Streaming SLU
And Wake up word Detection

A Streaming End-to-End Framework For Spoken Language Understanding

- the intent is identified 'online', which means one intention is identified whenever sufficient evidence has been accumulated, without the need to wait until the end of the utterance, which significantly reduce the latency;
- the intent is identified 'incremental', which means multiple intentions can be identified sequentially. To the best of the authors' knowledge, this is the first streaming end-to-end architecture for the SLU task.

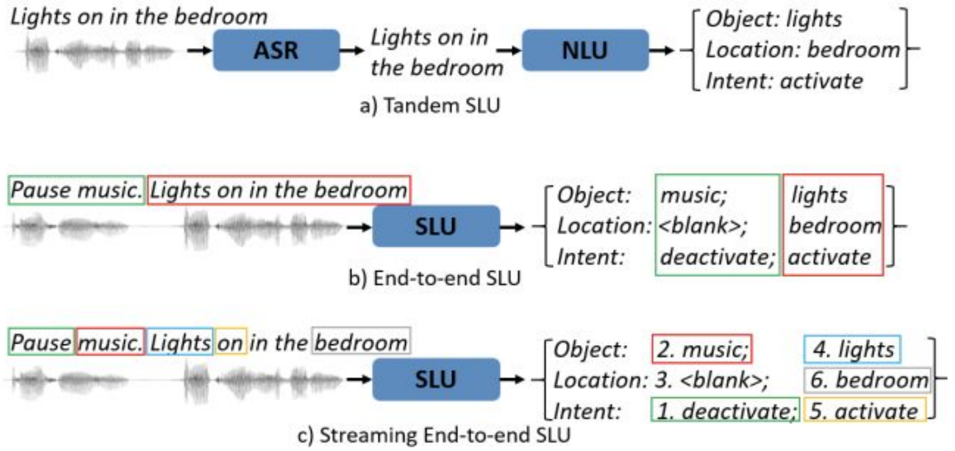

Lights on in the bedroom
ASR
Lights on in the bedroom
NLU
Object: lights
Location: bedroom
Intent: activate
a) Tandem SLU
Pause music. Lights on in the bedroom
SLU
Object: music; lights
Location: <blank>; bedroom
Intent: deactivate; activate
b) End-to-end SLU
Pause music. Lights on in the bedroom
SLU
Object: 2. music; 4. lights
Location: 3. <blank>; 6. bedroom
Intent: 1. deactivate; 5. activate
c) Streaming End-to-end SLU

Figure 1: Intent classification based on (a) a tandem SLU system, comprised of automatic speech recognition and natural language understanding, capable of processing only a single intent at a time; (b) an end-to-end SLU that classifies multiple intents from the speech signal, enabling multi-turn dialogue; (c) a streaming end-to-end SLU which incrementally predicts intents while processing incoming chunks of the speech signal.

A Streaming End-to-End Framework For Spoken Language Understanding

- Modelling
  - a unidirectional recurrent neural network (RNN), trained with the connectionist temporal classification (CTC) method, is proposed.
  - CTC optimization can help our framework to perform multi-intent classification without requiring prior alignment knowledge between the input and output sequences
  - CTC makes the model to fire an intent if sufficient evidence from the speech signal has been accumulated, and once it is fired, the evidence should be accumulated again. By this setting, multiple intents can be sequentially identified, no matter how long the speech signal is.
  - Do ASR pretraining
  - Pretrain SLU model using Cross entropy Loss before training STreaming with CTC

