Self-Attentional Models for Lattice Inputs

Matthias Sperber[1], **Graham Neubig**[2], **Ngoc-Quan Pham**[1], **Alex Waibel**[1,2]
[1]Karlsruhe Institute of Technology, Germany
[2]Carnegie Mellon University, USA
{first}.{last}@kit.edu, gneubig@cs.cmu.edu

Abstract

Lattices are an efficient and effective method to encode ambiguity of upstream systems in natural language processing tasks, for example to compactly capture multiple speech recognition hypotheses, or to represent multiple linguistic analyses. Previous work has extended recurrent neural networks to model lattice inputs and achieved improvements in various tasks, but these models suffer from very slow computation speeds. This paper extends the recently proposed paradigm of self-attention to handle lattice inputs. Self-attention is a sequence modeling technique that relates inputs to one another by computing pairwise similarities and has gained popularity for both its strong results and its computational efficiency. To extend such models to handle lattices, we introduce probabilistic reachability masks that incorporate lattice structure into the model and support lattice scores if available. We also propose a method for adapting positional embeddings to lattice structures. We apply the proposed model to a speech translation task and find that it outperforms all examined baselines while being much faster to compute than previous neural lattice models during both training and inference.

1 Introduction

In many natural language processing tasks, graph-based representations have proven useful tools to enable models to deal with highly structured knowledge. Lattices are a common instance of graph-based representations that allows capturing a large number of alternative sequences in a compact form (Figure 1). Example applications include speech recognition lattices that represent alternative decoding choices (Saleem et al., 2004; Zhang et al., 2005; Matusov et al., 2008), word segmentation lattices that capture ambiguous decisions on word boundaries or morphological alternatives (Dyer et al., 2008), word class lattices (Navigli and Velardi, 2010), and lattices for alternative video descriptions (Senina et al., 2014).

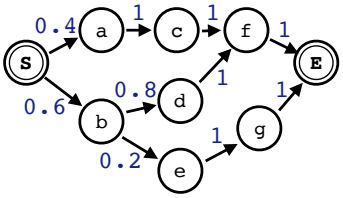


Figure 1: Example of a node-labeled lattice. Nodes are labeled with word tokens and posterior scores.

Prior work has made it possible to handle these through the use of recurrent neural network (RNN) lattice representations (Ladhak et al., 2016; Su et al., 2017; Sperber et al., 2017), inspired by earlier works that extended RNNs to tree structures (Socher et al., 2013; Tai et al., 2015; Zhu et al., 2015). Unfortunately, these models are computationally expensive, because the extension of the already slow RNNs to tree-structured inputs prevents convenient use of batched computation. An alternative model, graph convolutional networks (GCN) (Duvenaud et al., 2015; Defferrard et al., 2016; Kearnes et al., 2016; Kipf and Welling, 2017), is much faster but considers only local context and therefore requires combination with slower RNN layers for typical natural language processing tasks (Bastings et al., 2017; Cetoli et al., 2017; Vashishth et al., 2018).

For linear sequence modeling, self-attention (Cheng et al., 2016; Parikh et al., 2016; Lin et al., 2017; Vaswani et al., 2017) now provides an alternative to RNNs. Self-attention encodes sequences by relating sequence items to one another through computation of pairwise similarity, with addition of positional encoding to model positions of words in a linear sequence. Self-attention has gained popularity thanks to strong empirical results and computational efficiency afforded by paralleliz-

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1185–1197
Florence, Italy, July 28 - August 2, 2019.

able computations across sequence positions.

In this paper, we extend the previously purely sequential self-attentional models to lattice inputs. Our primary goal is to obtain additional modeling flexibility while avoiding the increased cost of previous lattice-RNN-based methods. Our technical contributions are two-fold: First, we incorporate the global lattice structure into the model through reachability masks that mimic the pairwise conditioning structure of previous recurrent approaches. These masks can account for lattice scores if available. Second, we propose the use of lattice positional embeddings to model positioning and ordering of lattice nodes.

We evaluate our method on two standard speech translation benchmarks, replacing the encoder component of an attentional encoder-decoder model with our proposed lattice self-attentional encoder. Results show that the proposed model outperforms all tested baselines, including LSTM-based and self-attentional sequential encoders, a LatticeLSTM encoder, and a recently proposed self-attentional model that is able to handle graphs but only considers local context, similar to GCNs. The proposed model performs well without support from RNNs and offers computational advantages in both training and inference settings.

2 Background

2.1 Masked Self-Attention

We start by introducing self-attentional models for sequential inputs, which we will extend to lattice-structured inputs in § 4.

Attentional models in general can be described using the terminology of queries, keys, and values. The input is a sequence of l values, along with a key corresponding to each value. For some given query, the model computes how closely each key matches the query. Here, we assume values, keys, and queries $\mathbf{v}_k, \mathbf{k}_k, \mathbf{q} \in \mathbb{R}^d$, for some dimensionality d and sequence indices $k \in \{1 \dots l\}$. Using the computed similarity scores $f(\mathbf{q}, \mathbf{k}_k)$, attention computes a weighted average of the values to obtain a fixed-size representation of the whole sequence conditioned on this query. In the self-attentional case, the sequence items themselves are used as queries, yielding a new sequence of same length as output in which each of the original input elements has been enriched by the respectively relevant global context.

The following equations formalize this idea. We are given a sequence of input vectors $\mathbf{x}_k \in \mathbb{R}^d$. For every query index i, we compute an output vector $\mathbf{y}_i$ as:

$$e_{ij} = f\left(q\left(\mathbf{x}_i\right), k\left(\mathbf{x}_j\right)\right) + m_{ij} \quad (\forall 1 \leq j \leq l) \quad (1)$$

$$\boldsymbol{\alpha}_i = \mathrm{softmax}\left(\mathbf{e}_i\right) \quad (2)$$

$$\mathbf{y}_i = \sum_{j=1}^{l} \alpha_{ij} v\left(\mathbf{x}_j\right). \quad (3)$$

Here, unnormalized pairwise similarities e_{ij} are computed through the similarity function f, and then normalized as α_{ij} for computation of a weighted sum of value vectors. q, k, v denote parametrized transformations (e.g. affine) of the inputs into queries, keys, and values.

