{3}$$

$$\begin{aligned} \mathcal{G}_i^{(t)}(a) = \sum_c \sum_{j \neq i} \sum_b \Big(& Q_{ic}^{(t)}(j) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)} \\ & + Q_{jc}^{(t)}(i) Q_j^{(t)}(b) \mathbf{T}_{b,a}^{(c)} \Big) \end{aligned} \tag{4}$$

where

$$Q_i^{(t)}(a) \propto \exp\left(\mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a) \right) \tag{5}$$

$$Q_{ic}^{(t)}(j) \propto \exp\left(\mathcal{F}_{ic}^{(t-1)}(j) \right) \tag{6}$$

are the approximate marginal distributions at time step t, with $Q_i^{(t)}(\cdot)$ over Z_i and $Q_{ic}^{(t)}(\cdot)$ over $H_i^{(c)}$. We initialize these distributions by

$$Q_i^{(0)}(a) \propto \exp\left(\mathbf{S}_{w_i,a} \right) \tag{7}$$

$$Q_{ic}^{(0)}(j) \propto 1 \tag{8}$$

After a fixed number of $T > 0$ iterations, we obtain the final posterior marginal distribution $Q_i^{(T)}(Z_i)$ for $i = 1, \cdots, n$. Resulted from interactions with all the words of the sentence, the distribution $Q_i^{(T)}(Z_i)$ incorporates information of not only the i-th word, but also its context. Therefore, we can treat the probability vector of this distribution as a contextual vector representation for the i-th word. In practice, we find that using unnormalized scores in log space as contextual word representations produces better results, i.e., we skip exponentiation and normalization when computing $Q_i^{(T)}(Z_i)$ using Equation 5 during the final iteration.

Since all the computation during MFVI is fully differentiable, we can regard the corresponding computation graph as a recurrent or graph neural network parameterized with score matrix $\mathbf{S}$ and tensor $\mathbf{T}$. We can use the contextual word representations for downstream tasks by connecting the network to any downstream task-specific network, and we can update the model parameters using any task-specific learning objective through gradient descent. This is exactly the same as how transformers are used.

2.3 Extensions and Variants

We introduce a few extensions and variants to the basic model that are empirically beneficial. Additional variants are discussed in Appendix B.

2.3.1 Distance

Similar to the case of transformers, our probabilistic model is insensitive to the word order of the input sentence. In order to capture the order information, we apply relative positional encoding to our model by using distance-sensitive ternary potential functions. Specifically, we use different ternary scores for different distances between words denoted by the two Z variables of the potential function. The ternary potential function in Equation 2 becomes:

$$\phi_t(H_i^{(c)}, Z_i, Z_j) = \begin{cases} \exp\left(\mathbf{T}[f(i-j)]_{Z_i,Z_j}^{(c)} \right) & H_i^{(c)} = j \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

where f is a clip function with threshold γ:

$$f(x) = \begin{cases} 0 & x < -\gamma \\ x + \gamma + 1 & -\gamma \leq x < 0 \\ x + \gamma & 0 < x \leq \gamma \\ 2\gamma + 1 & x > \gamma \end{cases} \tag{10}$$

Notice that x cannot be zero since the head of a word cannot be itself. We set $\gamma = 3$ by default.

2.3.2 Asynchronous Update

During inference of the basic model, we iteratively update all variables in a synchronous manner. This can be problematic. Consider the first iteration. The messages passed to Z variables from H variables do not contain meaningful information because the initial distributions over H are uniform.

Consequently, after one iteration, distributions over all Z variables become almost identical.

To fix this problem, we use the asynchronous update strategy by default in this work. For each iteration, we first update distributions over H variables, and then update distributions over Z variables based on the updated distributions over H variables. Formally, we rewrite Formula 6 as

$$Q_{ic}^{(t)}(j) \propto \exp\left(\mathcal{F}_{ic}^{(t)}(j)\right)$$

and eliminate Formula 8 because distributions over H variables no longer need initialization.

2.3.3 Message Weight

During inference, H variables have much fewer message sources than Z variables. This often pushes H variables towards being uniformly distributed. To balance the magnitude of the messages, we follow the Entropic Frank-Wolfe algorithm (Lê-Huu and Alahari, 2021), a generalization of MFVI, and introduce weight $\lambda_Z > 0$ and $\lambda_H > 0$ to Equation 5 and 6:

$$Q_i^{(t)}(a) \propto \exp\left(\frac{1}{\lambda_Z}\left(\mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a)\right)\right) \tag{11}$$

$$Q_{ic}^{(t)}(j) \propto \exp\left(\frac{1}{\lambda_H}\mathcal{F}_{ic}^{(t-1)}(j)\right) \tag{12}$$

We set $\lambda_Z = 1$ and $\lambda_H = \frac{1}{d}$ by default[2].

2.3.4 Tensor Decomposition

Ternary score $\mathbf{T}$ is a tensor of shape $d \times d \times h$. Since d is usually set to several hundred, such a tensor leads to a huge number of parameters. To reduce the number of parameters, we apply the Kruskal form (which is closely related to tensor rank decomposition) to build the ternary score from smaller tensors.

$$\mathbf{T}_{a,b}^{(c)} = \sum_{l=1}^{r} \mathbf{U}_{a,l} \cdot \mathbf{V}_{b,l} \cdot \mathbf{W}_{c,l} \tag{13}$$

where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d \times r}$ and $\mathbf{W} \in \mathbb{R}^{h \times r}$.

Since the number of channels h is relatively small, we may also choose only to decompose the first two dimensions.

$$\mathbf{T}_{a,b}^{(c)} = \sum_{l=1}^{r} \mathbf{U}_{a,c,l} \cdot \mathbf{V}_{b,c,l} \tag{14}$$

where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d \times h \times r}$.

[2]We choose these weights in a similar way to choosing the scaling factor in scaled dot-product attention of transformers. See more details in Appendix A.5.

2.3.5 Root Node

Dependency parsing assumes a dummy root node, which we may add to the CRF model. The root node is not associated with any word and instead can be seen as representing the entire sentence. Therefore, we assume that it has a different (and possibly larger) label set from words and hence requires a different ternary potential function. Specifically, we define Z_{ROOT} as a discrete latent label of the root node with a label set of size d_{root}. For $i \in \{1, 2, \cdots, n\}, c \in \{1, 2, \cdots, h\}$, we add a ternary potential function over $Z_i, H_i^{(c)}$ and Z_{ROOT}:

$$\phi_t(H_i^{(c)}, Z_i, Z_{ROOT}) = \begin{cases} \exp\left(\mathbf{T}'^{(c)}_{Z_i, Z_{ROOT}}\right) & H_i^{(c)} = ROOT \\ 1 & \text{otherwise} \end{cases}$$

where $\mathbf{T}' \in \mathbb{R}^{d \times d_{root} \times h}$ is the root score tensor. During inference, we initialize $Q^{(0)}(Z_{ROOT})$ with a uniform distribution. After inference, we can regard the posterior marginal distribution of Z_{ROOT} as a sentence representation.

3 Comparison with Transformers

Although our probabilistic transformers are derived as a probabilistic model of dependency structures over latent word labels, we find that its computational process has lots of similarities to that of transformers. Below, we first re-formulate a probabilistic transformer in a tensor form to facilitate its comparison with a transformer, and then discuss the similarities between the two models at three levels.

3.1 Probabilistic Transformers in Tensor Form

Consider a probabilistic transformer using a distance-insensitive ternary potential function without a dummy root node. We tensorize the update formulas in the inference process of probabilistic transformers. Suppose $Q_z^{(t)} \in \mathbb{R}^{n \times d}$ is a tensor that represents the posterior distributions of all the Z variables, and $Q_{h,c}^{(t)} \in \mathbb{R}^{n \times n}$ is a tensor that represents the posterior distributions of all the H variables in channel c (with a zero diagonal to rule out self-heading). We can rewrite Equation 3 and 4

as

$$\mathcal{F}_c^{(t)} = Q_z^{(t)} \mathbf{T}^{(c)} Q_z^{(t)T} \tag{15}$$

$$\mathcal{G}^{(t)} = \sum_c \left(Q_{h,c}^{(t)} Q_z^{(t)} \mathbf{T}^{(c)T} + Q_{h,c}^{(t)T} Q_z^{(t)} \mathbf{T}^{(c)} \right) \tag{16}$$

where

$$Q_z^{(t)} = \sigma(\mathbf{S} + \mathcal{G}^{(t-1)}) \tag{17}$$

$$Q_{h,c}^{(t)} = \sigma(\mathcal{F}_c^{(t-1)}) \tag{18}$$

and σ is the softmax function. We still set λ_Z to its default value 1 but regard λ_H as a hyperparameter. With asynchronous update, Equation 18 becomes:

$$Q_{h,c}^{(t)} = \sigma\left(\frac{\mathcal{F}_c^{(t)}}{\lambda_H} \right) \tag{19}$$

We assume that $\mathbf{T}^{(c)}$ is symmetric for $c = 1, \cdots, h$. This is the only assumption that we make in this section beyond the original definition from the previous section. Symmetric score matrices indicate that the ternary factors are insensitive to the head-child order, which is related to undirected dependency parsing (Sleator and Temperley, 1993). If $\mathbf{T}^{(c)}$ is symmetric, then $Q_{h,c}^{(t)}$ is also symmetric based on Formula 15 and 19. Thus, we can simplify Equation 16 to

$$\mathcal{G}^{(t)} = 2 \sum_c Q_{h,c}^{(t)} Q_z^{(t)} \mathbf{T}^{(c)T} \tag{20}$$