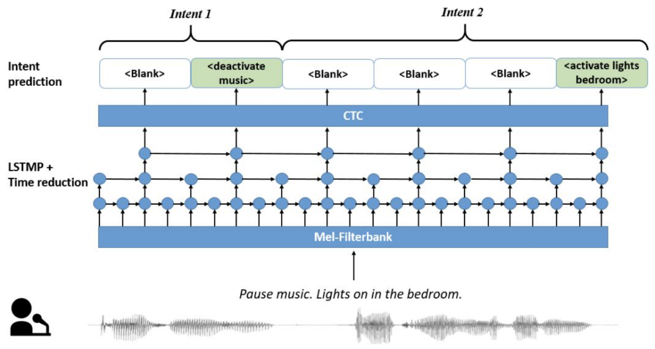
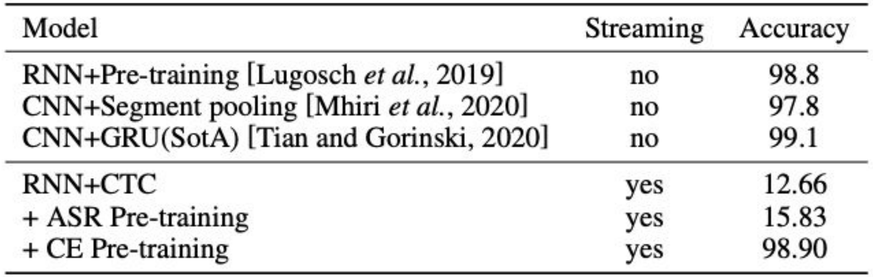


Figure 2: The proposed RNN-CTC based model enables the output to be updated in real-time as sufficient evidence is accumulated from the speech signal, hence allowing the identification of multiple intentions.

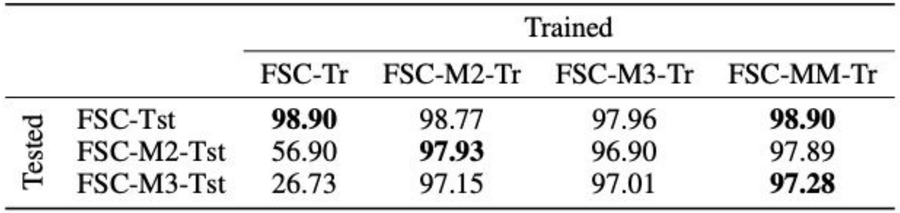
A Streaming End-to-End Framework For Spoken Language Understanding

- Evaluation
  - Single-intent classification on the FSC.
    - This result is comparable to the performance of the state-of-the-art non-streaming models, but is achieved in an online and incremental way.
  - Second, to evaluate the proposed framework on multiintent settings, we generate versions of the FSC dataset containing speech samples with multiple intents.



Model	Streaming	Accuracy
RNN+Pre-training [Lugosch *et al.*, 2019]	no	98.8
CNN+Segment pooling [Mhiri *et al.*, 2020]	no	97.8
CNN+GRU(SotA) [Tian and Gorinski, 2020]	no	99.1
RNN+CTC	yes	12.66
+ ASR Pre-training	yes	15.83
+ CE Pre-training	yes	98.90

Table 1: Experimental results on FSC for single-intent classification. Performance is reported in terms of accuracy (%).



		Trained			
		FSC-Tr	FSC-M2-Tr	FSC-M3-Tr	FSC-MM-Tr
Tested	FSC-Tst	**98.90**	98.77	97.96	**98.90**
	FSC-M2-Tst	56.90	**97.93**	96.90	97.89
	FSC-M3-Tst	26.73	97.15	97.01	**97.28**

Table 2: Experimental results for the multi-intent prediction. Performance is reported in terms of accuracy (%).

A Streaming End-to-End Framework For Spoken Language Understanding

- Evaluation
  - Google Speech Commands (GSC) dataset - Close to SOTA
  - Early spotting is observed particularly when sentence length is high. Early spotting refers to when intent is predicted before we have all evidence to predict intent
    - Treat the last non-silence frame corresponding to an intention as its ground truth position

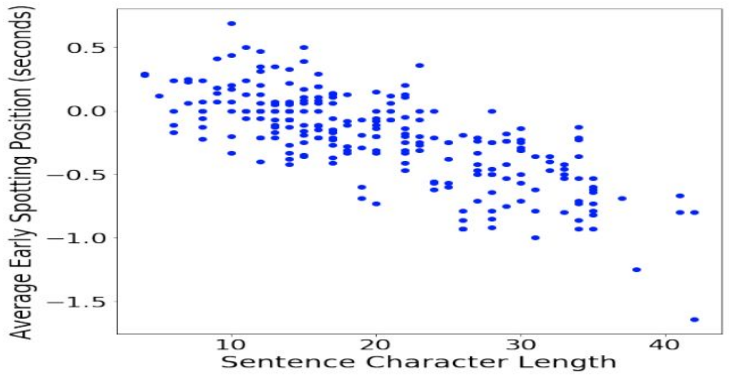


Figure 5: The relation between the early-spotting behavior and the sentence length in characters. The x-axis represents the sentence length, and the y-axis represents the relative spotting position.

