NEURAL INFORMATION PROCESSING SYSTEMS

Star Temporal Classification

Sequence Modeling with Partially Labeled data

Vineel Pratap[1], Awni Hannun[2], Gabriel Synnaeve[1], Ronan Collobert[1]

[1] Meta AI [2] Zoom AI

Overview

1. We develop an algorithm which can learn from partially labeled and unsegmented sequential data - Star Temporal Classification (STC)

2. STC uses a special star token to allow alignments which include all possible tokens whenever a token could be missing

3. STC is implemented using GTN, a framework for automatic differentiation with WFSTs

4. We demonstrate the effectiveness of STC on Automatic Speech Recognition (ASR) and Handwriting Recognition (HWR) tasks

Introduction

- Partially labeled data
- Practical Scenarios
- Quantifying label drop

Partially Labeled Data

- Incomplete labels for the samples in the data set
- Number of missing words nor their positions in the label sequence are not known in advance
- Weakly supervised temporal classification - the alignment of input sequence with the target labels is not known in advance

Input: 

Original Label: seen from an airplane the island looks like a big spider

Partial Label: from airplane the a spider

An example of speech with a complete and a partial label.

Practical scenarios

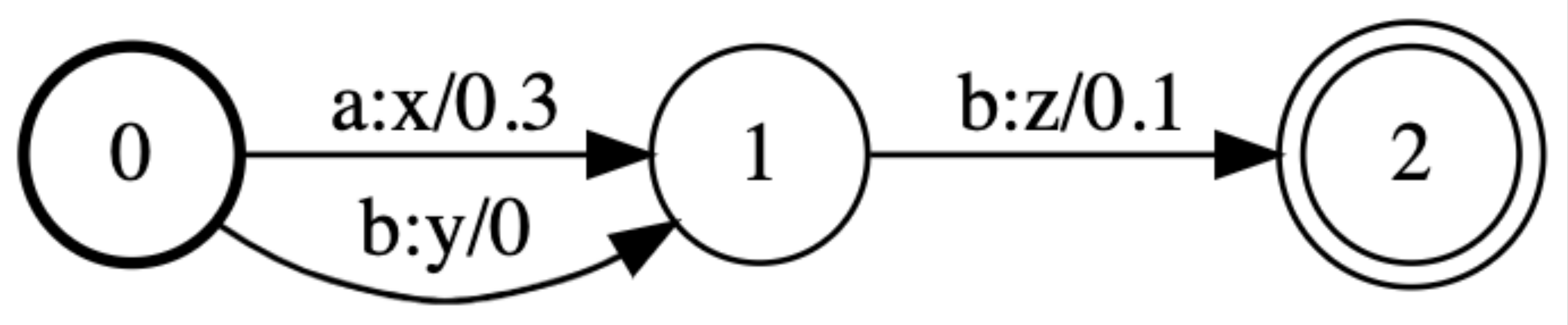
- Semi-supervised learning or transfer learning : Consider high confidence tokens in the label sequence and discard rest of the tokens
- Partial alignment : beginning and ending tokens are missing; subproblem
- Transcribing datasets : randomly label parts of the dataset and use them as partial labels
- Texts with some sections damaged to the point of illegibility.

Method

- WFST
- CTC; CTC as WFST composition
- STC

Weighted Finite-State Transducer (WFST)

- Encodes a mapping from input sequence to output sequence with a corresponding score

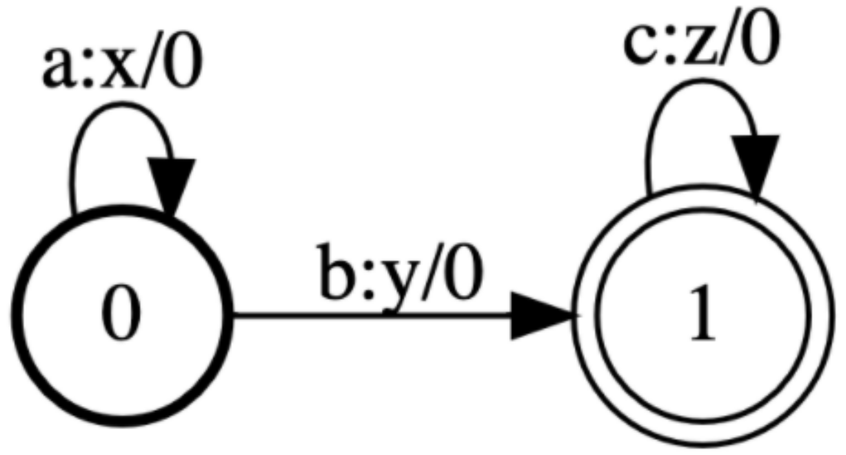


- A simple WFST which transduces ***ab*** → ***xz*** and ***bb*** → ***yz***
  - The score of ***ab*** → ***xz*** is 0.4
  - The score of ***bb*** → ***yz*** is 0.1