Suppose we decompose the ternary score tensor into two tensors $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d \times h \times r}$ according to Equation 14, which can be rewritten as:

$$\mathbf{T}^{(c)} = \mathbf{U}^{(c)} \mathbf{V}^{(c)T} \tag{21}$$

where $\mathbf{U}^{(c)}, \mathbf{V}^{(c)} \in \mathbb{R}^{d \times r}$ are the c-th channel of tensor $\mathbf{U}$ and $\mathbf{V}$ respectively. Substitute 21 into 15 and 20, we have

$$\mathcal{F}_c^{(t)} = Q_z^{(t)} \mathbf{U}^{(c)} \mathbf{V}^{(c)T} Q_z^{(t)T} \tag{22}$$

$$\mathcal{G}^{(t)} = 2 \sum_c Q_{h,c}^{(t)} Q_z^{(t)} \mathbf{V}^{(c)} \mathbf{U}^{(c)T} \tag{23}$$

We define

$$Q_c = Q_z^{(t-1)} \mathbf{U}^{(c)} \tag{24}$$

$$K_c = V_c = Q_z^{(t-1)} \mathbf{V}^{(c)} \tag{25}$$

For time step $t-1$, we could rewrite Formula 22 and 23 as

$$\mathcal{F}_c^{(t-1)} = Q_c K_c^T \tag{26}$$

$$\mathcal{G}^{(t-1)} = 2 \sum_c Q_{h,c}^{(t-1)} V_c \mathbf{U}^{(c)T} \tag{27}$$

Apply Equation 27, 19, 26 to 17, we have

$$Q_z^{(t)} = \sigma(\mathbf{S} + 2 \sum_c \text{channel}_c \, \mathbf{U}^{(c)T}) \tag{28}$$

where

$$\text{channel}_c = \sigma\left(\frac{Q_c K_c^T}{\lambda_H} \right) V_c \tag{29}$$

We call the computation of channel_c a *single-channel update* for channel c.

Now we have a tensorized formulation of the computation in probabilistic transformers and we are ready for its comparison with transformers at three different levels.

3.2 Single-Channel Update vs. Scaled Dot-Product Attention

Scaled dot-product attention in transformers is formulated as:

$$\text{Attention}(Q, K, V) = \sigma\left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

As we can see, our single-channel update in Equation 29 is almost identical to scaled dot-product attention in transformers. The only difference is that the diagonal of the tensor $Q_c K_c^T$ is zero in our model because the head of a word cannot be itself.

3.3 Multi-Channel Update vs. Multi-Head Attention

Multi-head attention in transformers is formulated as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_h\right) W^O$$

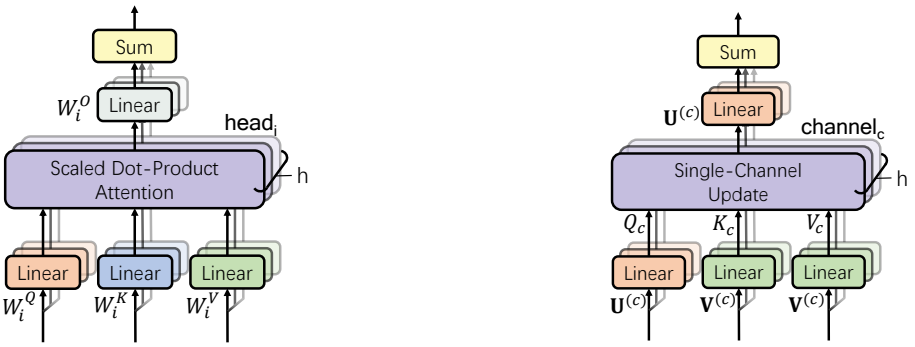
where

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

It is equivalent to

$$\text{MultiHead}(Q, K, V) = \sum_i \text{head}_i (W_i^O)^T$$

where $W^O \equiv \text{Concat}(W_1^O, \ldots, W_h^O)$ and $W_i^Q, W_i^K, W_i^V, W_i^O \in \mathbb{R}^{d \times r}$. Our multi-channel



(a) Multi-Head Attention

(b) Multi-Channel Update

Figure 2: Computation graphs for multi-head attention in transformers and multi-channel update in probabilistic transformers. See an explanation of replacing concat+linear with linear+sum in the upper part of multi-head attention in Section 3.3.

update formula (the second term within the softmax function in Equation 28) is similar to the multi-head attention in transformers, as shown in Figure 2. The main difference is that probabilistic transformers use the same parameters for W^K and W^V (both are $\mathbf{V}$, shown in green color in Figure 2b) and for W^Q and W^O (both are $\mathbf{U}$, shown in orange color in Figure 2b).

Recall that $\mathbf{U}$ and $\mathbf{V}$ are obtained from matrix decomposition (Equation 14). Therefore, the correspondence between $\mathbf{U}$, $\mathbf{V}$ and W^Q, W^K, W^O, W^V in transformers suggests that the latter can also be seen as derived from tensor decomposition. Previous work on transformers has the same findings (Elhage et al., 2021).

3.4 Full Model Comparison

Figure 3 compares the full computation graphs of the two models, which have a similar overall structure that repeats a module recurrently until outputting contextual word representations. Within the module, we have also established the correspondence between multi-channel update and multi-head attention. On the other hand, there are a few interesting differences.

First, our model does not have a feed-forward structure as in a transformer. However, we do propose a variant of our model that contains global variables representing topics (Appendix B.3), which may have similar functionality to the feed-forward structure.

Second, our model does not have residual connections or layer norms. Instead, it adds the initial distributions (unary scores) to the updated message at each iteration. This may replace the functionality of residual connections and may even make more sense when the downstream task strongly depends on the original word information.

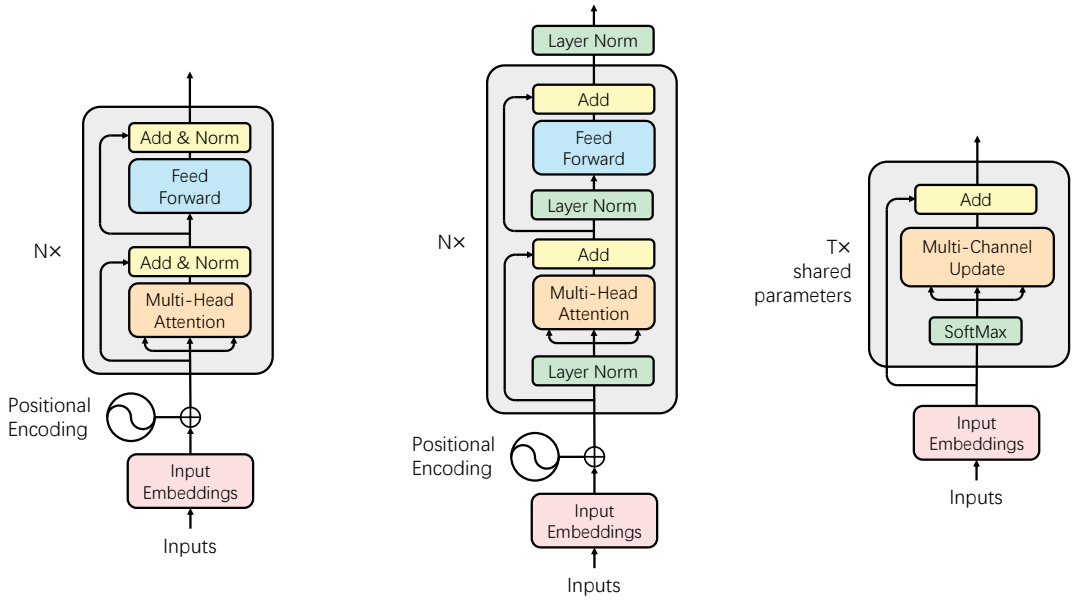
Third, we have an additional softmax in each iteration. Note that we do softmax before the first iteration (Equation 7) and also at the end of each iteration (Equation 28), but bypass it in the last iteration when producing the output word representations, so our model could be equivalently formulated as doing softmax before each iteration, which we show in Figure 3c. Doing softmax in this way is similar to the layer norm in pre-LN transformers (Xiong et al., 2020) (Figure 3b).

Finally, our model shares parameters in all iterations. This is similar to some variants of transformers that share parameters between layers, such as Universal Transformer (Dehghani et al., 2019) and ALBERT (Lan et al., 2019).

One consequence of these differences is that probabilistic transformers have much fewer parameters than transformers with the same number of layers, heads and embedding dimensions, because of shared parameters between iterations, absence of a feed-forward structure, and tied parameter matrices in multi-channel updates.

4 Experiments

We empirically compare probabilistic transformers with transformers on three tasks: masked language modeling, sequence labeling, and text classification. For each task, we use two different datasets. We also perform a syntactic test to evaluate the compositional generalization ability of our model.



(a) Transformer (b) Pre-LN Transformer (c) Probabilistic Transformer

Figure 3: Computation graphs for transformers and probabilistic transformers.

4.1 Tasks and Datasets

Here we briefly introduce our tasks and datasets. A detailed description is presented in Appendix D.

Masked Language Modeling (MLM). We perform MLM tasks on two corpora: the Penn TreeBank (PTB) (Marcus et al., 1993) and Brown Laboratory for Linguistic Information Processing (BLLIP) (Charniak et al., 2000). Following Shen et al. (2022), we randomly replace words with a mask token <mask> at a rate of 30%. The performance of MLM is evaluated by measuring perplexity (lower is better) on masked words.

We project the final word representation of each mask token to the vocabulary. For transformers, we tie the projection parameters to the initial word embeddings. We find that this trick improves the performance of transformers.

Sequence Labeling. For sequence labeling tasks, we perform part-of-speech (POS) tagging on two datasets: the Penn TreeBank (PTB) (Marcus et al., 1993) and the Universal Dependencies (UD) (De Marneffe et al., 2021). We also perform named entity recognition (NER) on CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003).