Equation 1 also adds an attention masking term $m_{ij} \in \mathbb{R}$ that allows adjusting or disabling the influence of context at key position j on the output representation at query position i. Masks have, for example, been used to restrict self-attention to ignore future decoder context (Vaswani et al., 2017) by setting $m_{ij} = -\infty$ for all $j > i$. We will use this concept in § 4.1 to model reachability structure.

2.2 Lattices

We aim to design models for lattice inputs that store a large number of sequences in a compact data structure, as illustrated in Figure 1. We define lattices as directed acyclic graphs (DAGs) with the additional property that there is exactly one start node (`S`) and one end node (`E`). We call the sequences contained in the lattice *complete* paths, running from the start node to the end node. Each node is labeled with a word token.[1]

To make matters precise, let $G = (V, E)$ be a DAG with nodes V and edges E. For $k \in V$, let $\mathcal{R}^+_{\mathrm{G}}(k)$ denote all successors (reachable nodes) of node k, and let $\mathcal{N}^+_{\mathrm{G}}(k)$ denote the neighborhood, defined as the set of all adjacent successor nodes. $\mathcal{R}^-_{\mathrm{G}}(k), \mathcal{N}^-_{\mathrm{G}}(k)$ are defined analogously for predecessors. $j \succ i$ indicates that node j is a successor of node i.

For arbitrary nodes i, j, let $p_G\left(j \succ i \mid i\right)$ be the probability that a complete path in G contains j as a successor of i, given that i is contained in the path. Note that $j \notin \mathcal{R}^+_{\mathrm{G}}(i)$ implies $p_G\left(j \succ i \mid i\right) = 0$. The probability structure

[1] Edge-labeled lattices can be easily converted to node-labeled lattices using the line-graph algorithm (Hemminger and Beineke, 1978).

of the whole lattice can be represented through transition probabilities $p^{\text{trans}}_{k,j} := p_G(k \succ j \mid j)$ for $j \in \mathcal{N}^+_G(k)$. We drop the subscript G when clear from context.

3 Baseline Model

Our proposed model builds on established architectures from prior work, described in this section.

3.1 Lattice-Biased Attentional Decoder

The common attentional encoder-decoder model (Bahdanau et al., 2015) serves as our starting point. The encoder will be described in § 4. As cross-attention mechanism, we use the lattice-biased variant (Sperber et al., 2017), which adjusts the attention scores $\alpha^{\text{cross}}_{ij}$ between encoder position j and decoder position i according to marginal lattice scores $p(j \succ \mathtt{S} \mid \mathtt{S})$ (§ 4.1.2 describes how to compute these) as follows:[2]

$$\alpha^{\text{cross}}_{ij} \propto \exp\left(\mathit{score}(\bullet) + \log p(j \succ \mathtt{S} \mid \mathtt{S})\right). \quad (4)$$

Here, $\mathit{score}(\bullet)$ is the unnormalized attention score.

In the decoder, we use long short-term memory (LSTM) networks, although it is straightforward to use alternative decoders in future work, such as the self-attentional decoder proposed by Vaswani et al. (2017). We further use input feeding (Luong et al., 2015), variational dropout in the decoder LSTM (Gal and Ghahramani, 2016), and label smoothing (Szegedy et al., 2016).

3.2 Multi-Head Transformer Layers

To design our self-attentional encoder, we use Vaswani et al. (2017)'s *Transformer* layers that combine self-attention with position-wise feed-forward connections, layer norm (Ba et al., 2016), and residual connections (He et al., 2016) to form deeper models. Self-attention is modeled with multiple heads, computing independent self-attentional representations for several separately parametrized attention heads, before concatenating the results to a single representation. This increases model expressiveness and allows using different masks (Equation 1) between different attention heads, a feature that we will exploit in § 4.1. Transformer layers are computed as follows:

$$\mathbf{Q}_k = \mathbf{X}\mathbf{W}^{(\mathrm{q})}_k, \mathbf{K}_k = \mathbf{X}\mathbf{W}^{(\mathrm{k})}_k, \mathbf{V}_k = \mathbf{X}\mathbf{W}^{(\mathrm{v})}_k \quad (5)$$

$$\mathbf{H}_k = \mathrm{softmax}\left(\frac{\mathrm{dropout}\left(\mathbf{Q}_i\mathbf{K}_k^\top + \mathbf{M}\right)}{\sqrt{d}}\right)\mathbf{V}_k \quad (6)$$

$$\mathbf{H} = \mathrm{concat}(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_n) \quad (7)$$

$$\mathbf{L} = \mathrm{LN}\left[\mathrm{dropout}\left(\mathbf{H} + \mathbf{X}\right)\right] \quad (8)$$

$$\mathbf{Y} = \mathrm{LN}\left[\mathrm{dropout}\left(\mathrm{FF}\left(\mathbf{L}\right) + \mathbf{L}\right)\right] \quad (9)$$

Here, $\mathbf{X} \in \mathbb{R}^{l \times d}, \mathbf{Q}_k, \mathbf{K}_k, \mathbf{V}_k \in \mathbb{R}^{l \times d/n}$ denote inputs and their query-, key-, and value transformations for attention heads with index $k \in \{1, \ldots, n\}$, sequence length l, and hidden dimension d. $\mathbf{M} \in \mathbb{R}^{l \times l}$ is an attention mask to be defined in § 4.1. Similarity between keys and queries is measured via the dot product. The inputs are word embeddings in the first layer, or the output of the previous layer in the case of stacked layers. $\mathbf{Y} \in \mathbb{R}^{l \times d}$ denotes the final output of the Transformer layer. $\mathbf{W}^{(\mathrm{q})}_k, \mathbf{W}^{(\mathrm{k})}_k, \mathbf{W}^{(\mathrm{v})}_k \in \mathbb{R}^{d \times d/n}$ are parameter matrices. FF is a position-wise feed-forward network intended to introduce additional depth and nonlinearities, defined as $\mathrm{FF}(x) = \max\left(\mathbf{0}, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2$. LN denotes layer norm. Note that dropout regularization (Srivastava et al., 2014) is added in three places.