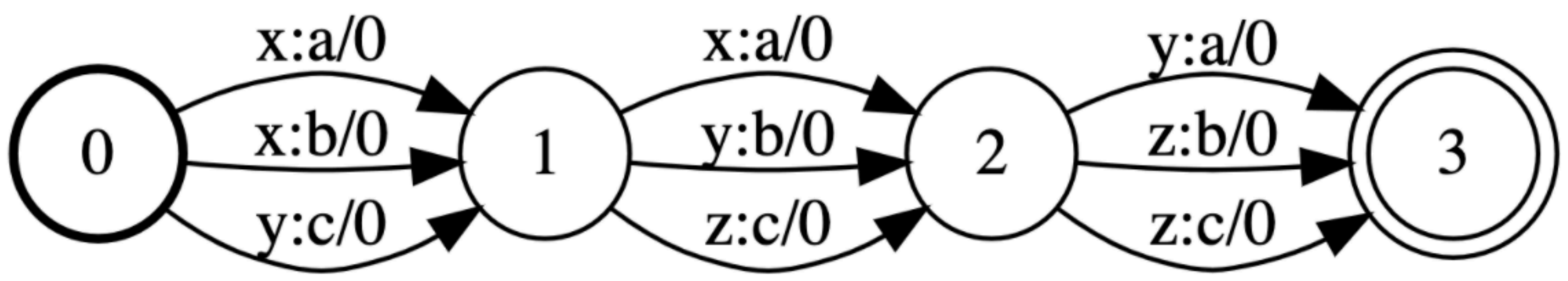
WFST Operations: Compose

- If x → y in the first graph and y → z in the second graph then x → z in the composed graph
- The score of the composed path is the sum of the scores of the paths in the input graphs