We directly project the final word representation of each word to the target tag set. For POS tagging, we evaluate the results by the accuracy of word-level predictions. For NER, we evaluate the results by measuring the F1 score of named entities.

Text Classification. We use the Stanford Sentiment Treebank (SST) (Socher et al., 2013) as the dataset. It has two variants: binary classification (SST-2) and fine-grained classification (SST-5).

For transformers, we add a <CLS> token at the front of the sentence and then project its representation to the tag set. For our model, we use the variant with a root node introduced in Section 2.3.5 and project the representation of the root node to the tag set.

Syntactic Test. To evaluate the compositional generalization abilities of our model, we perform a syntactic test on the COGS dataset (Kim and Linzen, 2020). We follow the settings in Ontanón et al. (2021), who cast the task as a sequence labeling task.

As in sequence labeling, we project word representations to tag sets. If all words in a sentence are correctly predicted, the sentence prediction will be counted as correct. We evaluate the results by the sentence-level accuracy of the predictions.

4.2 Settings

We tune transformers and our model separately for each task except the syntactic test. For the syntactic test, we find that both transformers and our model easily reach 100% accuracy on the validation set. This observation is consistent with Ontanón et al. (2021). Therefore, instead of tuning, we use the best-performed setting of transformers in Ontanón et al. (2021) for our experiments. The hyperparameters of our model are determined by their counter-

Task	Dataset	Metric	Transformer	Probabilistic Transformer
MLM	PTB	Perplexity	58.43 ± 0.58	62.86 ± 0.40
	BLLIP		101.91 ± 1.40	123.18 ± 1.50
POS	PTB	Accuracy	96.44 ± 0.04	96.29 ± 0.03
	UD		91.17 ± 0.11	90.96 ± 0.10
NER	CoNLL-2003	F1	74.02 ± 1.11	75.47 ± 0.35
CLS	SST-2	Accuracy	82.51 ± 0.26	82.04 ± 0.88
	SST-5		40.13 ± 1.09	42.77 ± 1.18
Syntactic Test	COGS	Sentence-level Accuracy	82.05 ± 2.18	84.60 ± 2.06

Table 1: Main results of probabilistic transformers compared with transformers.

parts of transformers based on the correspondence discussed in Section 3.

For our model, we integrate all the variants mentioned in Section 2.3 except the root node variant, which we only use for text classification tasks. We tune the tensor decomposition strategy on different tasks. For MLM tasks, we add a small L2 regularization term to the ternary scores in our model, which we experimentally find beneficial. We optimize both models using the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$.

4.3 Results

We report the average and standard deviation results of 5 random runs in Table 1. It shows that our model has a competitive performance compared with transformers. In most tasks, probabilistic transformers perform competitively to transformers. It is worth noting that in these experiments, probabilistic transformers have much fewer parameters than transformers. For most tasks, the number of parameters of our best model is about one-fifth to one-half of that of the best transformer.

We also conduct case studies of the dependency structures inferred by our model after training on downstream tasks. Similar to the case of self-attentions in transformers, the inferred dependency structures are only partially consistent with human intuition. See Appendix F for details.

5 Related Work

There have been several studies trying to incorporate syntactic structures to transformers. Strubell et al. (2018) force one attention head to attend to predicted syntactic governors of input tokens. Wang et al. (2019); Ahmad et al. (2021) try to integrate constituency or dependency structures into transformers. Shen et al. (2021) propose a dependency-constrained self-attention mechanism to induce dependency and constituency structures. Our work deviates from all these previous studies in that we start from scratch with probabilistic modeling of word representations and dependencies, but obtain a model that is strikingly similar to transformers.

6 Discussion

It is worth noting that in this work, our primary goal is not to propose and promote a new model to compete with transformers. Instead, it is our hope that our work could benefit the analysis and extension of transformers, as well as inspire future research of transformer-style models that are linguistically more principled, theoretically more well-founded, and empirically no less powerful than existing models. In the long run, we aim to bridge the gap between traditional statistical NLP and modern neural NLP, so that valuable ideas, techniques and insights developed over the past three decades in statistical NLP could find their place in modern NLP research and engineering.

The datasets used in our experiments have small to medium sizes (around 10k to 60k training sentences). Our preliminary experiments with MLM on larger data show that our models significantly underperform transformers, which suggests that our model may not be as scalable as transformers. One possible cause is the absence of a feed-forward structure in our model. Recent researches show that the feed-forward layers might serve as an important part of transformers (Dong et al., 2021). Further research is needed to analyze this problem.

Our model can be extended in a few directions. Instead of discrete labels, we may assume Z variables representing discrete vectors or even continuous vectors, which may lead to more complicated inference. We may model dependency labels by

pairing every H variable with a dependency label variable. While we focus on contextual word representation (i.e., encoding) in this paper, we may extend our probabilistic model to include a decoder. Considering the similarity between our model and transformers, we speculate that some of these extensions may be used to inspire extensions of transformers as well.

7 Conclusion

We present probabilistic transformers, a type of syntactic-aware probabilistic models for contextual word representation. A probabilistic transformer acquires discrete latent representations of all words in the input sentence by modeling a syntactic dependency structure of the input sentence. We use MFVI for approximate inference and find a striking resemblance between the computation graph of the inference procedure of our model and that of a transformer. Our experimental results demonstrate that our model performs competitively to transformers on small to medium sized datasets.

Limitations

Though we have found a tight connection between probabilistic transformers and transformers in Section 3, this does not mean that our model can be directly used to interpret or modify transformers. For instance, in Section 3.3, we find that W^K and W^V in transformers both correspond to $\mathbf{U}$ in probabilistic transformers. However, if we tie W^K and W^V in transformers, then we may observe a performance drop on some downstream tasks.

The performance of probabilistic transformers lags behind transformers on large datasets (>100k), which suggests that our model may not be as scalable as transformers. We have discussed this in Section 6.

The way of positional encoding for probabilistic transformers leads to slower training and inference speed. On masked language modeling tasks, our model is about 3 times slower than transformers with either absolute or relative positional encoding, though it has much fewer parameters than transformers.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61976139).

References

Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021. Gate: graph attention transformer encoder for cross-lingual relation and event extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12462–12470.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, and Mark Johnson. 2000. Bllip 1987–89 wsj corpus release 1, ldc no. *LDC2000T43. Linguistic Data Consortium*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Marie-Catherine De Marneffe, Christopher D Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal dependencies. *Computational linguistics*, 47(2):255–308.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Marco Dinarelli and Loïc Grobol. 2019. Seq2biseq: Bidirectional output-wise recurrent neural networks for sequence modelling. *arXiv preprint arXiv:1904.04733*.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2021. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2793–2803. PMLR.

N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2021. RealFormer: Transformer likes residual attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 929–943, Online. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv:1911.12246*.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev, Thomas Lu, and Dan Klein. 2022. Learned incremental representations for parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3086–3095, Dublin, Ireland. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, pages 1–16.

Đ Khuê Lê-Huu and Karteek Alahari. 2021. Regularized frank-wolfe for dense crfs: Generalizing mean field and beyond. *Advances in Neural Information Processing Systems*, 34:1453–1467.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach. In *International Conference on Learning Representations*, pages 1–15.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Tomáš Mikolov et al. 2012. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 80:26.

Santiago Ontanón, Joshua Ainslie, Vaclav Cvicek, and Zachary Fisher. 2021. Making transformers solve compositional tasks. *arXiv preprint arXiv:2108.04378*.

Vinit Ravishankar, Artur Kulmizev, Mostafa Abdou, Anders Søgaard, and Joakim Nivre. 2021. Attention can reflect syntactic structure (if you let it). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3031–3045, Online. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Yikang Shen, Shawn Tan, Alessandro Sordoni, Peng Li, Jie Zhou, and Aaron Courville. 2022. Unsupervised dependency graph network. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4767–4784, Dublin, Ireland. Association for Computational Linguistics.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021. StructFormer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages

7196–7209, Online. Association for Computational Linguistics.

Solomon Eyal Shimony. 1994. Finding maps for belief networks is np-hard. *Artificial intelligence*, 68(2):399–410.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*, pages 277–292, Tilburg, Netherlands and Durbuy, Belgium. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. 2019. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xinyu Wang and Kewei Tu. 2020. Second-order neural dependency parsing with message passing and end-to-end training. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 93–99, Suzhou, China. Association for Computational Linguistics.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

A Extended Entropic Frank-Wolfe

In Section 2.3.3, we add message weights to the update function of the posterior marginal distributions. It follows an extension of the Entropic Frank-Wolfe algorithm (Lê-Huu and Alahari, 2021), which is a generalization of MFVI. Below we briefly introduce the algorithm and our extension following most of the notations in their paper.

A.1 Entropic Frank-Wolfe

Suppose we want to minimize a continuous differentiable energy function $E(\cdot)$. Vanilla Frank-Wolfe solves the problem $\min_{\mathbf{x}\in\mathcal{X}} E(\mathbf{x})$ by starting from a feasible $\mathbf{x}^{(0)} \in \mathcal{X}$ at time step 0, and iterating the following steps:

$$\mathbf{p}^{(t)} \in \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\langle \nabla E\left(\mathbf{x}^{(t)}\right), \mathbf{p} \right\rangle$$
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha_t \left(\mathbf{p}^{(t)} - \mathbf{x}^{(t)}\right)$$

where $\alpha_t \in [0, 1]$ follows some stepsize scheme, $\mathcal{X}$ is the value range of $\mathbf{x}$, and here we let $\mathbf{x} \in \mathbb{R}^{n\times d}$ be the concatenation of the distributions over the label set of all variables in CRF.