RNN Based Incremental Online Spoken Language Understanding

- In this work, the online incremental SLU processing is setup along with (i) detection of utterance boundaries, and (ii) presence of erroneous ASR transcriptions.
    - computational complexity and memory constraints, most ASRs typically operate by chunking and processing the speech in segments referred to as end -pointing
- The proposed system is capable of recognizing intents on an arbitrarily long sequence of words with no sentence or utterance demarcations.
- Incremental Processing for ASR
    - intermediate querying of ASR output transcript is possible. This involves computation of the best path over intermediate, incomplete decoded lattices. Although, there is a possibility of the best path deviating between the complete and incomplete lattice decoding, the deviation is expected to be minimum in robust ASR systems.
    - In this scenario, the downstream application has no constraints of waiting until end-pointing and is free to process the ASR transcripts in an incremental manner. This also allows for online processing for downstream application in addition to the ASR itself for optimal latency.

Incremental Processing for SLU

- the NLU has to deal with the errors from: (i) ASR, (ii) end-point detection, and (iii) ASR errors due to sub-optimal end-pointing
- Propose an SLU system under Scenario 2, where the NLU module can also be run in an online fashion, in parallel with the ASR. Additionally, also propose incremental processing independent of ASR end-pointing and a lexical EOS detection module for utterance boundaries, operating on the ASR output.
- This comes with the advantage that the NLU has to deal with errors from only the ASR phase. However, note that the EOS module might still introduce errors into the system, due to improper segmentation. **Lattice SLU**