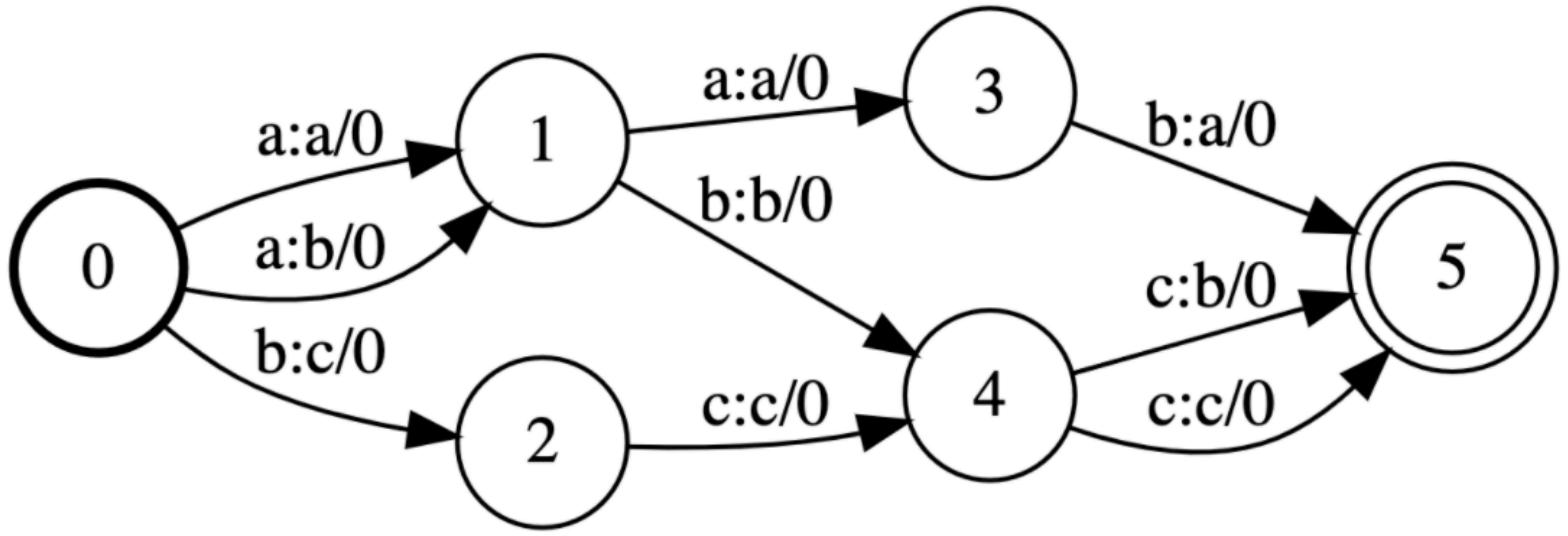
Graph g1



Graph g2



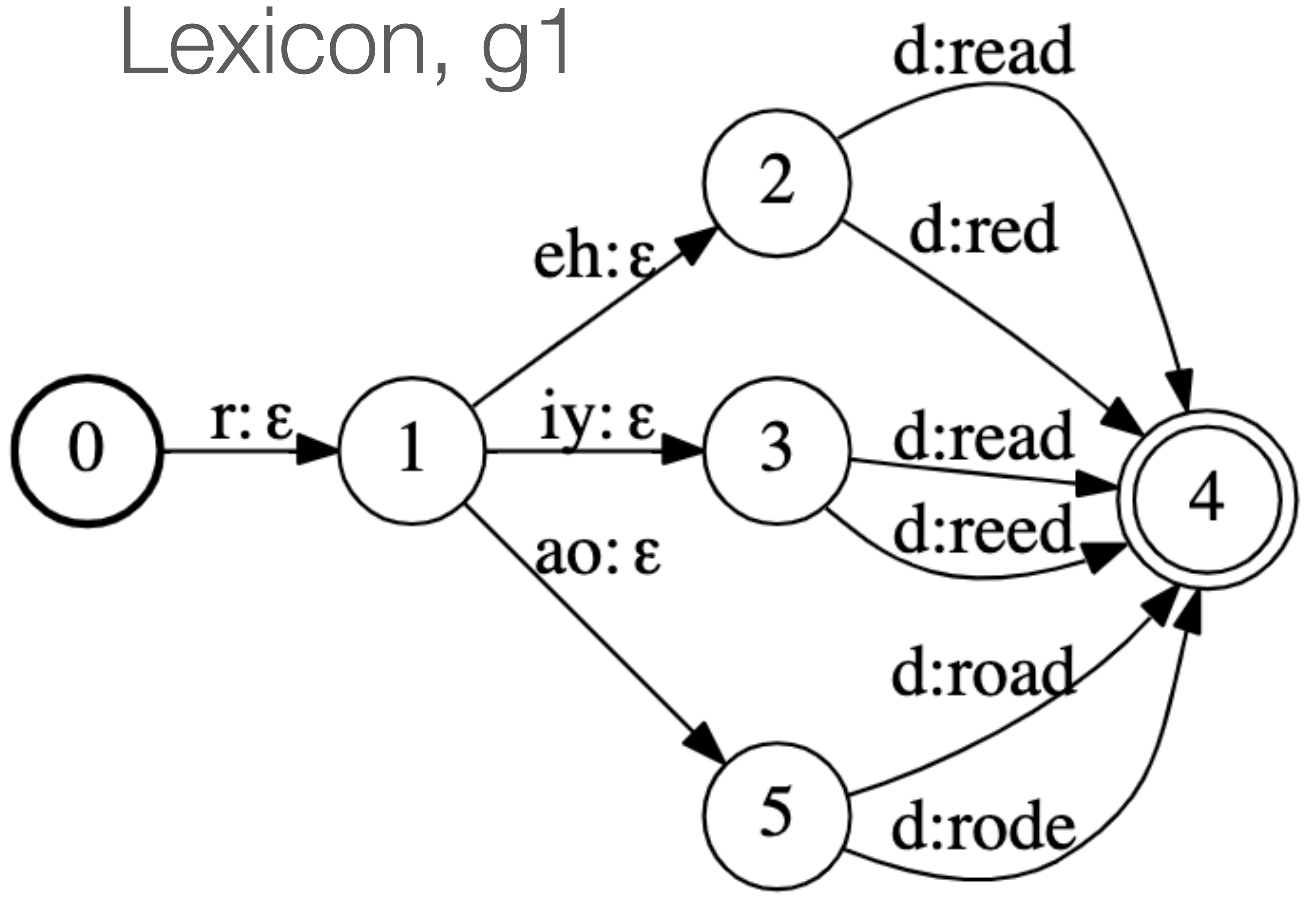
Compose(g1, g2)