Regularized Frank-Wolfe (Lê-Huu and Alahari, 2021) adds a regularization term $r(\cdot)$ to the objective. It solves the new objective $E(\mathbf{x}) + r(\mathbf{x})$ by iterating

$$\mathbf{p}^{(t)} \in \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\{ \left\langle \nabla E\left(\mathbf{x}^{(t)}\right), \mathbf{p} \right\rangle + r(\mathbf{p}) \right\}$$
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha_t \left(\mathbf{p}^{(t)} - \mathbf{x}^{(t)}\right)$$

It has been proved that regularized Frank-Wolfe achieves a sublinear rate of convergence $O(1/\sqrt{t})$ for suitable stepsize schemes.

Entropic Frank-Wolfe is a special case of regularized Frank-Wolfe, which sets the regularization term as an entropy function $r(\mathbf{x}) = -\lambda H(\mathbf{x})$,

where $H(\mathbf{x}) = -\sum_{i\in\mathcal{V}}\sum_{s\in\mathcal{S}} x_{is}\log x_{is}$, $\mathcal{S}$ is the label set of the variables, $\mathcal{V}$ is the set of indices of the variables. Entropy Frank-Wolfe has a closed-form solution for the update process

$$\begin{aligned}\mathbf{p}^{(t)} &= \underset{\mathbf{p}\in\mathcal{X}}{\operatorname{argmin}} \left\{ \left\langle \nabla E\left(\mathbf{x}^{(t)}\right), \mathbf{p}\right\rangle - \lambda H(\mathbf{p}) \right\} \\ &= \operatorname{softmax}\left(-\frac{1}{\lambda}\left(\nabla E\left(\mathbf{x}^{(t)}\right)\right)\right) \quad \forall t \geq 0\end{aligned} \tag{30}$$

When $\lambda = 1$ and $\alpha_t = 1, \forall t \geq 0$, it is the same as the mean field algorithm.

A.2 Extended Entropic Frank-Wolfe

We extend the Entropic Frank-Wolfe algorithm by using a more general regularization term

$$r(\mathbf{x}) = -\sum_{i\in\mathcal{V}} \lambda_i H(\mathbf{x}_i)$$

, where $\lambda_i > 0$ is the regularization weight of the i-th variable and $H(\mathbf{x}_i) = -\sum_{s\in\mathcal{S}} x_{is}\log x_{is}$ is the entropy of $\mathbf{x}_i$ over the probability simplex $\Delta = \left\{\mathbf{x}\in\mathbb{R}^d : \mathbf{x}\geq\mathbf{0}, \mathbf{1}^\top\mathbf{x} = 1\right\}$. It allows us to assign different regularization weights for different variables. We claim that the update function could be written as

$$\begin{aligned}\mathbf{p}^{(t)} &= \underset{\mathbf{p}\in\mathcal{X}}{\operatorname{argmin}} \left\{ \left\langle \nabla E\left(\mathbf{x}^{(t)}\right), \mathbf{p}\right\rangle - \lambda_i H(\mathbf{p}_i) \right\} \\ &= \operatorname{softmax}\left(\mathbf{R}\right) \quad \forall t \geq 0\end{aligned} \tag{31}$$

, where $\mathbf{R}\in\mathbb{R}^{nd}$ and

$$\mathbf{R}_i = -\frac{1}{\lambda_i}\left(\nabla E\left(\mathbf{x}_i^{(t)}\right)\right) \quad \forall i \in \mathcal{V}$$

This extension is still a special case of the regularized Frank-Wolfe algorithm. As a result, it inherits all the convergence properties from the regularized Frank-Wolfe mentioned in the previous section. On the other hand, it is also an extension of MFVI, which allows adding a message weight to each variable during inference.

A.3 A Proof for Extended Entropic Frank-Wolfe

We give a simple proof to the close-form solution of extended Entropic Frank-Wolfe in Equation 31. Since the optimization could reduce to n independent subproblems over each $i \in \mathcal{V}$, We only need to give the closed-form solution to each subproblem:

Lemma 1. *For a given vector $\mathbf{c}\in\mathbb{R}^d$, $\lambda > 0$, the optimal solution z^* to*

$$\min_{\mathbf{z}\in\Delta}\left\{\langle\mathbf{c},\mathbf{z}\rangle + \lambda\sum_{s=1}^{d} z_s\log z_s\right\}$$

is $\mathbf{z}^* = \operatorname{softmax}(-\frac{1}{\lambda}\mathbf{c})$, where Δ is the probability simplex $\left\{\mathbf{x}\in\mathbb{R}^d : \mathbf{x}\geq\mathbf{0}, \mathbf{1}^\top\mathbf{x} = 1\right\}$.

Proof. We can rewrite the problem as

$$\begin{aligned}\min_{\mathbf{z}} \quad & \langle\mathbf{c},\mathbf{z}\rangle + \lambda\sum_{s=1}^{d} z_s\log z_s \\ s.t. \quad & \mathbf{1}^\top\mathbf{z} = 1, \\ & -\mathbf{z} \leq \mathbf{0},\end{aligned}$$

The Lagrangian of the above problem is given by

$$\begin{aligned}L(\mathbf{z},\boldsymbol{\mu},\nu) &= \langle\mathbf{c},\mathbf{z}\rangle + \lambda\sum_{s=1}^{d} z_s\log z_s \\ &\quad + \boldsymbol{\mu}^\top(-\mathbf{z}) + \nu\left(\mathbf{1}^\top\mathbf{z} - 1\right) \\ &= -\nu + \sum_{s=1}^{d}(c_s z_s + \lambda z_s\log z_s \\ &\quad - \mu_s z_s + \nu z_s)\end{aligned}$$

where $\boldsymbol{\mu} = (\mu_1,\mu_2,\ldots,\mu_d) \geq \mathbf{0}$ and $\nu\in\mathbb{R}$ are the Lagrange multipliers.

Since the given problem is convex and there exists $\mathbf{z}\in\mathbb{R}^d$ such that $\mathbf{1}^\top\mathbf{z} = 1$ and $\mathbf{z} > \mathbf{0}$, the Slater's constraint qualification holds. Thus, it suffices to solve the following Karush-Kuhn-Tucker (KKT) system to obtain the optimal solution:

$$\begin{aligned} c_s + \lambda\log z_s + 1 - \mu_s + \nu = 0 \quad & \forall 1 \leq s \leq d, \\ \mathbf{1}^\top\mathbf{z} = 1, \\ \mathbf{z} \geq \mathbf{0}, \\ \boldsymbol{\mu} \geq \mathbf{0}, \\ \mu_s z_s = 0 \quad & \forall 1 \leq s \leq d.\end{aligned}$$

The first equation implies $\forall 1 \leq s \leq d, z_s > 0$, and thus in combination with the last, we obtain $\forall 1 \leq s \leq d, \mu_s = 0$. Therefore, the first equation becomes

$$c_s + \lambda\log z_s + 1 + \nu = 0$$

$\forall 1 \leq s \leq d$. Rewrite the equation as

$$z_s = \exp\left(\frac{-1-\nu}{\lambda}\right)\exp\left(-\frac{1}{\lambda}c_s\right)$$

$\forall 1 \leq s \leq d$. Summing up this result for all s, and taking into account the second equation, we have

$$\sum_{s=1}^{d} \exp\left(\frac{-1-\nu}{\lambda}\right) \exp\left(-\frac{1}{\lambda} c_s\right) = 1$$

That is,

$$\exp\left(\frac{-1-\nu}{\lambda}\right) = \frac{1}{\sum_{s=1}^{d} \exp\left(-\frac{1}{\lambda} c_s\right)}$$

Combine these two formulas, we have

$$z_s = \frac{\exp\left(-\frac{1}{\lambda} c_s\right)}{\sum_{t=1}^{d} \exp\left(-\frac{1}{\lambda} c_t\right)}$$

$\forall 1 \leq s \leq d$. In other words, $\mathbf{z} = \text{softmax}(-\frac{1}{\lambda}\mathbf{c})$. □

A.4 Inference in CRF

In this work, we apply the extended Entropic Frank-Wolfe to do inference in the CRF. Let $\mathbf{s} = (Z_1, \cdots, Z_n, H_1^{(1)}, \cdots, H_n^{(1)}, H_1^{(2)}, \cdots, H_n^{(h)})$ denote an assignment to all the random variables. Our CRF encodes the joint distribution

$$p(\mathbf{s}) = \frac{1}{Z} \prod_i \phi_u(Z_i) \prod_c \prod_i \prod_{j \neq i} \phi_t(H_i^{(c)}, Z_i, Z_j)$$

where Z is a normalization factor. The objective is to find an assignment $\mathbf{s}$ that maximizes the joint distribution $p(\mathbf{s})$. To express in the form of an energy function, let $p(\mathbf{s}) = \frac{1}{Z}\exp(-e(\mathbf{s}))$, we have

$$e(\mathbf{s}) = -\sum_i \mathbf{S}_{w_i, Z_i} - \sum_c \sum_i \sum_{j \neq i} \mathbb{1}_{H_i = j} \mathbf{T}^{(c)}_{Z_i, Z_j}$$

where $\mathbb{1}_{H_i=j}$ is an indicator function, which is equal to 1 if $H_i = j$ and is equal to 0 otherwise. The objective could now be expressed as minimizing the energy function $e(\mathbf{s})$.

