Neural Speech Translation using Lattice Transformations and Graph Networks

Daniel Beck[†] **Trevor Cohn**[†] **Gholamreza Haffari**[‡]
[†]School of Computing and Information Systems
University of Melbourne, Australia
{d.beck,t.cohn}@unimelb.edu.au
[‡]Faculty of Information Technology
Monash University, Australia
gholamreza.haffari@monash.edu

Abstract

Speech translation systems usually follow a pipeline approach, using word lattices as an intermediate representation. However, previous work assume access to the original transcriptions used to train the ASR system, which can limit applicability in real scenarios. In this work we propose an approach for speech translation through lattice transformations and neural models based on graph networks. Experimental results show that our approach reaches competitive performance without relying on transcriptions, while also being orders of magnitude faster than previous work.

1 Introduction

Translation from speech utterances is a challenging problem that has been studied both under statistical, symbolic approaches (Ney, 1999; Casacuberta et al., 2004; Kumar et al., 2015) and more recently using neural models (Sperber et al., 2017). Most previous work rely on pipeline approaches, using the output of a speech recognition system (ASR) as an input to a machine translation (MT) one. These inputs can be simply the 1-best sentence returned by the ASR system or a more structured representation such as a lattice.

Some recent work on end-to-end systems bypass the need for intermediate representations, with impressive results (Weiss et al., 2017). However, such a scenario has drawbacks. From a practical perspective, it requires access to the original speech utterances and transcriptions, which can be unrealistic if a user needs to employ an out-of-the-box ASR system. From a theoretical perspective, intermediate representations such as lattices can be enriched through external, textual resources such as monolingual corpora or dictionaries.

Sperber et al. (2017) proposes a lattice-to-sequence model which, in theory, can address both problems above. However, their model suffers from training speed performance due to the lack of efficient batching procedures and they rely on transcriptions for pretraining. In this work, we address these two problems by applying lattice transformations and graph networks as encoders. More specifically, we enrich the lattices by applying subword segmentation using byte-pair encoding (Sennrich et al., 2016, BPE) and perform a minimisation step to remove redundant nodes arising from this procedure. Together with the standard batching strategies provided by graph networks, we are able to decrease training time by two orders of magnitude, enabling us to match their translation performance under the same training speed constraints *without relying on gold transcriptions.*

2 Approach

Many graph network options exist in the literature (Bruna et al., 2014; Duvenaud et al., 2015; Kipf and Welling, 2017; Gilmer et al., 2017): in this work we opt for a Gated Graph Neural Network (Li et al., 2016, GGNN), which was recently incorporated in an encoder-decoder architecture by Beck et al. (2018). Assume a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, L_\mathcal{V}, L_\mathcal{E}\}$, where $\mathcal{V}$ is a set of nodes (v, ℓ_v), $\mathcal{E}$ is a set of edges (v_i, v_j, ℓ_e) and $L_\mathcal{V}$ and $L_\mathcal{E}$ are respectively vocabularies for nodes and edges, from which node and edge labels (ℓ_v and ℓ_e) are defined. Given an input graph with node embeddings $\mathbf{X}$, a GGNN is defined as

$$\mathbf{h}_v^0 = \mathbf{x}_v$$
$$\mathbf{r}_v^t = \sigma\left(c_v^r \sum_{u \in \mathcal{N}_v} \mathbf{W}_{\ell_e}^r \mathbf{h}_u^{(t-1)} + \mathbf{b}_{\ell_e}^r\right)$$
$$\mathbf{z}_v^t = \sigma\left(c_v^z \sum_{u \in \mathcal{N}_v} \mathbf{W}_{\ell_e}^z \mathbf{h}_u^{(t-1)} + \mathbf{b}_{\ell_e}^z\right)$$

$$\widetilde{\mathbf{h}}_v^t = \rho\left(c_v \sum_{u\in\mathcal{N}_v} \mathbf{W}_{\ell_e}\left(\mathbf{r}_u^t \odot \mathbf{h}_u^{(t-1)}\right) + \mathbf{b}_{\ell_e}\right)$$
$$\mathbf{h}_v^t = (1-\mathbf{z}_v^t)\odot \mathbf{h}_v^{(i-1)} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}}_v^t$$

where $e = (u, v, \ell_e)$ is the edge between nodes u and v, $\mathcal{N}(v)$ is the set of neighbour nodes for v, ρ is a non-linear function, σ is the sigmoid function and $c_v = c_v^z = c_v^r = |\mathcal{N}_v|^{-1}$ are normalisation constants.