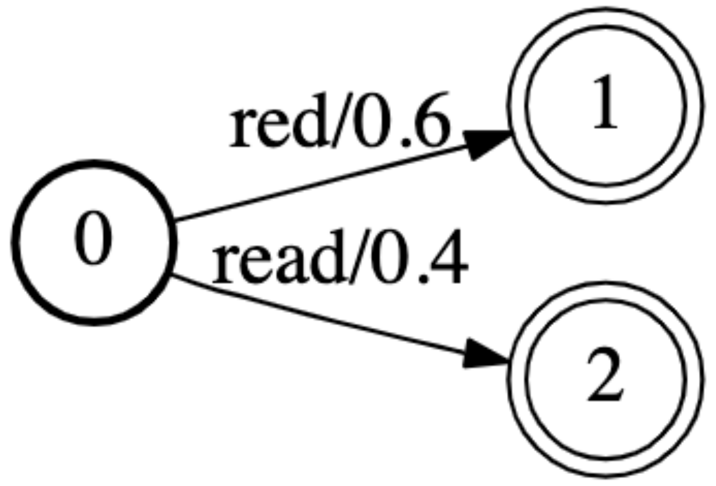
WFST Operations: Compose

- Can be used to combine graphs from different modalities

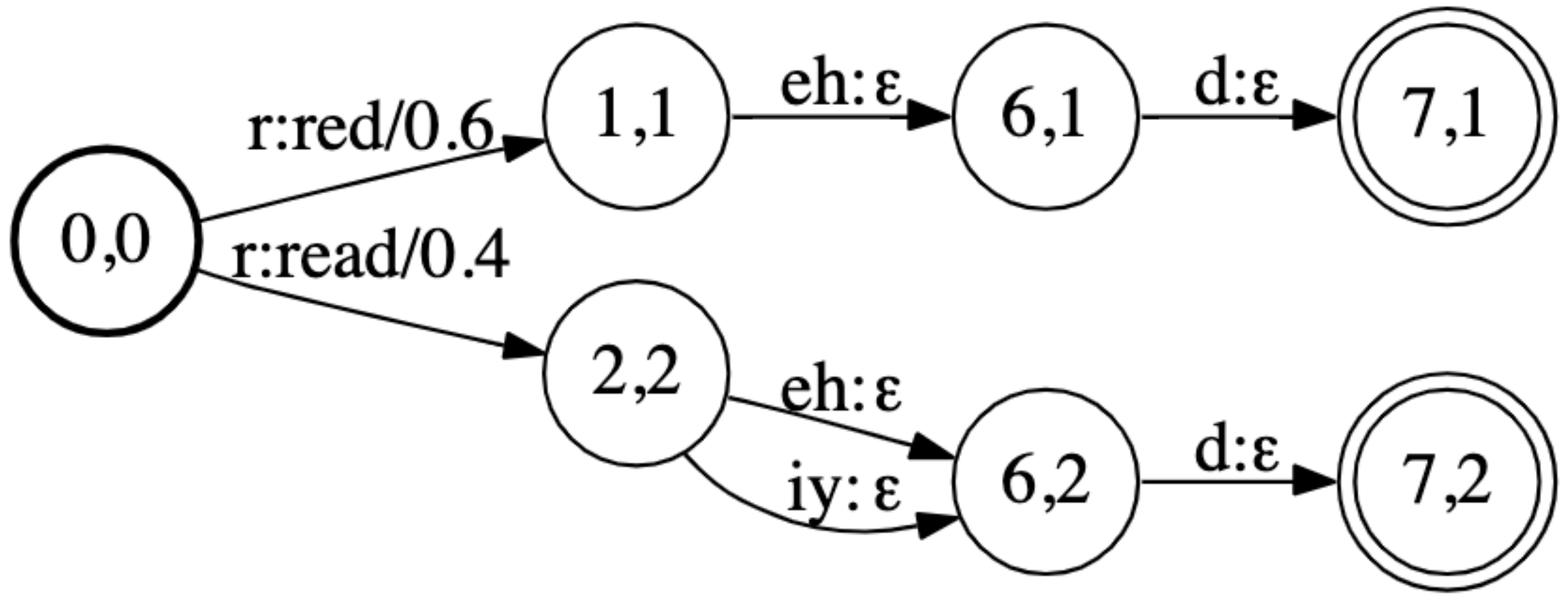
Lexicon, g1



Grammar, g2

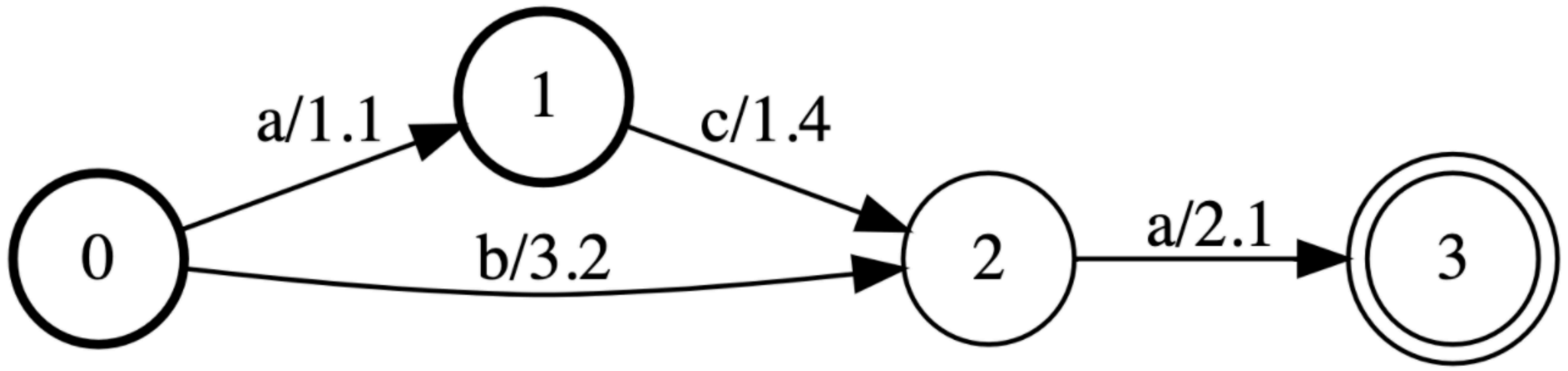


Compose(g1, g2)