Up to now, the model is completely agnostic of sequence positions. However, position information is crucial in natural language, so a mechanism to represent such information in the model is needed. A common approach is to add positional encodings to the word embeddings used as inputs to the first layer. We opt to use learned positional embeddings (Gehring et al., 2017), and obtain the following after applying dropout:

$$\mathbf{x}'_i = \mathrm{dropout}\left(\mathbf{x}_i + \mathrm{embed}\,[i]\right). \quad (10)$$

Here, a position embedding $\mathrm{embed}\,[i]$ of equal dimension with sequence item $\mathbf{x}_i$ at position i is added to the input.

4 Self-Attentional Lattice Encoders

A simple way to realize self-attentional modeling for lattice inputs would be to linearize the lattice in topological order and then apply the above model. However, such a strategy would ignore the lattice structure and relate queries to keys that cannot possibly appear together according to the lattice.

[2] We have removed the trainable peakiness coefficient from the original formulation for simplicity and because gains of this additional parameter were unclear according to Sperber et al. (2017).

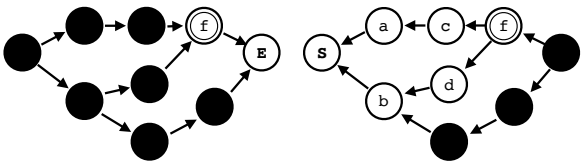


Figure 2: Example for binary masks in forward- and backward directions. The currently selected query is node `f`, and the mask prevents all solid black nodes from being attended to.

We find empirically that this naive approach performs poorly (§ 5.4). As a remedy, we introduce a masking scheme to incorporate lattice structure into the model (§ 4.1), before addressing positional encoding for lattices (§ 4.2).

4.1 Lattice Reachability Masks

We draw inspiration from prior works such as the TreeLSTM (Tai et al., 2015) and related works. Consider how the recurrent conditioning of hidden representations in these models is informed by the graph structure of the inputs: Each node is conditioned on its direct predecessors in the graph, and via recurrent modeling on all its predecessor nodes up to the root or leaf nodes.

4.1.1 Binary Masks

We propose a masking strategy that results in the same conditioning among tokens based on the lattice structure, preventing the self-attentional model from attending to lattice nodes that are not reachable from some given query node i. Figure 2 illustrates the concept of such reachability masks. Formally, we obtain masks in forward and backward direction as follows:

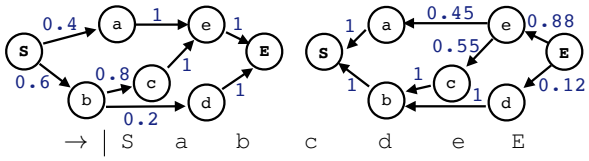
$$\overrightarrow{m}^{\text{bin}}_{ij} = \begin{cases} 0 & \text{if } i \in \mathcal{R}^{-}(j) \vee i{=}j \\ -\infty & \text{else} \end{cases}$$

$$\overleftarrow{m}^{\text{bin}}_{ij} = \begin{cases} 0 & \text{if } i \in \mathcal{R}^{+}(j) \vee i{=}j \\ -\infty & \text{else} \end{cases}$$

The resulting conditioning structure is analogous to the conditioning in lattice RNNs (Ladhak et al., 2016) in the backward and forward directions, respectively. These masks can be obtained using standard graph traversal algorithms.

4.1.2 Probabilistic Masks

Binary masks capture the graph structure of the inputs, but do not account for potentially available lattice scores that associate lattice nodes with a probability of being correct. Prior work has found



→	S	a	b	c	d	e	E
S	1	0.4	0.6	0.48	0.12	0.88	1
a	0	1	0	1	0	1	1
b	0	0	1	0.8	0.2	0.8	1
c	0	0	0	1	0	1	1
d	0	0	0	0	1	0	1
e	0	0	0	0	0	1	1
E	0	0	0	0	0	0	1

←	S	a	b	c	d	e	E
S	1	0	0	0	0	0	0
a	1	1	0	0	0	0	0
b	1	0	1	0	0	0	0
c	1	0	0	1	0	0	0
d	1	0	1	0	1	0	0
e	1	0.45	0.55	0.55	0	1	0
E	1	0.4	0.6	0.48	0.12	0.88	1

Figure 3: Example for pairwise conditional reaching probabilities for a given lattice, which we logarithmize to obtain self-attention masks. Rows are queries, columns are keys.

it critical to exploit lattice scores, especially for noisy inputs such as speech recognition lattices (Sperber et al., 2017). In fact, the previous binary masks place equal weight on all nodes, which will cause the influence of low-confidence regions (i.e., dense regions with many alternative nodes) on computed representations to be greater than the influence of high-confidence regions (sparse regions with few alternative nodes).

It is therefore desirable to make the self-attentional lattice model aware of these scores, so that it can place higher emphasis on confident context and lower emphasis on context with low confidence. The probabilistic masks below generalize binary masks according to this intuition:

$$\overrightarrow{m}^{\text{prob}}_{ij} = \begin{cases} \log p_G(j \succ i \mid i) & \text{if } i{\neq}j \\ 0 & \text{if } i{=}j \end{cases}$$

$$\overleftarrow{m}^{\text{prob}}_{ij} = \begin{cases} \log p_{G^\top}(j \succ i \mid i) & \text{if } i{\neq}j \\ 0 & \text{if } i{=}j \end{cases}$$

Here, we set $\log(0){:=}{-}\infty$. Figure 3 illustrates the resulting pairwise probability matrix for a given lattice and its reverse, prior to applying the logarithm. Note that the first row in the forward matrix and the last row in the backward matrix are the globally normalized scores of Equation 4.