Intuitively, a GGNN reduces to a GRU (Cho et al., 2014) if the graph is a linear chain. Therefore, the GGNN acts as a generalised encoder that updates nodes according to their neighbourhood. Multiple layers can be stacked, allowing information to be propagated through longer paths in the graph. Batching can be done by using adjacency matrices and matrix operations to perform the updates, enabling efficient processing on a GPU.

2.1 Lattice Transformations

As pointed out by Beck et al. (2018), GGNNs can suffer from parameter explosion when the edge label space is large, as the number of parameters is proportional to the set of edge labels. This is a problem for lattices, since most of the information is encoded on the edges. We tackle this problem by transforming the lattices into their corresponding *line graphs*, which swaps nodes and edges.[1] After this transformation, we also add start and end symbols, which enable the encoder to propagate information through all possible paths in the lattice. Importantly, we also remove node scores from the lattice in most of our experiments, but we do revisit this idea in §3.3.

Having lattices as inputs allow us to incorporate additional steps of textual transformations. To showcase this, in this work we perform subword segmentation on the lattice nodes using BPE. If a node is not present in the subword vocabulary, we split it into subwords and connect them in a left-to-right manner.

The BPE segmentation can lead to redundant nodes in the lattice. Our next transformation step is a minimisation procedure, where such nodes are joined into a single node in the graph. To perform this step, we leverage an efficient algorithm for automata minimisation (Hopcroft, 1971), which traverses the graph detecting redundant nodes by using equivalence classes, running in $O(n \log n)$ time, where n is the number of nodes.

[1]This procedure is also done in Sperber et al. (2017).

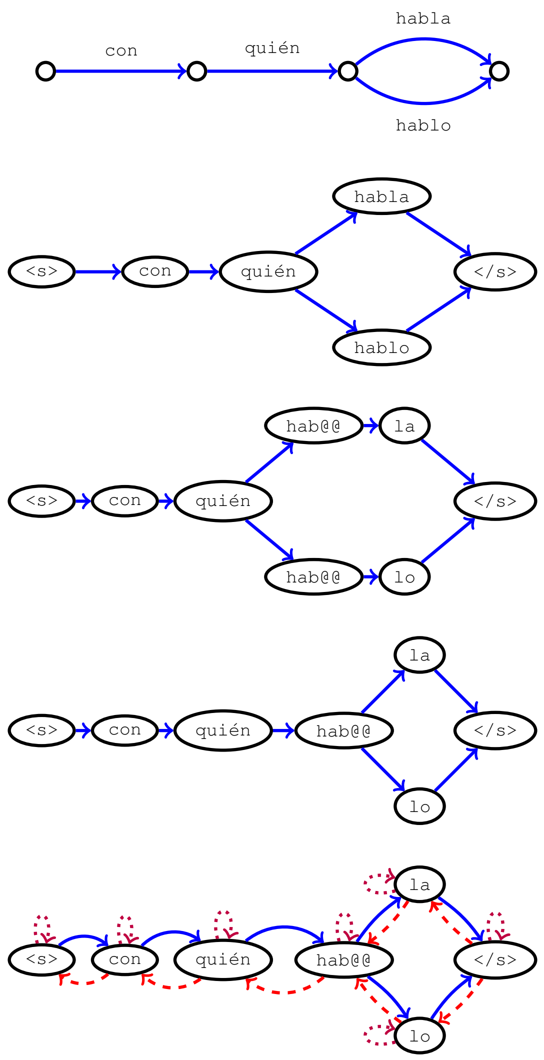

Figure 1: Proposed lattice transformations. From top to bottom: 1) Original lattice with scores removed; 2) Line graph transformation; 3) Subword segmentation; 4) Lattice minimisation; 5) Addition of reverse and self-loop edges.

The final step adds reverse and self-loop edges to the lattice, where these new edges have specific parameters in the encoder. This eases propagation of information and is standard practice when using graph networks as encoders (Marcheggiani and Titov, 2017; Bastings et al., 2017; Beck et al., 2018). We show an example of all the transformation steps on Figure 1.

In Figure 2 we show the architecture of our system, using the final lattice from Figure 1 as an example. Nodes are represented as embeddings that are updated according to the lattice structure, resulting in a set of hidden states as the output. Other components follow a standard `seq2seq` model, using a bilinear attention module (Luong et al., 2015) and a 2-layer LSTM (Hochreiter and