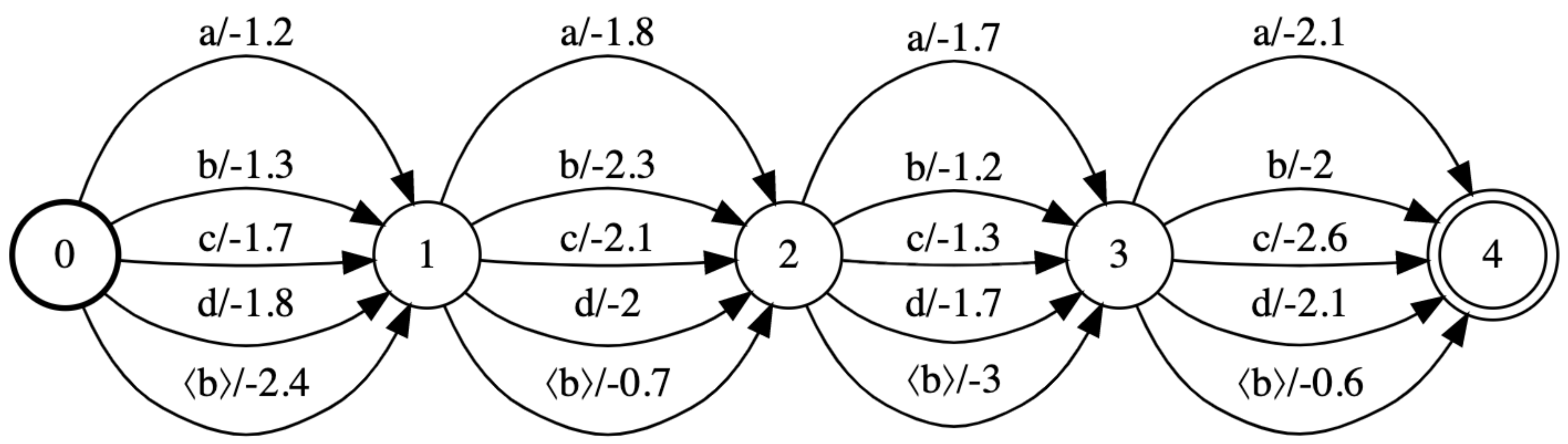
WFST Operations: Forward Score

- Accumulate (log-sum-exp) the score of all possible paths

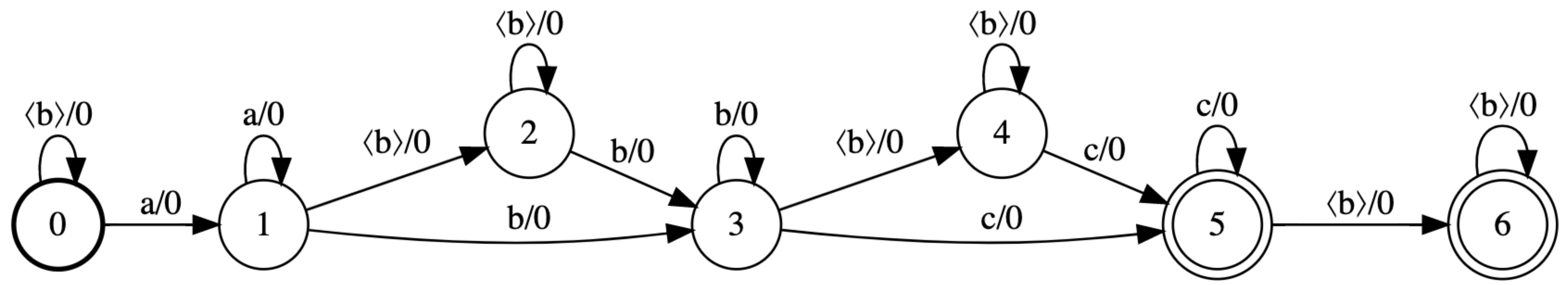


- The graph accepts three paths
    - aca with score =1.1+1.4+2.1
    - ba with score = 3.2+2.1
    - ca with score = 1.4+2.1
- forwardScore(g) is the log-sum-exp of the path scores

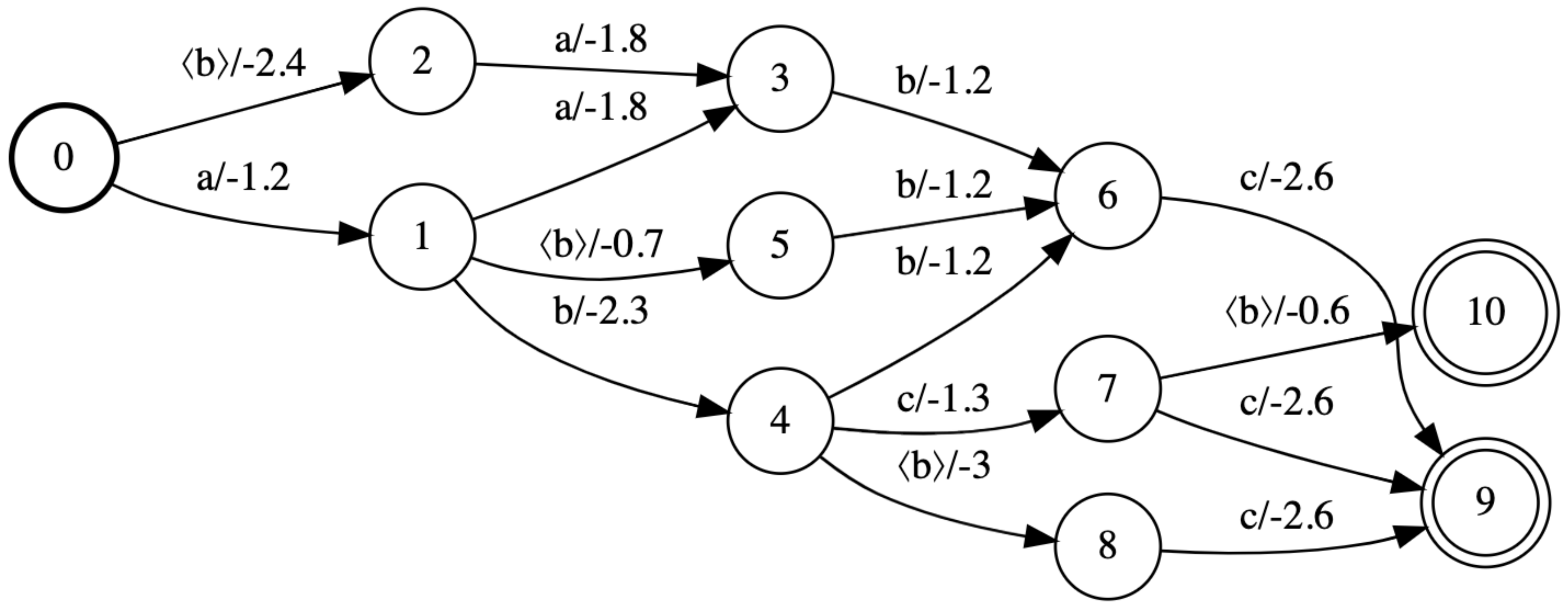
CTC using WFST Operations



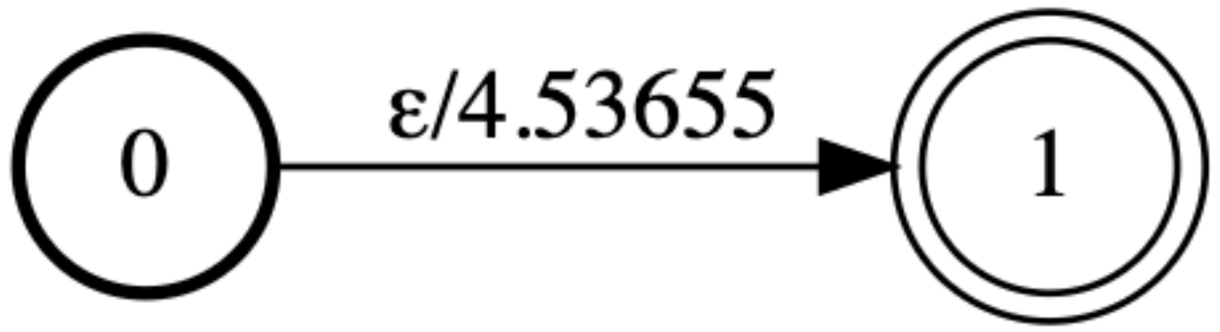
Emissions graph constructed from the log probabilities over the alphabet A = {a, b, c, d} and blank symbol <b>.