Per our convention regarding $\log(0)$, the $-\infty$ entries in the mask will occur at exactly the same

Algorithm 1 Computation of logarithmized probabilistic masks via dynamic programming.
– given: DAG $G = (V, E)$; transition probs $p^{\text{trans}}_{k,j}$

$\forall i, j \in V : q_{i,j} \leftarrow 0$
for $i \in V$ **do** ▷ loop over queries
  $q_{i,i} \leftarrow 1$
  for $k \in \text{topologic-order}\,(V)$ **do**
    for next $\in \mathcal{N}^+(k)$ **do**
      $q_{i,\text{next}} \leftarrow q_{i,\text{next}} + p^{\text{trans}}_{\text{k,next}} \cdot q_{i,k}$
    end for
  end for
end for
$\forall i, j \in V : m^{\text{prob}}_{ij} \leftarrow \log q_{i,j}$

places as with the binary reachability mask, because the traversal probability is 0 for unreachable nodes. For reachable nodes, the probabilistic mask causes the computed similarity for low-confident nodes (keys) to be decreased, thus increasing the impact of confident nodes on the computed hidden representations. The proposed probabilistic masks are further justified by observing that the resulting model is invariant to path duplication (see Appendix A), unlike the model with binary masks.

The introduced probabilistic masks can be computed in $\mathcal{O}\left(|V|^3\right)$ from the given transition probabilities by using the dynamic programming approach described in Algorithm 1. The backward-directed probabilistic mask can be obtained by applying the same algorithm on the reversed graph.

4.1.3 Directional and Non-Directional Masks

The above masks are designed to be plugged into each Transformer layer via the masking term $\mathbf{M}$ in Equation 6. However, note that we have defined two different masks, $\overrightarrow{m}_{ij}$ and $\overleftarrow{m}_{ij}$. To employ both we can follow two strategies: (1) Merge both into a single, *non-directional* mask by using $\overleftrightarrow{m}_{ij} = \max\{\overrightarrow{m}_{ij}, \overleftarrow{m}_{ij}\}$. (2) Use half of the attention heads in each multi-head Transformer layer (§ 3.2) with forward masks, the other half with backward masks, for a *directional* strategy.

Note that when the input is a sequence (i.e., a lattice with only one complete path), the non-directional strategy reduces to unmasked sequential self-attention. The second strategy, in contrast, reduces to the directional masks proposed by Shen et al. (2018) for sequence modeling.

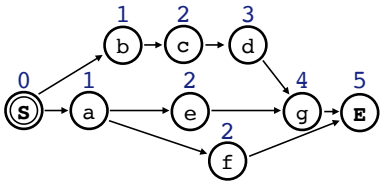


Figure 4: Lattice positions, computed as longest-path distance from the start node **S**.

4.2 Lattice Positional Encoding

Encoding positional information in the inputs is a crucial component in self-attentional architectures as explained in § 3.2. To devise a strategy to encode positions of lattice nodes in a suitable fashion, we state a number of desiderata: (1) Positions should be integers, so that positional embeddings (§ 3.2) can be used. (2) Every possible lattice path should be assigned strictly monotonically increasing positions, so that relative ordering can be inferred from positions. (3) For a compact representation, unnecessary jumps should be avoided. In particular, for at least one complete path the positions should increase by exactly 1 across all adjacent succeeding lattice nodes.

A naive strategy would be to use a topological order of the nodes to encode positions, but this clearly violates the compactness desideratum. Dyer et al. (2008) used shortest-path distances between lattice nodes to account for distortion, but this violates monotonicity. Instead, we propose using the longest-path distance (ldist) from the start node, replacing Equation 10 with:

$$\mathbf{x}'_i = \text{dropout}\left(\mathbf{x}_i + \text{embed}\left[\text{ldist}\left(\texttt{S} \rightarrow i\right)\right]\right).$$

This strategy fulfills all three desiderata, as illustrated in Figure 4. Longest-path distances from the start node to all other nodes can be computed in $\mathcal{O}\left(|V|^2\right)$ using e.g. Dijkstra's shortest-path algorithm with edge weights set to -1.

4.3 Computational Complexity

The computational complexity in the self-attentional encoder is dominated by generating the masks ($\mathcal{O}\left(|V|^3\right)$), or by the computation of pairwise similarities ($\mathcal{O}\left(|V|^2\right)$) if we assume that masks are precomputed prior to training. Our main baseline model, the LatticeLSTM, can be computed in $\mathcal{O}\left(|E|\right)$, where $|E| \leq |V|^2$. Nevertheless, constant factors and the effect of batched operations lead to considerably faster computations for the self-attentional approach in practice (§ 5.3).

5 Experiments

We examine the effectiveness of our method on a speech translation task, in which we directly translate decoding lattices from a speech recognizer into a foreign language.

5.1 Settings

We conduct experiments on the Fisher–Callhome Spanish–English Speech Translation corpus (Post et al., 2013). This corpus contains translated telephone conversations, along with speech recognition transcripts and lattices. The Fisher portion (138k training sentences) contains conversations between strangers, and the smaller Callhome portion (15k sentences) contains conversations between family members. Both and especially the latter are acoustically challenging, indicated by speech recognition word error rates of 36.4% and 65.3% on respective test sets for the transcripts contained in the corpus. The included lattices have oracle word error rates of 16.1% and 37.9%.

We use XNMT (Neubig et al., 2018) which is based on DyNet (Neubig et al., 2017a), with the provided self-attention example as a starting point.[3] Hidden dimensions are set to 512 unless otherwise noted. We use a single-layer LSTM-based decoder with dropout rate 0.5. All self-attentional encoders use three layers with hidden dimension of the FF operation set to 2048, and dropout rate set to 0.1. LSTM-based encoders use 2 layers. We follow Sperber et al. (2017) to tokenize and lowercase data, remove punctuation, and replace singletons with a special `unk` token. Beam size is set to 8.

For training, we find it important to pretrain on sequential data and finetune on lattice data (§ 5.6). This is in line with prior work (Sperber et al., 2017) and likely owed to the fact that the lattices in this dataset are rather noisy, hampering training especially during the early stages. We use Adam for training (Kingma and Ba, 2014). For sequential pretraining, we follow the learning schedule with warm-up and decay of Vaswani et al. (2017). Finetuning was sometimes unstable, so we finetune both using the warm-up/decay strategy and using a fixed learning rate of 0.0001 and report the better result. We use large-batch training with minibatch size of 1024 sentences, accumulated over 16 batched computations of 64 sentences each, due to memory constraints. Early stopping is applied when the BLEU score on a held-out validation set does not improve over 15 epochs, and the model with the highest validation BLEU score is kept.