In general, the problem of CRF inference is NP-Hard (Shimony, 1994). In MFVI, we solve the continuous relaxation of the CRF problem instead. Let $\mathcal{X}$ be the simplex. That is, we allow a marginal distribution for each random variable. As in Section 2.2, let $Q_i(\cdot)$ be the approximate marginal distribution over Z_i and $Q_{ic}(\cdot)$ be the approximate marginal distribution over $H_i^{(c)}$. The energy function is then

$$E(Q_*) = -\sum_i \sum_a Q_i(a) \mathbf{S}_{w_i, a} - \sum_c \sum_i \sum_{j \neq i} \sum_a \sum_b Q_i(a) Q_j(b) Q_{ic}(j) \mathbf{T}^{(c)}_{a,b}$$

Then we have

$$\frac{\partial E}{\partial Q_i(a)} = -\mathbf{S}_{w_i, a} - \sum_c \sum_{j \neq i} \sum_b \left(Q_j(b) Q_{ic}(j) \mathbf{T}^{(c)}_{a,b} + Q_j(b) Q_{jc}(i) \mathbf{T}^{(c)}_{b,a} \right)$$

$$\frac{\partial E}{\partial Q_{ic}(j)} = -\sum_a \sum_b Q_i(a) Q_j(b) \mathbf{T}^{(c)}_{a,b}$$

In MFVI, the update for each distribution is the softmax of the derivative (let $\lambda = 1$ and $\alpha_t = 1, \forall t \geq 0$ in Equation 30). That is,

$$Q_i^{(t)}(a) \propto \exp\left(-\frac{\partial E^{(t-1)}}{\partial Q_i^{(t-1)}(a)}\right)$$

$$Q_{ic}^{(t)}(j) \propto \exp\left(-\frac{\partial E^{(t-1)}}{\partial Q_{ic}^{(t-1)}(j)}\right)$$

Together with Equation 3 and 4, we have

$$\frac{\partial E^{(t-1)}}{\partial Q_i^{(t-1)}(a)} = -\mathbf{S}_{w_i, a} - \mathcal{G}_i^{(t-1)}(a)$$

$$\frac{\partial E^{(t-1)}}{\partial Q_{ic}^{(t-1)}(j)} = -\mathcal{F}_{ic}^{(t-1)}(j)$$

, which directly leads us to Formula 5 and 6.

In the extended Entropic Frank-Wolfe, the update for each distribution is the regularized softmax of the derivative (Equation 31). That is,

$$Q_i^{(t)}(a) \propto \exp\left(-\frac{1}{\lambda_i}\frac{\partial E^{(t-1)}}{\partial Q_i^{(t-1)}(a)}\right)$$

$$Q_{ic}^{(t)}(j) \propto \exp\left(-\frac{1}{\lambda_{ic}}\frac{\partial E^{(t-1)}}{\partial Q_{ic}^{(t-1)}(j)}\right)$$

Let $\lambda_i = \lambda_Z > 0, \lambda_{ic} = \lambda_H > 0, \forall i, c$. Then it is equivalent to Formula 11 and 12 with regularization weight $\lambda_Z > 0$ for Z variables and $\lambda_H > 0$ for H variables.

A.5 The Choice of Message Weights

In Section 2.3.3, we set $\lambda_Z = 1$ and $\lambda_H = \frac{1}{d}$ by default. This choice comes from a theoretical analysis similar to Vaswani et al. (2017), and we empirically find it helpful to improve the performance.

Assume that the ternary scores in $\mathbf{T}$ are independent random variables with mean 0 and variance σ^2. Then from Equation 3, we know that $\mathcal{F}_{ic}^{(t)}(j)$ is a weighted sum of these random variables. Suppose the weights are uniformly distributed, then $\mathcal{F}_{ic}^{(t)}(j)$

has mean 0 and variance $\frac{d^2}{(d^2)^2}\sigma^2 = \frac{1}{d^2}\sigma^2$. Since d is usually set to several hundred, this might result in a small variance in the message received by H variables and thus lead to uniformly distributed H variables. To balance this effect, we set $\lambda_H = \frac{1}{d}$ such that the variance of $\frac{1}{\lambda_H}\mathcal{F}_{ic}^{(t)}(j)$ is still σ^2. From Equation 4 we know that the variance of $\mathcal{G}_i^{(t)}(a)$ is $\frac{2(n-1)}{hd}\sigma^2$. Here, since n varies in sentences, it is impossible to set a fixed λ_Z that always recovers the original variance σ^2. Compared to $\mathcal{F}_{ic}^{(t)}(j)$, the variance of $\mathcal{G}_i^{(t)}(a)$ does not change significantly. For simplicity, we set $\lambda_Z = 1$.

B More Extensions and Variants

We have introduced several extensions and variants that are beneficial to the model performance in Section 2.3. There are some other variants that we find do not bring significant improvement empirically, but might also be meaningful and have interesting correspondences to transformers.

B.1 Step Size

In our model, we can retain information between iterations and do partially update with a proper step size. Let

$$Q_i^{\star(t)}(a) \propto \exp\left(\mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a)\right)$$
$$Q_{ic}^{\star(t)}(j) \propto \exp\left(\mathcal{F}_{ic}^{(t-1)}(j)\right)$$

be the original posterior marginal distributions of the variables at time step t, which is the same as Formula 5 and 6. We have the posterior distributions with step size

$$Q_i^{(t)}(Z_i) = \alpha_Z Q_i^{\star(t)}(Z_i) + (1-\alpha_Z)Q_i^{(t-1)}(Z_i)$$
$$Q_{ic}^{(t)}(H_i^{(c)}) = \alpha_H Q_{ic}^{\star(t)} H_i^{(c)} + (1-\alpha_H)Q_{ic}^{(t-1)} H_i^{(c)}$$

where $\alpha_Z, \alpha_H \in (0, 1]$ are the step sizes of each update. When $\alpha_Z = \alpha_H = 1$, it is equivalent to the original model. We initialize these distribution by Formula 7 and 8.

B.2 Damping

Similar to step size in Appendix B.1, the damping approach also aims at retaining information between iterations. Instead of partially updating the posterior distribution, the damping approach partially updates the messages.

We define messages in time step t as

$$M_i^{(t)}(a) = \mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a) \tag{32}$$
$$M_{ic}^{(t)}(j) = \mathcal{F}_{ic}^{(t-1)}(j) \tag{33}$$

where $M_i^{(t)}(Z_i)$ is the message passed to Z_i and $M_{ic}^{(t)}(H_i^{(c)})$ is the message passed to $H_i^{(c)}$. Thus, Formula 5 and 6 can be written as

$$Q_i^{(t)}(a) \propto \exp\left(M_i^{(t)}(a)\right)$$
$$Q_{ic}^{(t)}(j) \propto \exp\left(M_{ic}^{(t)}(j)\right)$$

Now, we add damping factors β_Z and β_H, which restrict the message update between iterations. We change Equation 32 and 33 to

$$\begin{aligned} M_i^{(t)}(a) =& (1-\beta_Z)\left(\mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a)\right) \\ &+ \beta_Z M_i^{(t-1)}(a) \\ M_{ic}^{(t)}(j) =& (1-\beta_H)\left(\mathcal{F}_{ic}^{(t-1)}(j)\right) + \beta_H M_{ic}^{(t-1)}(j) \end{aligned}$$

We initialize the message by

$$M_i^{(0)}(a) = \mathbf{S}_{w_i,a}$$
$$M_{ic}^{(0)}(j) = 0$$

When $\beta_Z = \beta_H = 0$, there is no damping in the update process and it is equivalent to the original model. When $\beta_Z = 0.5$ and $\beta_H = 0$, it is similar to the residual connection in transformers. When $\beta_Z = \beta_H = 0.5$, it is similar to the residual attention mechanism proposed in RealFormer (He et al., 2021).

B.3 Global Variables

As we mentioned in Section 3.4, probabilistic transformers do not have a feed-forward structure as in transformers. Feed-forward layers, however, constitute two-thirds of a transformer model's parameters. Recent researches show that the feed-forward layers might serve as an important part of transformers (Dong et al., 2021; Geva et al., 2021, 2022).

Inspired by Sukhbaatar et al. (2019), who combines the feed-forward layer and the self-attention layer into a unified all-attention layer, we design a similar structure based on dependency relations. Intuitively, we could add some global variables that are similar to the latent word representations (Z variables) but these representations are global features that do not change with input sentences. We will introduce 3 different model designs below.

B.3.1 All-dep

Based on the intuition above, we add some global variables to the CRF model. Define F_i as the i-th discrete global feature variable with the same label set as Z variables, representing the global features of the corpus. The total number of global feature variables is m. These variables are observed and the distributions on the label set will not change during inference. The head of each word could either be another word or a global feature variable. That is, $H_i^{(c)} \in \{1, 2, \cdots, n, n+1, \cdots, n+m\}$.

Then, for each word w_i and global feature F_j in channel c, we define a ternary potential function over Z_i, $H_i^{(c)}$ and F_j, which evaluates the compatibility between the labels of the word and the global feature of the entire corpus.

$$\phi_t(H_i^{(c)}, Z_i, F_j) = \begin{cases} \exp(\mathbf{T}''^{(c)}_{Z_i,F_j}), & H_i^{(c)} = n + j \\ 1, & \text{otherwise} \end{cases}$$

where $\mathbf{T}''^{(c)} \in \mathbb{R}^{d \times d}$ is a score matrix for channel c.

An illustration of the CRF model is shown in Figure 4. We call this setting *all-dep* since the head of each word could either be another word or a dummy global feature variable. It follows the *all-attn* setting in Sukhbaatar et al. (2019).