The CTC label graph corresponding to the target sequence (a, b, c)



Compose label graph with emission graph to get all valid paths which collapse to the target sequence



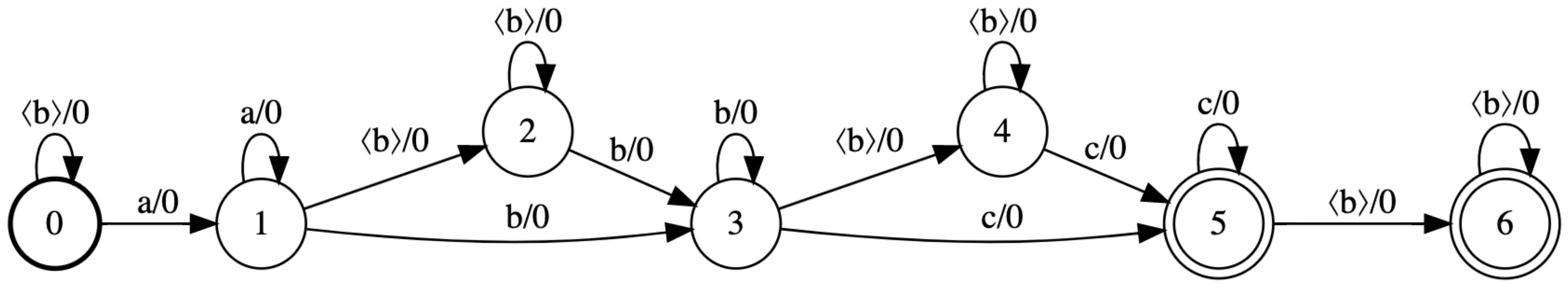
Sum the probabilities of all of the valid paths (in log-space) and negate the result to yield the CTC loss.

From CTC to STC

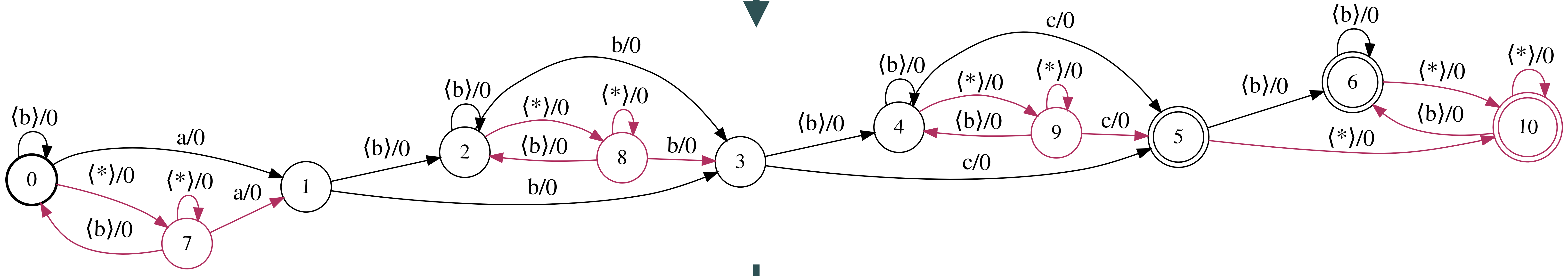
- star token <s> - allows every token in the alphabet

Example showing the collapsing of tokens in alphabet A = {a, b, c, d} to star token, <s>

From CTC to STC



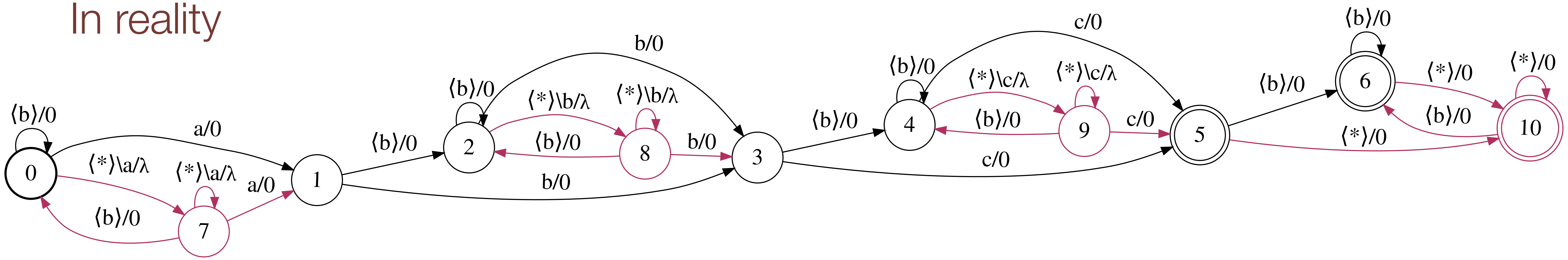
a b c



.* a .* b .* c .*

STC allows for zero or more tokens between any two tokens in the partial label.