[3] Our code is available: `http://msperber.com/research/acl-lattice-selfatt/`

Encoder model	Inputs	Fisher	Callh.
LSTM[4]	1-best	35.9	11.8
Seq. SA	1-best	35.71	12.36
Seq. SA (directional)	1-best	37.42	13.00
Graph attention	lattice	35.71	11.87
LatticeLSTM[4]	lattice	38.0	14.1
Lattice SA (proposed)	lattice	**38.73**	**14.74**

Table 1: BLEU scores on Fisher (4 references) and Callhome (1 reference), for proposed method and several baselines.

5.2 Main Results

Table 1 compares our model against several baselines. Lattice models tested on Callhome are pretrained on Fisher and finetuned on Callhome lattices (Fisher+Callhome setting), while lattice models tested on Fisher use a Fisher+Fisher training setting. All sequential baselines are trained on the reference transcripts of Fisher. The first set of baselines operates on 1-best (sequential) inputs and includes a bidirectional LSTM, an unmasked self-attentional encoder (SA) of otherwise identical architecture with our proposed model, and a variant with directional masks (Shen et al., 2018). Next, we include a graph-attentional model that masks all but adjacent lattice nodes (Veličković et al., 2018) but is otherwise identical to the proposed model, and a LatticeLSTM. Note that these lattice models both use the cross-attention lattice-score bias (§ 3.1).

Results show that our proposed model outperforms all examined baselines. Compared to the sequential self-attentional model, our models improves by 1.31–1.74 BLEU points. Compared to the LatticeLSTM, our model improves results by 0.64–0.73 BLEU points, while at the same time being more computationally efficient (§ 5.3). Graph attention is not able to improve over the sequential baselines on our task due to its restriction to local context.

Encoder	Training Batching	Training Speed	Inference Batching	Inference Speed
Sequential encoder models				
LSTM	M	4629	–	715
SA	M	**5021**	–	**796**
LatticeLSTM and lattice SA encoders				
LSTM	–	178	–	391
LSTM	A	710	A	538
SA	M	**2963**	–	687
SA	A	748	A	**718**

Table 2: Computation speed (words/sec), averaged over 3 runs. Batching is conducted manually (M), through autobatching (A), or disabled (–). The self-attentional lattice model displays superior speed despite using 3 encoder layers, compared to 2 layers for the LSTM-based models.

5.3 Computation Speed

The self-attentional lattice model was motivated not only by promising model accuracy (as confirmed above), but also by potential speed gains. We therefore test computation speed for training and inference, comparing against LSTM- and LatticeLSTM-based models. For fair comparison, we use a reimplementation of the LatticeLSTM so that all models are run with the exact same toolkits and identical decoder architectures. Again, LSTM-based models have two encoder layers, while self-attentional models have three layers. LatticeLSTMs are difficult to speed up through manually implemented batched computations, but similar models have been reported to strongly benefit from autobatching (Neubig et al., 2017b) which automatically finds operations that can be grouped after the computation graph has been defined. Autobatching is implemented in DyNet but not available in many other deep learning toolkits, so we test both with and without autobatching. Training computations are manually or automatically batched across 64 parallel sentences, while inference speed is tested for single sentences with forced decoding of gold translations and without beam search. We test with DyNet commit 8260090 on an Nvidia Titan Xp GPU and average results over three runs.

Table 2 shows the results. For sequential inputs, the self-attentional model is slightly faster than the LSTM-based model. The difference is perhaps smaller than expected, which can be explained by the larger number of layers in the self-attentional model, and the relatively short sentences of the Fisher corpus that reduce the positive effect of parallel computation across sequence positions. For lattice-based inputs, we can see a large speed-up of the self-attentional approach when no autobatching is used. Replacing manual batching with autobatching during training for the self-attentional model yields no benefits. Enabling autobatching at inference time provides some speed-up for both models. Overall, the speed advantage of the self-attentional approach is still very visible even with autobatching available.

[4]BLEU scores taken from Sperber et al. (2017).

reachability mask	dir.	prob.	latt. pos.	Fisher	Callh.
✓	✓	✓	✓	**38.73**	**14.74**
✓	✓	✓		38.25	12.45
✓	✓		✓	37.52	14.37
✓		✓	✓	35.49	12.83
	✓			30.58	9.41

Table 3: Ablation over proposed features, including reachability masks, directional (vs. non-directional) masking, probabilistic (vs. binary) masking, and lattice positions (vs. topological positions).

5.4 Feature Ablation

We next conduct a feature ablation to examine the individual effect of the improvements introduced in § 4. Table 3 shows that longest-path position encoding outperforms topological positions, the probabilistic approach outperforms binary reachability masks, and modeling forward and reversed lattices with separate attention heads outperforms the non-directional approach. Consistently with the findings by Sperber et al. (2017), lattice scores are more effectively exploited on Fisher than on Callhome as a result of the poor lattice quality for the latter. The experiment in the last row demonstrates the effect of keeping the lattice contents but removing all structural information, by rearranging nodes in linear, arbitrary topological order, and applying the best sequential model. Results are poor and structural information clearly beneficial.

5.5 Behavior At Test Time

To obtain a better understanding of the proposed model, we compare accuracies to the sequential

	Lattice oracle	1-best	Lattice
	Fisher		
Sequential SA	47.84	37.42	–
Lattice SA	47.69	37.56	38.73
	Callhome		
Sequential SA	17.94	13.00	–
Lattice SA	18.54	13.90	14.74

Table 4: Fisher and Callhome models, tested by inputting lattice oracle paths, 1-best paths, and full lattices.

self-attentional model when translating either lattice oracle paths, 1-best transcripts, or lattices. The lattice model translates sequences by treating them as lattices with only a single complete path and all transition probabilities set to 1. Table 4 shows the results for the Fisher+Fisher model evaluated on Fisher test data, and for the Fisher+Callhome model evaluated on Callhome test data. We can see that the lattice model outperforms the sequential model even when translating sequential 1-best transcripts, indicating benefits perhaps due to more robustness or increased training data size for the lattice model. However, the largest gains stem from using lattices at test time, indicating that our model is able to exploit the actual test-time lattices. Note that there is still a considerable gap to the translation of lattice oracles which form a top-line to our experiments.