Notice that F_j is a variable that does not participate in inference. It could be seen as part of the model. Thus, we could design an equivalent model that does not contain global feature variables but have a binary factor between Z_i and $H_i^{(c)}$:

$$\phi_b(H_i^{(c)}, Z_i) = \begin{cases} \prod_g \exp(P(F_{H_i^{(c)}-n} = g)\mathbf{T}''^{(c)}_{Z_i,g}), & H_i^{(c)} > n \\ 1, & \text{otherwise} \end{cases}$$

where $P(F_i = g)$ is the probability that the i-th global variable has label g. It can be proved that the MFVI inference process for the model with global feature variables and the model with binary factors is the same. Move the product inside the exponential term, we have

$$\phi_b(H_i^{(c)}, Z_i) = \begin{cases} \exp(\sum_g P(F_{H_i^{(c)}-n} = g)\mathbf{T}''^{(c)}_{Z_i,g}), & H_i^{(c)} > n \\ 1, & \text{otherwise} \end{cases}$$

The term inside the exponential is a weighted sum of ternary scores. We may re-formulate this potential function with a simplified term:

$$\phi_b(H_i^{(c)}, Z_i) = \begin{cases} \exp(\mathbf{B}^{(c)}_{H_i^{(c)}-n,Z_i}), & H_i^{(c)} > n \\ 1, & \text{otherwise} \end{cases}$$

where $\mathbf{B}^{(c)} \in \mathbb{R}^{m,d}$ is a score matrix for channel c. The weighted sum of ternary scores could be regarded as a neural parameterization of the binary scores $\mathbf{B}^{(c)}$. An illustration of the simplified CRF model is shown in Figure 5.

Given the model above, we can now derive the following iterative update equations of posterior distribution:

$$\mathcal{F}_{ic}^{(t)}(j) = \begin{cases} \sum_a \sum_b \left(Q_i^{(t)}(a) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)}\right), & j \le n \\ \sum_a \left(Q_i^{(t)}(a) \mathbf{B}_{j,a}^{(c)}\right), & j > n \end{cases} \tag{34}$$

$$\begin{aligned} \mathcal{G}_i^{(t)}(a) = &\sum_c \sum_{j \neq i, j \le n} \sum_b \Big(Q_{ic}^{(t)}(j) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)} \\ &+ Q_{jc}^{(t)}(i) Q_j^{(t)}(b) \mathbf{T}_{b,a}^{(c)}\Big) \\ &+ \sum_c \sum_{j > n} Q_{ic}^{(t)}(j) \mathbf{B}_{j,a}^{(c)} \end{aligned} \tag{35}$$

where

$$Q_i^{(t)}(a) \propto \exp\left(\mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a)\right) \tag{36}$$

$$Q_{ic}^{(t)}(j) \propto \exp\left(\mathcal{F}_{ic}^{(t-1)}(j)\right) \tag{37}$$

The initialization of the posterior marginal distributions $Q_i^{(t)}(\cdot)$ and $Q_{ic}^{(t)}(\cdot)$ is the same as Formula 7 and 8. Notice that $F_{ic}^{(t)} \in \mathbb{R}^{n+m}$ looks like a concatenation of a context vector and a persistent vector in all-attention networks (Sukhbaatar et al., 2019).

B.3.2 Dep-split

Following the *attn-split* setting in Sukhbaatar et al. (2019), we also design a *dep-split* version of our model. In each channel, we split the head of each word into two heads: one for the head word in the sentence and one for the global feature. We call the heads for global features 'global heads'.

Denote $G_i^{(c)} \in \{1, \cdot, m\}$ as the global head variable for i-th word in channel c. $H_i^{(c)} \in \{1, \cdot, n\}$

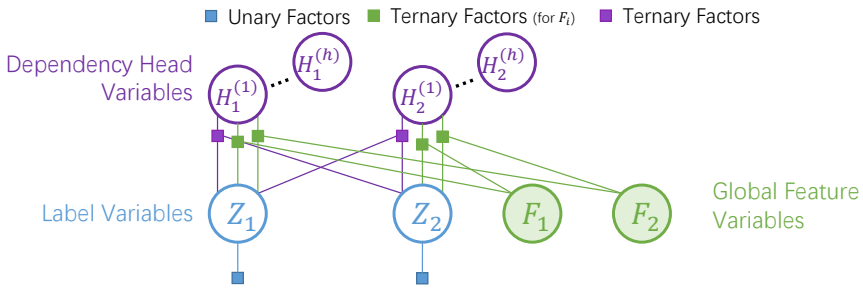


Figure 4: The factor graph for an intuitive design of CRF model with global variables where $n = m = 2$. For clarity, ternary factors that connect to $H_i^{(c)}$ with $c > 1$ are not shown in the figure.

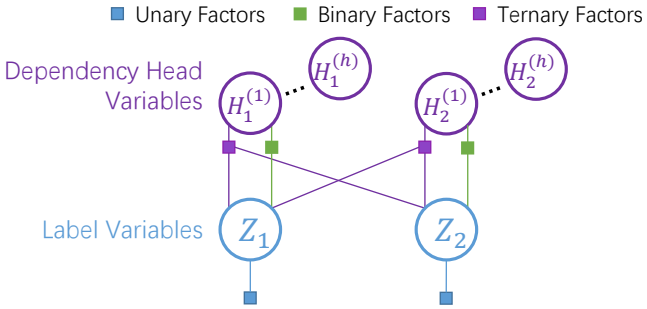


Figure 5: An equivalent factor graph for the *all-dep* CRF model in Figure 4.

is still the variable representing the syntactic dependency head of the i-th word in the c-th channel. Similar to the approaches in the *all-dep* setting, we define a simplified binary potential function for Z_i and $G_i^{(c)}$

$$\phi_b(G_i^{(c)} = k, Z_i = a) = \exp\left(\mathbf{B}_{k,a}^{(c)}\right) \tag{38}$$

Figure 6 illustrates the CRF model of the *dep-split* setting.

We could derive the following iterative update equations of posterior distribution:

$$\mathcal{F}_{ic}^{(t)}(j) = \sum_a \sum_b \left(Q_i^{(t)}(a) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)}\right) \tag{39}$$

$$\mathcal{H}_{i,k,c}^{(t)} = \sum_a \left(Q_i^{(t)}(a) \mathbf{B}_{k,a}^{(c)}\right) \tag{40}$$

$$\begin{aligned}\mathcal{G}_i^{(t)}(a) = &\sum_c \sum_{j \neq i} \sum_b Q_{ic}^{(t)}(j) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)} \\ &+ \sum_c \sum_{j \neq i} \sum_b Q_{jc}^{(t)}(i) Q_j^{(t)}(b) \mathbf{T}_{b,a}^{(c)} \\ &+ \sum_c \sum_k Q_{ic}^{\prime(t)}(k) \mathbf{B}_{k,a}^{(c)}\end{aligned} \tag{41}$$

where

$$Q_i^{(t)}(a) \propto \exp\left(\mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a)\right) \tag{42}$$

$$Q_{ic}^{(t)}(j) \propto \exp\left(\mathcal{F}_{ic}^{(t-1)}(j)\right) \tag{43}$$

$$Q_{ic}^{\prime(t)}(k) \propto \exp\left(\mathcal{H}_{i,k,c}^{(t-1)}\right) \tag{44}$$

are the approximate marginal distributions at time step t, with $Q_{ic}^{\prime(t)}(\cdot)$ over $G_i^{(c)}$. We initialize these distributions by Formula 7, 8 and

$$Q_{ic}^{\prime(0)}(k) \propto 1 \tag{45}$$

B.3.3 Single-split

Following the *single-split* setting in Sukhbaatar et al. (2019), we design a CRF model that is similar to the *dep-split* model but only allows one global head for each word. We also call this setting *single-split*. Denote G_i as the global head variable for i-th word with a label set of size m. We define a binary potential for Z_i and G_i

$$\phi_b(G_i = k, Z_i = a) = \exp\left(\mathbf{B}_{k,a}\right) \tag{46}$$

where $\mathbf{B} \in \mathbb{R}^{m \times d}$ is a score matrix. Figure 7 illustrates the CRF model of the *single-split* setting.

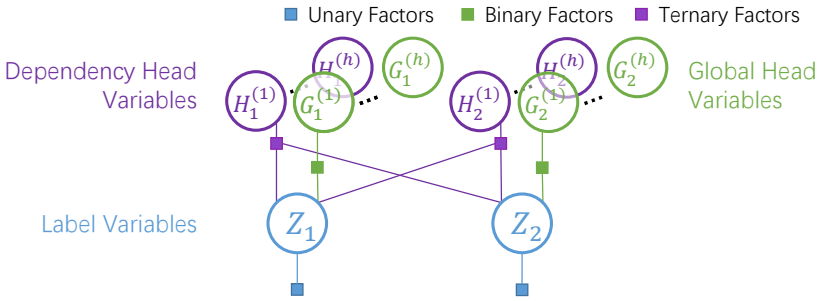


Figure 6: The factor graph for the *dep-split* CRF model where $n = 2$. For clarity, binary and ternary factors with channel $c > 1$ are not shown in the figure.

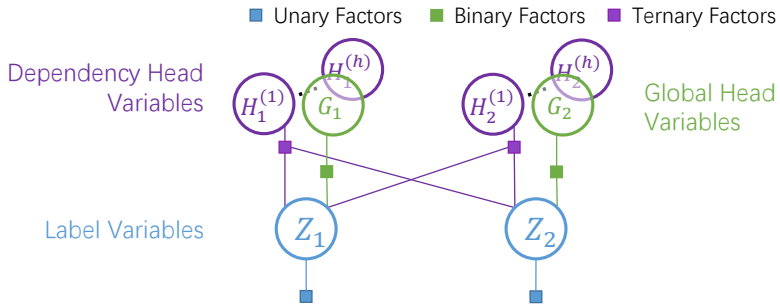


Figure 7: The factor graph for the *single-split* CRF model where $n = 2$. For clarity, ternary factors with channel $c > 1$ are not shown in the figure.