In reality



[^a]* a [^b]* b [^c]* c .*

avoid counting the same alignment multiple times;

λ - token insertion penalty

GTN - A framework for differentiable WFSTs

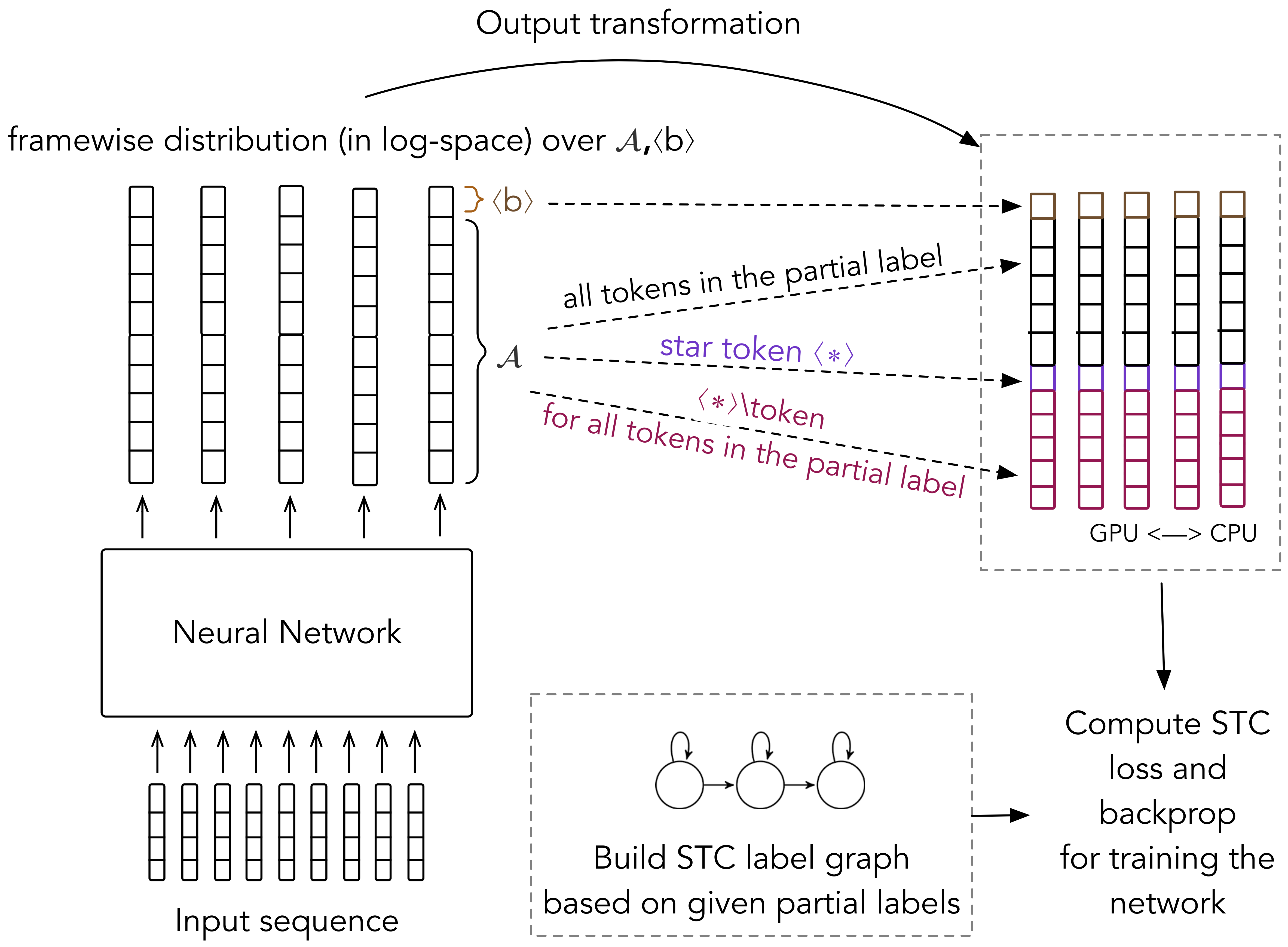
- Most of the WFST operations are differentiable with respect to the arc weights of the input graphs.
- This allows WFSTs to be used dynamically to train neural networks
- STC implemented using GTN
-

```
import gtn

gA = gtn.Graph()
gB = gtn.Graph()

gCompose = gtn.compose(gA, gB)
gFwd = gtn.forward_score(gCompose)

gtn.backward(gFwd)
gA_grad = gA.grad()
```

Implementation details

- Uses the CPU-based WFST algorithms from GTN
- Multiple threads are used to compute STC loss in parallel for all of the examples in a batch.
- Neural network model is run on GPU ⇒
  - emissions needed by STC must be copied to the CPU
  - the STC gradients must be copied back to GPU.
- Reduce the amount of data transfer between the CPU and the GPU
  - Transfer values corresponding only to the tokens present in the partial labels and the star tokens since the gradients corresponding to all other tokens are zero.

The STC pipeline



Experimental Results

- Automatic Speech Recognition
- Handwriting Recognition

Experiments

Automatic Speech Recognition (ASR)

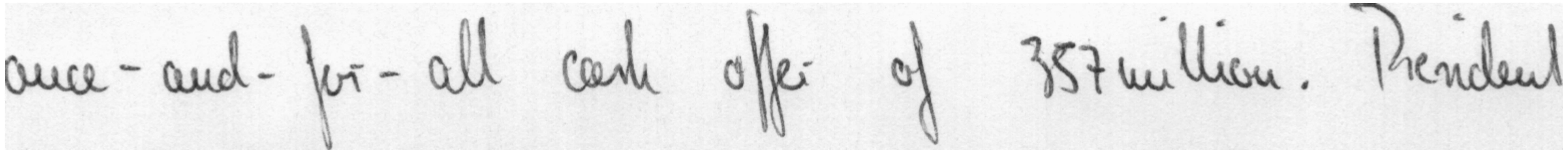
- LibriSpeech, a large-scale (1000 hours) corpus of read English speech.
- Word-level label drop

Handwriting Recognition (HWR)

- IAM DB, a database of handwritten English text lines
- Character-level label drop

Quantifying the label drop

- pDrop - the probability of dropping a token from the transcript label
- higher pDrop ⇒ higher number of missing tokens in the label and vice-versa.