5.6 Effect of Pretraining and Finetuning

Finally, we analyze the importance of our strategy of pretraining on clean sequential data before finetuning on lattice data. Table 5 shows the results for several combinations of pretraining and finetuning data. The first thing to notice is that pretraining is critical for good results. Skipping pretraining performs extremely poorly, while pretraining on the much smaller Callhome data yields results no better than the sequential baselines (§ 5.2). We conjecture that pretraining is beneficial mainly due to the rather noisy lattice training data, while for tasks with cleaner training lattices pretraining may play a less critical role.

The second observation is that for the finetuning stage, domain appears more important than data size: Finetuning on Fisher works best when testing on Fisher, while finetuning on Callhome works best when testing on Callhome, despite the Callhome finetuning data being an order of magnitude smaller. This is encouraging, because the collection of large amounts of training lattices can be difficult in practice.

Sequential data	Lattice data	Fisher	Callh.
–	Fisher	1.45	1.78
Callhome	Fisher	34.52	13.04
Fisher	Callhome	35.47	**14.74**
Fisher	Fisher	**38.73**	14.59

Table 5: BLEU scores for several combinations of Fisher (138k sentences) and Callhome (15k sentences) training data.

6 Related Work

The translation of lattices rather than sequences has been investigated with traditional machine translation models (Ney, 1999; Casacuberta et al., 2004; Saleem et al., 2004; Zhang et al., 2005; Matusov et al., 2008; Dyer et al., 2008), but these approaches rely on independence assumptions in the decoding process that no longer hold for neural encoder-decoder models. Neural lattice-to-sequence models were proposed by Su et al. (2017); Sperber et al. (2017), with promising results but slow computation speeds. Other related work includes gated graph neural networks (Li et al., 2016; Beck et al., 2018). As an alternative to these RNN-based models, GCNs have been investigated (Duvenaud et al., 2015; Defferrard et al., 2016; Kearnes et al., 2016; Kipf and Welling, 2017), and used for devising tree-to-sequence models (Bastings et al., 2017; Marcheggiani et al., 2018). We are not aware of any application of GCNs to lattice modeling. Unlike our approach, GCNs consider only local context, must be combined with slower LSTM layers for good performance, and lack support for lattice scores.

Our model builds on previous works on self-attentional models (Cheng et al., 2016; Parikh et al., 2016; Lin et al., 2017; Vaswani et al., 2017). The idea of masking has been used for various purposes, including occlusion of future information during training (Vaswani et al., 2017), introducing directionality (Shen et al., 2018) with good results for machine translation confirmed by Song et al. (2018), and soft masking (Im and Cho, 2017; Sperber et al., 2018). The only extension of self-attention beyond sequence modeling we

are aware of is graph attention (Veličković et al., 2018) which uses only local context and is outperformed by our model.

7 Conclusion

This work extended existing sequential self-attentional models to lattice inputs, which have been useful for various purposes in the past. We achieve this by introducing probabilistic reachability masks and lattice positional encodings. Experiments in a speech translation task show that our method outperforms previous approaches and is much faster than RNN-based alternatives in both training and inference settings. Promising future work includes extension to tree-structured inputs and application to other tasks.

Acknowledgments

The work leading to these results has received funding from the European Union under grant agreement no 825460.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffry E. Hinton. 2016. Layer Normalization. *arXiv:1607.06450*.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Representation Learning (ICLR)*, San Diego, USA.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *Association for Computational Linguistic (ACL)*, pages 273–283, Melbourne, Australia.

Francisco Casacuberta, Hermann Ney, Franz Josef Och, Enrique Vidal, J. M. Vilar, S. Barrachina, I. García-Varea, D. Llorens, C. Martínez, S. Molau, F. Nevado, M. Pastor, D. Picó, A. Sanchis, and C. Tillmann. 2004. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18(1):25–47.

Alberto Cetoli, Stefano Bragaglia, Andrew D. O'Harney, and Marc Sloan. 2017. Graph Convolutional Networks for Named Entity Recognition. In *International Workshop on Treebanks and Linguistic Theories (TLT16)*, pages 37–45, Prague, Czech Republic.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas, USA.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3844–3852, Barcelona, Spain.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2224–2232, Montréal, Canada.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing Word Lattice Translation. Technical Report LAMP-TR-149, University of Maryland, Institute For Advanced Computer Studies.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Neural Information Processing Systems Conference (NIPS)*, pages 1019–1027, Barcelona, Spain.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *International Conference on Machine Learning (ICML)*, Sydney, Australia.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770—-778, Las Vegas, USA.

Robert L. Hemminger and Lowell W. Beineke. 1978. Line graphs and line digraphs. In *Selected Topics in Graph Theory*, pages 271–305. Academic Press Inc.

Jinbae Im and Sungzoon Cho. 2017. Distance-based Self-Attention Network for Natural Language Inference. *arXiv:1712.02047*.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular Graph Convolutions: Moving Beyond Fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608.

Diederik P. Kingma and Jimmy L. Ba. 2014. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, Banff, Canada.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations (ICLR)*.

Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister. 2016. LatticeRnn: Recurrent Neural Networks over Lattices. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 695–699, San Francisco, USA.

Yujia Li, Richard Zemel, Mark Brockschmeidt, and Daniel Tarlow. 2016. Gated Graph Sequence Neural Networks. In *International Conference on Learning Representations (ICLR)*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *International Conference on Representation Learning (ICLR)*, Toulon, France.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting Semantics in Neural Machine Translation with Graph Convolutional Networks. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 486–492, New Orleans, USA.

Evgeny Matusov, Björn Hoffmeister, and Hermann Ney. 2008. ASR word lattice translation with exhaustive reordering is possible. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2342–2345, Brisbane, Australia.