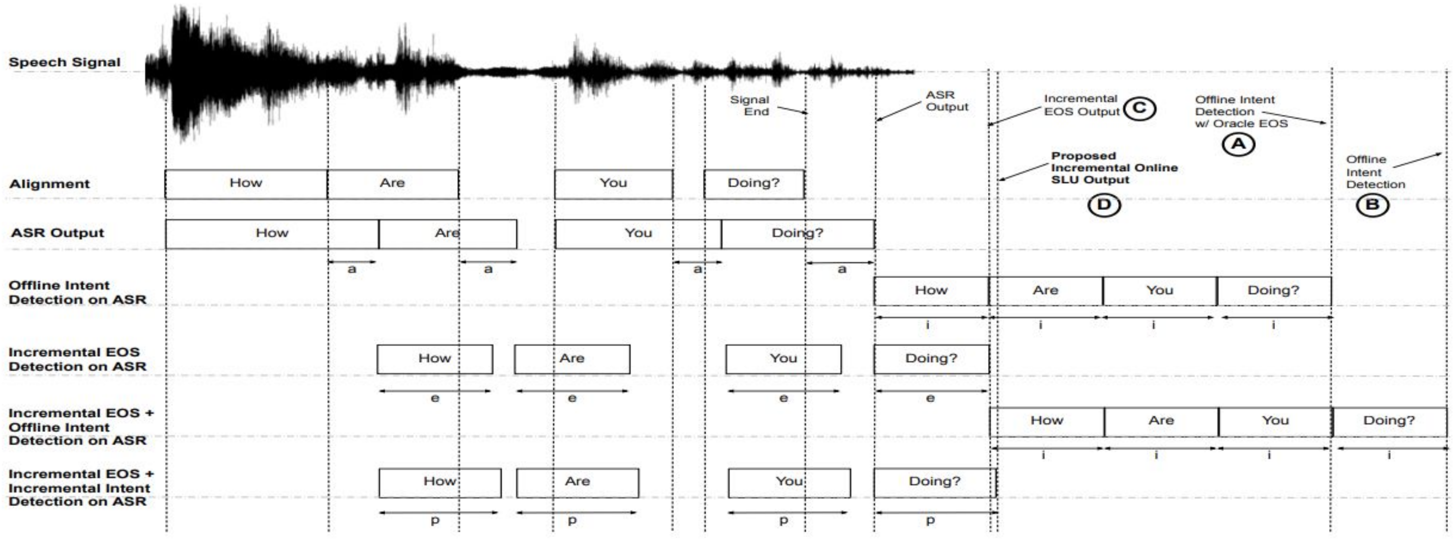
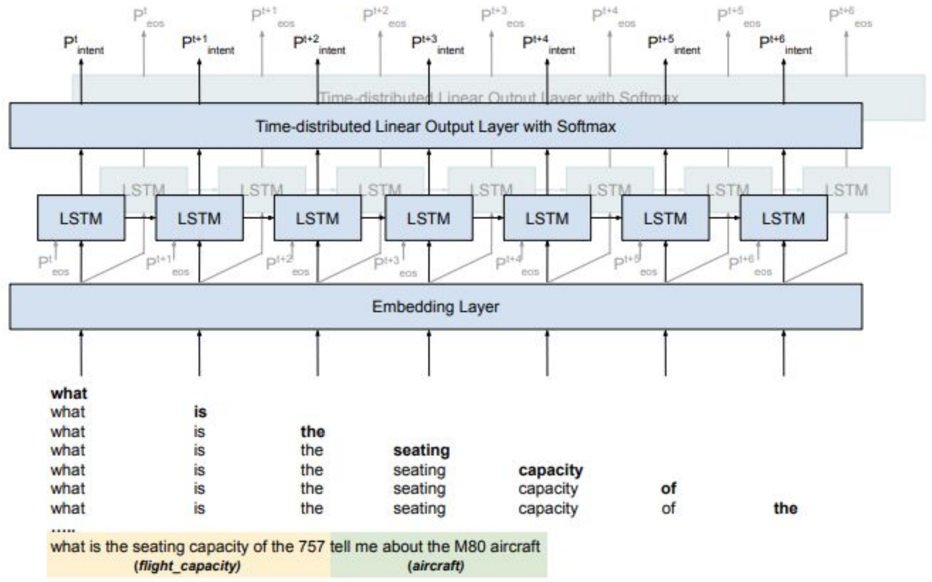

Speech Signal
Signal End
ASR Output
Incremental EOS Output C
Offline Intent Detection w/ Oracle EOS A
Proposed Incremental Online SLU Output D
Offline Intent Detection B
Alignment
How
Are
You
Doing?
ASR Output
Offline Intent Detection on ASR
Incremental EOS Detection on ASR
Incremental EOS + Offline Intent Detection on ASR
Incremental EOS + Incremental Intent Detection on ASR
a
i
e
p

Fig. 1: Time-line of the entire ASR-SLU pipeline and latency implication of offline versus incremental online systems (a, i, e, p are the latencies of ASR, intent classifier, EOS detector, and the latency of the proposed system respectively. a, i, e, p assumed to be constant for simplicity)

Modelling



Time-distributed Linear Output Layer with Softmax
LSTM
Embedding Layer
what
what is
what is the
what is the seating
what is the seating capacity
what is the seating capacity of
what is the seating capacity of the
.....
what is the seating capacity of the 757
(flight_capacity)
tell me about the M80 aircraft
(aircraft)

Fig. 2: Proposed LSTM Architecture (simulation of ASR partials, bold words are increments. 2 utterances, 2 intents)

- Can use only unidirectional LSTM as NLU since no access to future transcript
- To assess its performance for the online task, during evaluation, we derive the output per each time step by sharing the output layer over all the time-steps.
- This task is jointly trained with End of sentence (EOS) detection task which has an independent EOS classifier

Evaluation

- the ATIS (Airline Travel Information Systems) benchmark dataset
- Snips dataset
- Facebook’s multilingual task oriented dialog (FMTOD)
- For each dataset, random number of samples from manual transcripts of the dataset were stitched together without exceeding a maximum utterances limit
- DIfferent between Online and offline SLU increases as the utterance length increases
- They also show early spotting of intent in results