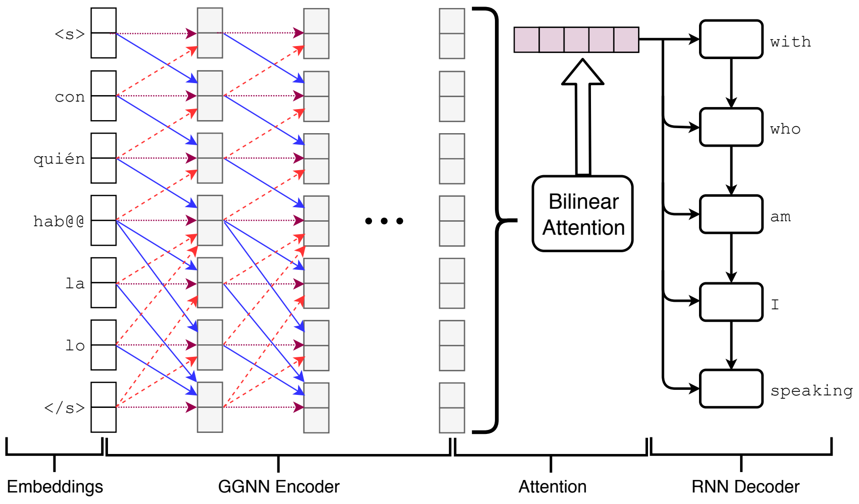

Figure 2: Model architecture, using the final Spanish lattice from Figure 1 and its corresponding English translation as an example.

Schmidhuber, 1997) as the decoder.

3 Experiments

Data We perform experiments using the Fisher/Callhome Speech Translation corpus, composed of Spanish telephone conversations with their corresponding English translations. We use the original release by Post et al. (2013), containing both 1-best and pruned lattice outputs from an ASR system for each Spanish utterance.[2] The Fisher corpus contain 150K instances and we use the original splits provided with the datasets. Following previous work (Post et al., 2013; Sperber et al., 2017), we lowercase and remove punctuation from the English translations. To build the BPE models, we extract the vocabulary from the Spanish training lattices, using 8K split operations.

Models and Evaluation All our models are trained on the Fisher training set. For the 1-best baseline we use a standard `seq2seq` architecture and for the GGNN models, we use the same setup as Beck et al. (2018). Our implementation is based on the Sockeye toolkit (Hieber et al., 2017) and we use default values for most hyperparameters, except for batch size (16) and GGNN layers (8).[3] For regularisation, we apply 0.5 dropout on the input embeddings and perform early stopping on the corresponding Fisher dev set.

[2]We refer the reader to Post et al. (2013) for details on the ASR system and how the lattices were generated.

[3]A complete description of hyperparameter values is available in the Supplementary Material.

	1-best	L	L+S	L+S+M
Median	32.4	34.4	34.5	34.3
Ensemble	36.1	38.3	38.7	**39.1**

Table 1: Out-of-the-box scenario results, in BLEU scores. “L” corresponds to word lattice inputs, “L+S” and “L+S+M” correspond to lattices after subword segmentation and after minimisation, respectively.

Each model is trained using 5 different seeds and we report BLEU (Papineni et al., 2001) results using the median performance according to the dev set and an ensemble of the 5 models. For the word-based models, we remove any tokens with frequency lower than 2 (as in Sperber et al. (2017)), while for subword models we do not perform any threshold pruning. We report all results on the Fisher “dev2” set.[4]

3.1 Out-of-the-box ASR scenario

In this scenario we assume only lattices and 1-best outputs are available, simulating a setting where we do not have access to the transcriptions. Table 1 shows that results are consistent with previous work: lattices provide significant improvements over simply using the 1-best output. More importantly though, the results also highlight the benefits of our proposed transformations and we obtain the best ensemble performance using minimised lattices.

[4]We also experimented with the Callhome test set, similar to previous work. However, we did not see any different trends so we omit the results.

	L+S+M	L+S+M+T
Median	34.3	37.1
Ensemble	39.1	**42.3**
Previous Work - no lattice scores		
Sperber et al. (2017)	–	36.9
Previous Work - with lattice scores		
Post et al. (2013)	–	36.8
Sperber et al. (2017)	–	38.5

Table 2: Results with transcriptions, in BLEU scores. "L+S+M" corresponds to the same results in Table 1 and "L+S+M+T" is the setting with gold transcriptions added to the training set.

3.2 Adding Transcriptions

The out-of-the-box results in §3.1 are arguably more general in terms of applicability in real scenarios. However, in order to compare with the state-of-the-art, we also experiment with a scenario where we have access to the original Spanish transcriptions. To incorporate transcriptions into our model, we convert them into a linear chain graph, after segmenting using BPE. With this, we can simply take the union of transcriptions and lattices into a single training set. We keep the dev and test sets with lattices only, as this emulates test time conditions.

The results shown in Table 2 are consistent with previous work: adding transcriptions further enhance the system performance. We also slightly outperform Sperber et al. (2017) in the setting where they ignore lattice scores, as in our approach. Most importantly, we are able to reach those results while being two orders of magnitude faster at training time: Sperber et al. (2017) report taking 1.5 days for each epoch while our architecture can process each epoch in 15min. The reason is because their model relies on the CPU while our GGNN-based model can be easily batched and computed in a GPU.