True Label : once-and-for-all|cash|offer|of|357million|.|President

***pDrop* = 0.1 :** oc-andforall|cah|offer|o|357million|.|Preident

***pDrop* = 0.3 :** once-nd-fralcasr|of|357milon|Presidnt

***pDrop* = 0.7 :** nc-llaf|7mlioPreit

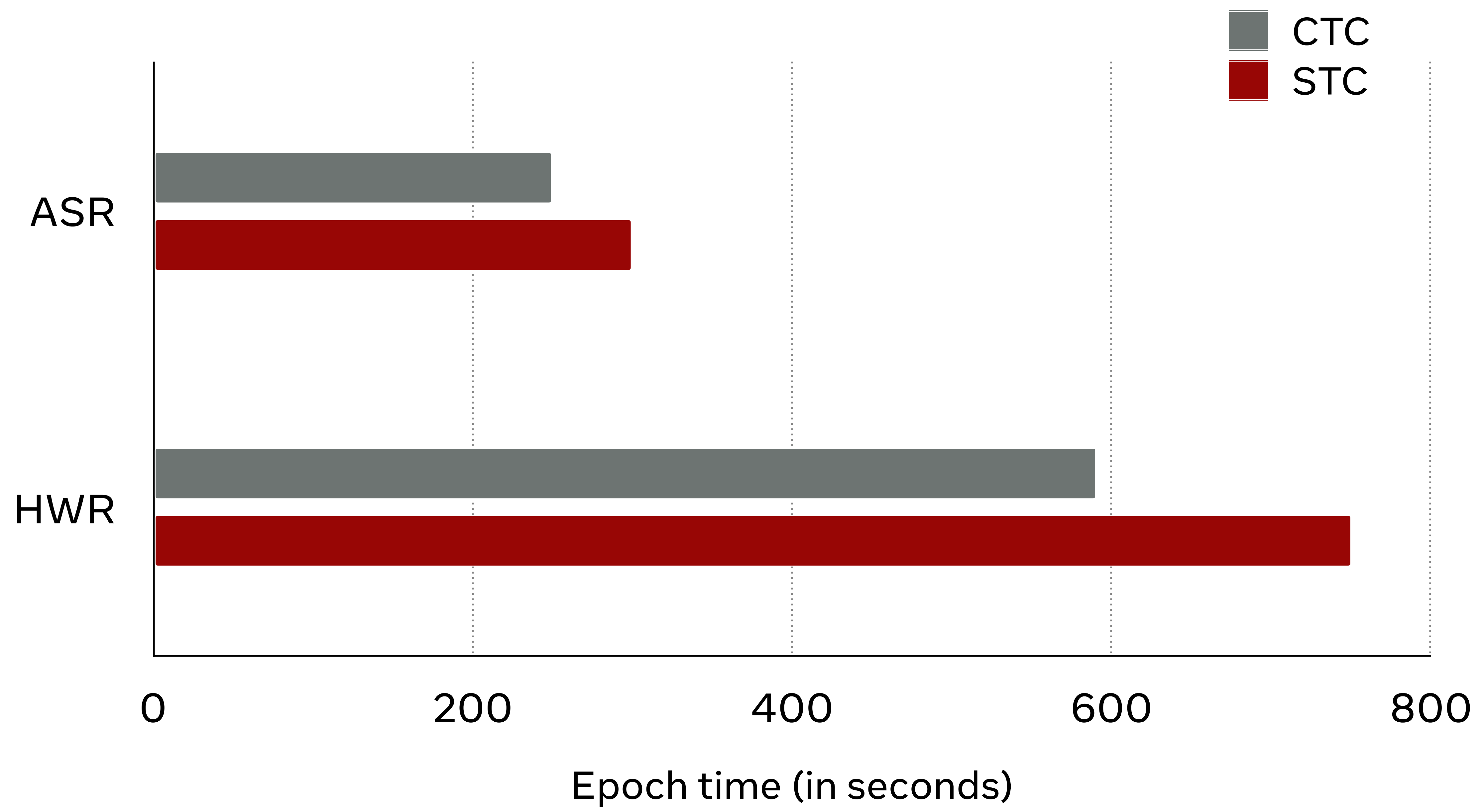
Automatic Speech Recognition

Method	Criterion	test-clean WER	test-other WER
Supervised Baseline	CTC	2.1	4.5
pDrop=0.1	CTC	4.5	8.1
	STC	3.3	6.9
	STC → Pseudo labels → CTC	2.7	5.2
pDrop=0.4	CTC	48.2	56.9
	STC	3.6	7.7
	STC → Pseudo labels → CTC	2.9	5.4
pDrop=0.7	CTC	100	100
	STC	5.1	11.1
	STC → Pseudo labels → CTC	3.1	6.2

Handwriting Recognition

Method	Criterion	test CER
Supervised Baseline	CTC	5.4
pDrop=0.1	CTC	7.7
	STC	7.2
pDrop=0.3	CTC	11.6
	STC	8.1
pDrop=0.7	CTC	78.5
	STC	26.7

Performance



More details on the work

- Paper - https://openreview.net/forum?id=ldRyJb_cjXa
- STC Implementation - https://github.com/facebookresearch/gtn_applications/
- GTN framework - `pip install gtn`