We could derive the following iterative update equations of posterior distribution:

$$\mathcal{F}_{ic}^{(t)}(j) = \sum_a \sum_b \left(Q_i^{(t)}(a) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)} \right) \tag{47}$$

$$\mathcal{H}_{i,k}^{(t)} = \sum_a \left(Q_i^{(t)}(a) \mathbf{B}_{k,a} \right) \tag{48}$$

$$\begin{aligned} \mathcal{G}_i^{(t)}(a) = & \sum_c \sum_{j \neq i} \sum_b Q_{ic}^{(t)}(j) Q_j^{(t)}(b) \mathbf{T}_{a,b}^{(c)} \\ & + \sum_c \sum_{j \neq i} \sum_b Q_{jc}^{(t)}(i) Q_j^{(t)}(b) \mathbf{T}_{b,a}^{(c)} \\ & + \sum_k Q_i^{'(t)}(k) \mathbf{B}_{k,a} \end{aligned} \tag{49}$$

where

$$Q_i^{(t)}(a) \propto \exp\left(\mathbf{S}_{w_i,a} + \mathcal{G}_i^{(t-1)}(a) \right) \tag{50}$$

$$Q_{ic}^{(t)}(j) \propto \exp\left(\mathcal{F}_{ic}^{(t-1)}(j) \right) \tag{51}$$

$$Q_i^{'(t)}(k) \propto \exp\left(\mathcal{H}_{i,k}^{(t-1)} \right) \tag{52}$$

are the approximate marginal distributions at time step t, with $Q_i^{'(t)}(\cdot)$ over G_i. We initialize these distributions by Formula 7, 8 and

$$Q_i^{'(0)}(k) \propto 1 \tag{53}$$

single-split might be the setting that has the most similar computation process to that of transformers. If we consider the tensorized form of *single-split*, then for the posterior distributions of all the G variables $Q_g^{(t)} \in \mathbb{R}^{n \times m}$, we have

$$\mathcal{F}_c^{(t)} = Q_z^{(t)} \mathbf{T}^{(c)} Q_z^{(t)T} \tag{54}$$

$$\mathcal{H}^{(t)} = Q_z^{(t)} \mathbf{B}^T \tag{55}$$

$$\begin{aligned} \mathcal{G}^{(t)} = & \sum_c Q_{h,c}^{(t)} Q_z^{(t)} \mathbf{T}^{(c)T} \\ & + \sum_c Q_{h,c}^{(t)T} Q_z^{(t)} \mathbf{T}^{(c)} \\ & + Q_g^{(t)} \mathbf{B} \end{aligned} \tag{56}$$

where

$$Q_z^{(t)} = \sigma\left(\mathbf{S} + \mathcal{G}^{(t-1)} \right) \tag{57}$$

$$Q_{h,c}^{(t)} = \sigma\left(\mathcal{F}_c^{(t-1)} \right) \tag{58}$$

$$Q_g^{(t)} = \sigma\left(\mathcal{H}^{(t-1)} \right) \tag{59}$$

With the similar trick in Section 3, we have

$$\begin{aligned} Q_z^{(t)} = & \sigma(\mathbf{S} + 2 \sum_c \text{channel}_c \, \mathbf{U}^{(c)T} \\ & + \text{GFU}(Q_z^{(t-1)})) \end{aligned} \tag{60}$$

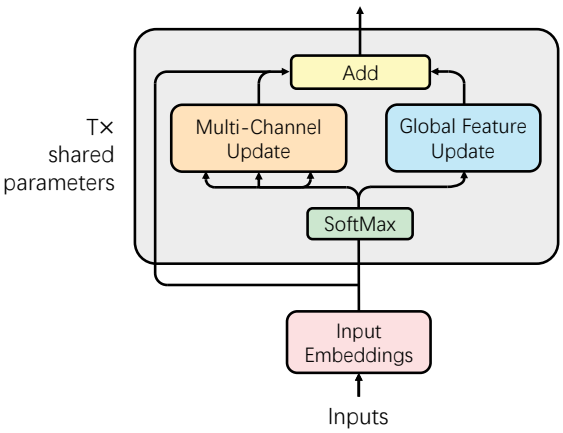


Figure 8: Computation graph for the *single-split* probabilistic transformer.

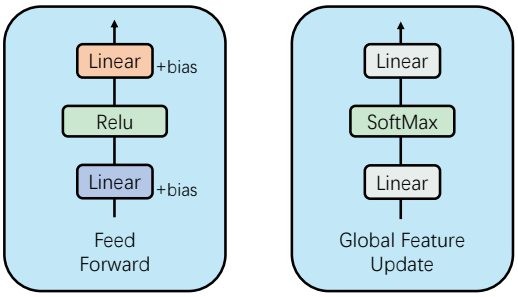


Figure 9: Computation graph for feed-forward in transformers and global feature update in probabilistic transformers with global variables.

where

$$\text{channel}_c = \sigma\left(\frac{Q_c K_c^T}{\lambda_H}\right) V_c \tag{61}$$

$$\text{GFU}(x) = \sigma\left(x\mathbf{B}^T\right)\mathbf{B} \tag{62}$$

where we can regard GFU as an operator that updates the latent word representations from global features. An illustration of the computation process is shown in Figure 8. From Figure 9, we can see that the feed-forward structure in transformers is very similar to the global feature update process in probabilistic transformers with global variables.

C Distance and Relative Positional Encoding (RPE)

In Section 3.2, we find that the single-channel update (Equation 29) in probabilistic transformers is almost identical to scaled dot-product attention in transformers. This observation is based on the hypothesis that probabilistic transformers and transformers are sharing the same positional encoding method. But this is not the case.

In section 2.3.1, we mention that to capture the word order information, we use a clip function to select the ternary potential function based on the distance of two words (Equation 9). This is similar to the relative positional encoding (RPE) in transformers. Shaw et al. (2018) proposes a method to add an additional component to key and value, based on the clipped distance. Specifically, the scaled dot-product attention with RPE could be rewritten as

$$e_{ij} = \frac{x_i W^Q \left(x_j W^K + a_{ij}^K\right)^T}{\sqrt{d_k}}$$

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V + a_{ij}^V\right)$$

where x_i is the input representation of the i-th word, z_i is the output representation, $\alpha_{ij} = \frac{\exp e_{ij}}{\sum_k \exp e_{ik}}$. The additional component is a learnable parameter that based on the clipped distance

$$a_{ij}^K = w_{\text{clip}(j-i,k)}^K$$
$$a_{ij}^V = w_{\text{clip}(j-i,k)}^V$$
$$\text{clip}(x,k) = \max(-k, \min(k,x))$$

For probabilistic transformers, we directly add the distance information to the ternary potential function. Combining Equation 9 and 29, we could rewrite the single-channel update as

$$e_{ij} = \frac{x_i \mathbf{U}_{ij} \left(x_j \mathbf{V}_{ij}\right)^T}{\lambda_H}$$

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j \mathbf{V}_{ij}\right)$$

where $\alpha_{ij} = \frac{\exp e_{ij}}{\sum_k \exp e_{ik}}$. The weights are based on the clip function f in Equation 10

$$\mathbf{U}_{ij} = \mathbf{U}[f(i-j)]$$
$$\mathbf{V}_{ij} = \mathbf{V}[f(i-j)]$$

Notice that this way of positional encoding is quite parameter inefficient. It also makes our training process much slower than that of transformers.

D Details for Tasks and Datasets

In this section, we will introduce our tasks and datasets in detail. A brief introduction is shown in Section 4.1.

D.1 Masked Language Modeling

Masked Language Modeling (MLM) tasks generally evaluate the expressiveness of contextural word representations. We perform MLM tasks on two corpora: the Penn TreeBank (PTB) and Brown Laboratory for Linguistic Information Processing (BLLIP). We randomly replace words with a mask token <mask> at a rate of 30% and the model is required to predict the original word. Following Shen et al. (2022), we never mask <unk> tokens. The performance of MLM is evaluated by measuring perplexity (lower is better) on masked words.

PTB. The Penn Treebank (Marcus et al., 1993), in particular the sections of the corpus corresponding to the articles of Wall Street Journal (WSJ), is a standard dataset for language modeling (Mikolov et al., 2012) and sequence labeling (Dinarelli and Grobol, 2019). Following the setting in Shen et al. (2021), we use the preprocessing method proposed in Mikolov et al. (2012). It removes all punctuation and replaces low-frequency words with <unk>. The processed dataset has a vocabulary size of 10000, including <unk> and <mask>.

BLLIP. The Brown Laboratory for Linguistic Information Processing dataset (Charniak et al., 2000) is a large corpus similar to the PTB dataset in style. The entire dataset contains 24 million sentences from Wall Street Journal. In our experiments, we only use a small subset of this corpus. Following the same setting as Shen et al. (2022), we use the BLLIP-XS split proposed in Hu et al. (2020) with around 40k sentences and 1M tokens as the train set. The validation set consists of the first section each year and the test set consists of the second section each year. We remove all punctuation, replace numbers with a single character N and use lower-case letters. The vocabulary contains words that appear more than 27 times in the entire BLLIP dataset, with size 30231 including <unk> and <mask>.

D.2 Sequence Labeling

Sequence labeling tasks require models to predict the tag for each word in the sequence. For sequence labeling tasks, we perform part-of-speech (POS) tagging on two datasets: the Penn TreeBank (PTB) and the Universal Dependencies (UD). We also perform named entity recognition (NER) on CoNLL-2003.

PTB. As introduced in Appendix D.1, we also use the PTB dataset for POS tagging but with a different setting. We use the most commons split of this corpus for POS tagging, where sections from 0 to 18 are used as the train set, sections from 19 to 21 are used as the validation set, and sections from 22 to 24 are used as the test set. All words in the train set compose the vocabulary.

UD. UD is a project that develops cross-linguistically consistent treebank annotation for many languages (De Marneffe et al., 2021). We test our model on the language-specific part-of-speech (XPOS) tags of the English EWT dataset with the standard splits. All words in the train set compose the vocabulary.