Roberto Navigli and Paola Velardi. 2010. Learning Word-Class Lattices for Definition and Hypernym Extraction. In *Association for Computational Linguistic (ACL)*, pages 1318–1327, Uppsala, Sweden.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017a. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980*.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017b. On-the-fly Operation Batching in Dynamic Computation Graphs. In *Neural Information Processing Systems Conference (NIPS)*, Long Beach, USA.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang. 2018. XNMT: The eXtensible Neural Machine Translation Toolkit. In *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase*, Boston, USA.

Hermann Ney. 1999. Speech Translation: Coupling of Recognition and Translation. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 517–520, Phoenix, USA.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2249–2255, Austin, USA.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved Speech-to-Text Translation with the Fisher and Callhome Spanish–English Speech Translation Corpus. In *International Workshop on Spoken Language Translation (IWSLT)*, Heidelberg, Germany.

Shirin Saleem, Szu-Chen Jou, Stephan Vogel, and Tanja Schultz. 2004. Using Word Lattice Information for a Tighter Coupling in Speech Translation Systems. In *International Conference on Spoken Language Processing (ICSLP)*, pages 41–44, Jeju Island, Korea.

Anna Senina, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *German Conference on Pattern Recognition (GCPR)*, pages 184–195, Münster, Germany. Springer.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding. In *Conference on Artificial Intelligence (AAAI)*, New Orleans, USA.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, Seattle, USA.

Kaitao Song, Xu Tan, Furong Peng, and Jianfeng Lu. 2018. Hybrid Self-Attention Network for Machine Translation. *arXiv:1811.00253v2*.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural Lattice-to-Sequence Models for Uncertain Inputs. In *Conference on*

Empirical Methods in Natural Language Processing (EMNLP), pages 1380–1389, Copenhagen, Denmark.

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. 2018. Self-Attentional Acoustic Models. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Hyderabad, India.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-Based Recurrent Neural Network Encoders for Neural Machine Translation. In *Conference on Artificial Intelligence (AAAI)*, pages 3302–3308, San Francisco, USA.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, USA.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Association for Computational Linguistic (ACL)*, pages 1556–1566, Beijing, China.

Shikhar Vashishth, Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. Dating Documents using Graph Convolution Networks. In *Association for Computational Linguistic (ACL)*, pages 1605–1615, Melbourne, Australia.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Neural Information Processing Systems Conference (NIPS)*, pages 5998–6008, Long Beach, USA.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*, Vancouver, Canada.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In *Neural Information Processing Systems Conference (NIPS)*, Montréal, Canada.

Ruiqiang Zhang, Genichiro Kikui, Hirofumi Yamamoto, and Wai-Kit Lo. 2005. A Decoding Algorithm for Word Lattice Translation in Speech Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 23–29, Pittsburgh, USA.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long Short-Term Memory Over Recursive Structures. In *International Conference on Machine Learning (ICML)*, pages 1604–1612, Lille, France.

A Path Duplication Invariance

Figure 5 shows a sequential lattice, and a lattice derived from it but with a duplicated path. Semantically, both are equivalent, and should therefore result in identical neural representations. Note that while in practice duplicated paths should not occur, paths with partial overlap are quite frequent. It is therefore instructive to consider this hypothetical situation. Below, we demonstrate that the binary masking approach (§ 4.1.1) is biased such that computed representations are impacted by path duplication. In contrast, the probabilistic approach (§ 4.1.2) is invariant to path duplication.

We consider the example of Figure 5, discussing only the forward direction, because the lattice is symmetric and computations for the backward direction are identical. We follow notation of Equations 1 through 3, using $\langle \mathtt{a}, \mathtt{b} \rangle$ as abbrevation for $f\left(q\left(\mathbf{x}_{\mathtt{a}}\right), k\left(\mathbf{x}_{\mathtt{b}}\right)\right)$ and $\mathbf{v}_{\mathtt{a}}$ to abbreviate $v(\mathbf{x}_{\mathtt{a}})$. Let us consider the computed representation for the node S as query. For the sequential lattice with binary mask, it is:

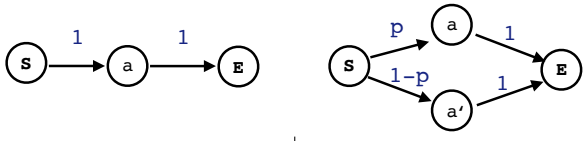
$$\mathbf{y}_{\mathtt{S}} = \frac{1}{C}\left(e^{\langle \mathtt{S},\mathtt{S}\rangle}\mathbf{v}_{\mathtt{S}} + e^{\langle \mathtt{S},\mathtt{a}\rangle}\mathbf{v}_{\mathtt{a}} + e^{\langle \mathtt{S},\mathtt{b}\rangle}\mathbf{v}_{\mathtt{b}}\right) \quad (11)$$

Here, C is the softmax normalization term that ensures that exponentiated similarities sum up to 1.

In contrast, the lattice with duplication results in a doubled influence of $\mathbf{v}_{\mathtt{a}}$:

$$\begin{aligned}\mathbf{y}_{\mathtt{S}} =& \frac{1}{C}\Big(e^{\langle \mathtt{S},\mathtt{S}\rangle}\mathbf{v}_{\mathtt{S}} + e^{\langle \mathtt{S},\mathtt{a}\rangle}\mathbf{v}_{\mathtt{a}} \\ &+ e^{\langle \mathtt{S},\mathtt{a}'\rangle}\mathbf{v}_{\mathtt{a}'} + e^{\langle \mathtt{S},\mathtt{E}\rangle}\mathbf{v}_{\mathtt{E}}\Big) \\ =& \frac{1}{C}\left(e^{\langle \mathtt{S},\mathtt{S}\rangle}\mathbf{v}_{\mathtt{S}} + 2e^{\langle \mathtt{S},\mathtt{a}\rangle}\mathbf{v}_{\mathtt{a}} + e^{\langle \mathtt{S},\mathtt{E}\rangle}\mathbf{v}_{\mathtt{E}}\right).\end{aligned}$$