Given those differences in training time, it is worth mentioning that the best model in Sperber et al. (2017) is surpassed by our best ensemble *using lattices only*. This means that we can obtain state-of-the-art performance even in an out-of-the-box scenario, under the same training speed constraints. While there are other constraints that may be considered (such as parameter budget), we nevertheless believe this is an encouraging result for real world scenarios.

3.3 Adding Lattice Scores

Our approach is not without limitations. In particular, the GGNN encoder ignores lattice scores, which can help the model disambiguate between different paths in the lattice. As a simple first approach to incorporate scores, we embed them using a multilayer perceptron, using the score as the input. This however did not produce good results: performance dropped to 32.9 BLEU in the single model setting and 38.4 for the ensemble.

It is worth noticing that Sperber et al. (2017) has a more principled approach to incorporate scores: by modifying the attention module. This is arguably a better choice, since the scores can directly inform the decoder about the ambiguity in the lattice. Since this approach does not affect the encoder, it is theoretically possible to combine our GGNN encoder with their attention module, we leave this avenue for future work.

4 Conclusions and Future Work

In this work we proposed an architecture for lattice-to-string translation by treating lattices as general graphs and leveraging on recent advances in neural networks for graphs.[5] Compared to previous similar work, our model permits easy mini-batching and allows one to freely enrich the lattices with additional information, which we exploit by incorporating BPE segmentation and lattice minimisation. We show promising results and outperform baselines in speech translation, particularly in out-of-the-box ASR scenarios, when one has no access to transcriptions.

For future work, we plan to investigate better approaches to incorporate scores in the lattices. The approaches used by Sperber et al. (2017) can provide a starting point in this direction. The same minimisation procedures we employ can be adapted to weighted lattices (Eisner, 2003). Another important avenue is to explore this approach in low-resource scenarios such as ones involving endangered languages (Adams et al., 2017; Anastasopoulos and Chiang, 2018).

Acknowledgements

This work was supported by the Australian Research Council (DP160102686). The research reported in this paper was partly conducted at the 2017 Frederick Jelinek Memorial Summer

[5]Code to replicate results available at `github.com/beckdaniel/textgraphs2019_lat2seq`.

Workshop on Speech and Language Technologies, hosted at Carnegie Mellon University and sponsored by Johns Hopkins University with unrestricted gifts from Amazon, Apple, Facebook, Google, and Microsoft.

References

Oliver Adams, Trevor Cohn, Graham Neubig, and Alexis Michaud. 2017. Phonemic transcription of low-resource tonal languages. In *Proceedings of ALTA*, pages 53–60.

Antonis Anastasopoulos and David Chiang. 2018. Tied Multitask Learning for Neural Speech Translation. In *Proceedings of NAACL*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Proceedings of EMNLP*, pages 1947–1957.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *Proceedings of ACL*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Proceedings of ICLR*, page 14.

Francisco Casacuberta, Hermann Ney, Franz Josef Och, Enrique Vidal, Juan Miguel Vilar, Sergio Barrachina, Ismael García-Varea, David Llorens, Carlos D. Martínez, Sirko Molau, Francisco Nevado, Moisés Ángeles Pastor, David Picó, Alberto Sanchis, and Christoph Tillmann. 2004. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18(1):25–47.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*, pages 1724–1734.

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 *Proceedings of NIPS*, pages 2215–2223.

Jason Eisner. 2003. Simpler and More General Minimization for Weighted Finite-State Automata. In *Proceedings of NAACL*, pages 64–71.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of ICML*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *arXiv preprint*, pages 1–18.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

John Hopcroft. 1971. An O(n log n) Algorithm for Minimizing States in a Finite Automaton. *Theory of machines and computations*.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR*.

Gaurav Kumar, Graeme Blackwood, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. 2015. A Coarse-Grained Model for Optimal Coupling of ASR and SMT Systems for Speech Translation. In *Proceedings of EMNLP*, pages 1902–1907.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated Graph Sequence Neural Networks. In *Proceedings of ICLR*, 1, pages 1–20.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*, pages 1412–1421.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of EMNLP*.

Hermann Ney. 1999. Speech Translation: Coupling of Recognition and Translation. In *Proceedings of ICASSP*, pages 517–520.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

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 *Proceedings of IWSLT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL*, pages 1715–1725.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural Lattice-to-Sequence Models for Uncertain Inputs. In *Proceedings of EMNLP*, pages 1380–1389.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. In *Proceedings of INTERSPEECH*, pages 2625–2629.