CoNLL-2003. It is a named entity recognition dataset which is released as part of CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003). We test our model on the English dataset. All words in the train set compose the vocabulary. We only project the final word representation of each word to the tag set with the BIOES scheme without using a CRF decoder.

D.3 Text Classification

Text Classification tasks need to classify sentences into different classes. We use the Stanford Sentiment Treebank (SST) (Socher et al., 2013) as the dataset. It has two variants: binary classification (SST-2) and fine-grained classification (SST-5). The dataset comes from SentEval (Conneau and Kiela, 2018).

SST-2. SST-2 classifies each movie review into positive or negative classes. It contains 67k sentences in the train set.

SST-5. SST-5 classifies sentences into 5 classes: negative, somewhat negative, neutral, somewhat positive and positive. It contains 8.5k sentences in the train set.

In text classification, all words in the train set compose the vocabulary.

D.4 Syntactic Test

To evaluate the compositional generalization abilities of our model, we perform a syntactic test on the COGS (Kim and Linzen, 2020) dataset. COGS is a semantic parsing dataset that measures the compositional generalization abilities of models. We follow the settings in Ontanón et al. (2021), which turns the task from seq2seq into a sequence tagging task. The model needs to predict 5 tags for each input word: a *parent* word, the *role* of the relation between the word and its parent (if applicable), the *category*, the *noun determiner* (for nouns) and the

verb name (for verbs). With these tags, one can reconstruct the original output deterministically.

For *role*, *category*, *noun determiner* and *verb name*, we directly project word representations to each tag set. For the *parent* tag, (Ontanón et al., 2021) propose 3 types of prediction heads:

- *Absolute* uses a direct projection to predict the absolute index of the parent in the input sequence (-1 for no parent).
- *Relative* uses a direct projection to predict the relative offset of the parent token with respect to the current token, or self for no parent.
- *Attention* uses the attention weights from a new attention layer with a single head to predict the parent.

We empirically find that *relative* performs the best in most settings for both transformers and probabilistic transformers. This is not consistent with the observations in Ontanón et al. (2021) who finds that *attention* outperforms other settings. We still apply the *relative* setting in our experiments.

E Hyperparameters and Implementation

We report our hyperparameters in Table 2 for probabilistic transformers and Table 3 for transformers. We tune the models for each task except the syntactic test through random search. We run experiments on one NVIDIA GeForce RTX 2080 Ti and all the experiments could finish in one day. Our implementation is based on the flair framework (Akbik et al., 2019).

F Case Studies of Learned Dependency Structures

A probabilistic transformer infers marginal distributions over both Z and H variables, the latter of which can be used to extract a dependency structure. Since our model is trained on downstream tasks such as MLM without access to gold parse trees, it can be seen as performing unsupervised dependency parsing. We visualize the dependency structures learned by a probabilistic transformer by looking at the most probable head of each word in the sentence.

Figure 10 illustrates the dependency structures extracted from a probabilistic transformer trained on the PTB dataset under the MLM task. The sentence comes from the test set of the PTB dataset. We show the head of each word in all the channels. The numbers on the dependency arcs represent probabilities estimated by the model. The model does not contain a root node, so there is at least one circle in the dependency graph.

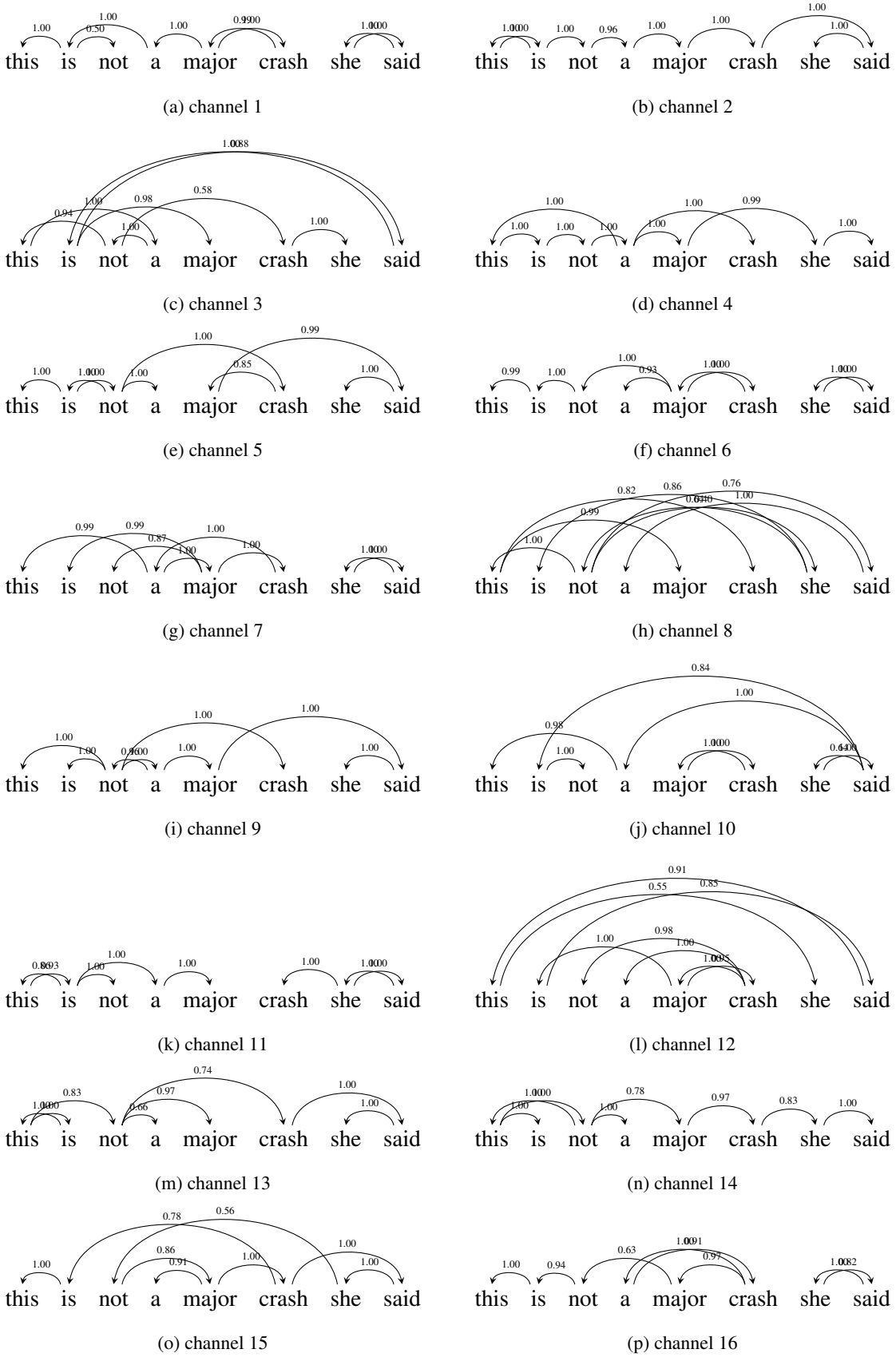
From the figure, we can see that our model is very confident in its choices of dependency arcs, with all the probabilities close to 1, which indicates strong compatibilities between the latent representations of connected word pairs. The predicted structure somewhat makes sense. For example, it puts 'she said' together. But generally, most of the dependency arcs are not consistent with human-designed dependency relations.

Probabilistic Transformer	MLM		POS		CLS		SYN
	PTB	BLLIP	PTB	UD	SST-2	SST-5	COGS
Label set size d	384	384	128	128	512	256	64
Root label set size d_{root}	–	–	–	–	1024	512	–
# of channels h	16	16	12	18	10	18	4
# of iterations T	5	5	3	2	1	4	2
Distance threshold γ	3	3	3	3	3	3	8
Decomposition	UV	UV	UV	–	UV	UVW	UV
Decomposition rank r	64	64	128	–	64	64	16
Dropout	0.15	0.15	0.05	0.1	0.1	0.05	0.1
Asynchronous update				Yes			
Learning rate	0.001	0.001	0.0024	0.0062	0.0001	0.0002	0.0025
Weight decay	1.4e-6	1.4e-6	8e-6	2.2e-6	3e-7	3e-7	1e-9
L2 reg for $\mathbf{T}$	5e-4	5e-4	0	4e-4	0	0	0

Table 2: Hyperparameters for probabilistic transformers in our experiments.

Transformer	MLM		POS		CLS		SYN
	PTB	BLLIP	PTB	UD	SST-2	SST-5	COGS
Embedding size d_{model}	384	256	512	384	256	128	64
FFN inner layer size d_{ff}	2048	2048	2048	512	512	1024	256
# of heads h	8	14	14	14	10	14	4
# of layers N	5	4	5	4	8	4	2
Positional Encoding	abs	abs	abs	abs	abs	abs	rel-8
Head dimension d_{qkv}	256	128	32	16	256	256	16
Dropout	0.15	0.15	0.15	0	0.05	0	0.1
Learning rate	0.0001	0.0002	0.0004	0.0004	0.0001	0.0002	0.0005
Weight decay	1.2e-6	3.5e-6	3.2e-6	1.4e-6	1.9e-6	2.7e-6	1e-9

Table 3: Hyperparameters for transformers in our experiments.



(a) channel 1

(b) channel 2

(c) channel 3

(d) channel 4

(e) channel 5

(f) channel 6

(g) channel 7

(h) channel 8

(i) channel 9

(j) channel 10

(k) channel 11

(l) channel 12

(m) channel 13

(n) channel 14

(o) channel 15

(p) channel 16

Figure 10: Dependency structures learned by a probabilistic transformer under the MLM task. The numbers on the dependency arcs represent the confidence of the head word.