The probabilistic approach yields the same result as the binary approach for the sequential lattice (Equation 11). For the lattice with path duplication, the representation for the node S is computed as follows:



sequential	S	a	E
S	1	1	1
a	0	1	1
E	0	0	1

duplicated	S	a	a′	E
S	1	p	$(1-p)$	1
a	0	1	0	1
a′	0	0	1	1
E	0	0	0	1

Figure 5: A sequential lattice, and a variant with a duplicated path, where nodes a and a′ are labeled with the same word token. The matrices contain pairwise reaching probabilities in forward direction, where rows are queries, columns are keys.

$$\begin{aligned}\mathbf{y}_{\mathtt{S}} =& \frac{1}{C}\Big(e^{\langle \mathtt{S},\mathtt{S}\rangle}\mathbf{v}_{\mathtt{S}} + e^{\langle \mathtt{S},\mathtt{a}\rangle+\log p}\mathbf{v}_{\mathtt{a}} \\ &+ e^{\langle \mathtt{S},\mathtt{a}'\rangle+\log(1-p)}\mathbf{v}_{\mathtt{a}'} + e^{\langle \mathtt{S},\mathtt{E}\rangle}\mathbf{v}_{\mathtt{E}}\Big) \\ =& \frac{1}{C}\Big(e^{\langle \mathtt{S},\mathtt{S}\rangle}\mathbf{v}_{\mathtt{S}} + e^{\langle \mathtt{S},\mathtt{a}\rangle}e^{\log p}\mathbf{v}_{\mathtt{a}} \\ &+ e^{\langle \mathtt{S},\mathtt{a}'\rangle}e^{\log(1-p)}\mathbf{v}_{\mathtt{a}'} + e^{\langle \mathtt{S},\mathtt{E}\rangle}\mathbf{v}_{\mathtt{E}}\Big) \\ =& \frac{1}{C}\Big(e^{\langle \mathtt{S},\mathtt{S}\rangle}\mathbf{v}_{\mathtt{S}} + pe^{\langle \mathtt{S},\mathtt{a}\rangle}\mathbf{v}_{\mathtt{a}} \\ &+ (1-p)e^{\langle \mathtt{S},\mathtt{a}'\rangle}\mathbf{v}_{\mathtt{a}'} + e^{\langle \mathtt{S},\mathtt{E}\rangle}\mathbf{v}_{\mathtt{E}}\Big) \\ =& \frac{1}{C}\Big(e^{\langle \mathtt{S},\mathtt{S}\rangle}\mathbf{v}_{\mathtt{S}} + e^{\langle \mathtt{S},\mathtt{a}\rangle}\mathbf{v}_{\mathtt{a}} + e^{\langle \mathtt{S},\mathtt{E}\rangle}\mathbf{v}_{\mathtt{E}}\Big).\end{aligned}$$

The result is the same as in the semantically equivalent sequential case (Equation 11), the computation is therefore invariant to path duplication. The same argument can be extended to other queries, to other lattices with duplicated paths, as well as to the lattice-biased encoder-decoder attention.

B Qualitative Analysis

We conduct a manual inspection and showcase several common patterns in which the lattice input helps improve translation quality, as well as one counter example. In particular, we compare the outputs of the sequential and lattice models according to the 3rd and the last row in Table 1, on Fisher.

B.1 Example 1

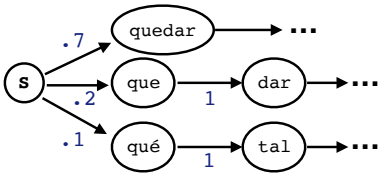
In this example, the ASR 1-best contains a bad word choice (*quedar* instead of *qué tal*). The correct word is in the lattice, and can be disambiguated by exploiting long-range self-attentional encoder context.

gold transcript: *Qué tal, eh, yo soy Guillermo, ¿Cómo estás?*

ASR 1-best: *quedar eh yo soy guillermo cómo estás*

seq2seq output: *stay eh i ' m guillermo how are you*

ASR lattice:



lat2seq output: *how are you eh i ' m guillermo how are you*

B.2 Example 2

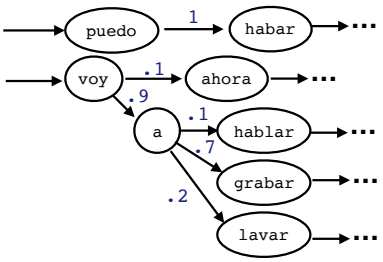
Here, the correct word *graduar* does not appear in the lattice, instead the lattice offers many incorrect alternatives of high uncertainty. The translation model evidently goes with a linguistically plausible guess, ignoring the source side.

gold transcript: *Claro Es, eh, eh, o sea, yo me, me voy a graduar con un título de esta universidad.*

ASR 1-best: *claro existe eh o sea yo me me puedo habar con un título esta universidad*

seq2seq output: *sure it exists i mean i can talk with a title*

ASR lattice:



lat2seq output: *sure i mean i ' m going to take a university title*

B.3 Example 3

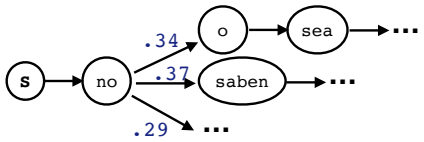
In this example, *o sea* (*I mean*) appears with slightly lower confidence than *saben* (*they know*), but is chosen for a more natural sounding target sentence

gold transcript: *No, o sea, eso es eh, clarísimo para mi*

ASR 1-best: *no saben eso es eh clarísimo para mi*

seq2seq output: *they don ' t know that ' s eh sure for me*

ASR lattice:



lat2seq output: *no i mean that ' s very clear for me*

B.4 Counter Example

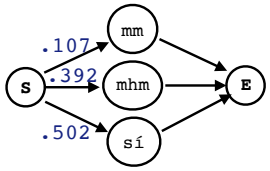
In this counter example, the translation model gets confused from the additional and wrong lattice context and no longer produces the correct output.

gold transcript: *sí*

ASR 1-best: *sí*

seq2seq output: *yes*

ASR lattice:



lat2seq output: *mm*