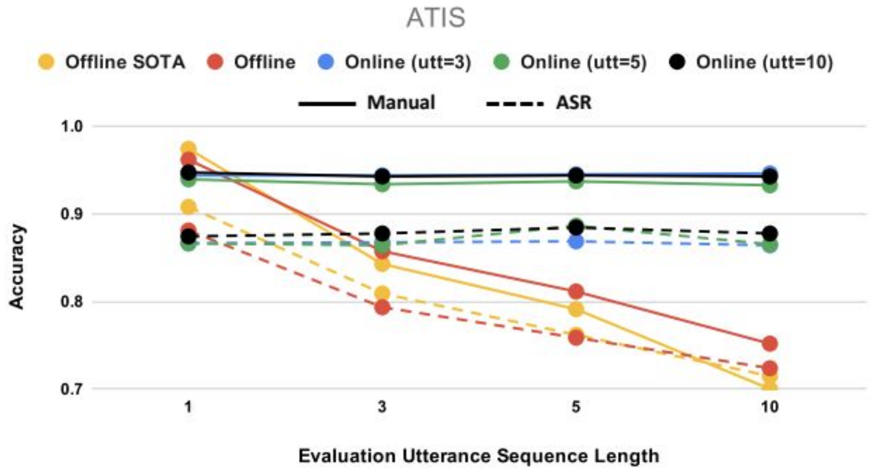


Fig. 3: Accuracy of offline, SOTA vs. proposed online models at oracle EOS on Manual and ASR ATIS transcripts.

Wake up word Detection

- wake word detection system which constantly listens to its environment, expecting a predefined word to be spotted before turning into a more power consumptive state to understand users’ intention
- Also referred in some works as keyword spotting
- A wake word detection system usually runs on devices,
- Hence the neural networks are limited to:
  - small memory footprint
  - small computational cost
  - low latency
  - When people wish to to interact with such devices by voice, they wake up the device by saying a predefined word like “Alexa” for Amazon Echo or “Okay Google” for Google Home. If the word is identified and accepted, the device turns on

Neural Network methods

- Pure neural models abandon HMMs and completely rely on neural networks for acoustic modeling, where the subwords or even the whole word of the wake word phrase (wake phrase, for short) is directly used as modeling units.
- However, they still need a forced alignment obtained from an existing HMM-based ASR system, to obtain training labels, which limits their applications if an ASR system is unavailable.
- There are also several proposals that do not require frame-level alignment for training, like streaming transformers

MINING EFFECTIVE NEGATIVE TRAINING SAMPLES FOR KEYWORD SPOTTING

- E2E architecture
  - They treat a keyword as a single modeling unit and simply detect its presence in the streaming utterance.
  - As each frame arrives, the model decides if a keyword has been discovered. In this case, KWS becomes a keyword/non-keyword binary classification task.
  - The sequence binary classification model is trained to minimize the keyword category cross entropy loss
  - To model the acoustic sequence for keyword spotting, recurrent neural networks (RNNs) and TCNs are two common choices for E2E modeling
  - On top of the RNN or TCN, a linear layer with a sigmoid activation is applied to do the binary classification.

MINING EFFECTIVE NEGATIVE TRAINING SAMPLES FOR KEYWORD SPOTTING

- E2E architecture
  - Max-pooling based loss function.
    - no need to teach the LSTM to fire every frame within the keyword segment.
    - Instead, we want to teach the LSTM to fire at its highest confidence time.
    - The LSTM should fire near the end of keyword segment in general, where it has seen enough context to make a decision
    - The first item states that we calculate the cross-entropy loss for input frames not aligned to any keyword
    - We only back propagate for a single frame whose posterior for target is the largest among all frames within current segment lp, and discard all other frames within current segment.
    - do max-pooling over the ending area of each keyword

$$\mathcal{L}_T^{maxpool} = -\sum_{t \in \hat{\mathbf{L}}} \ln y_t^{k_t^\dagger} - \sum_{p=1}^{P} \ln y_{l_p^\dagger}^{k_p^\dagger}$$

MINING EFFECTIVE NEGATIVE TRAINING SAMPLES FOR KEYWORD SPOTTING

- E2E architecture
  - Class balancing issue
    - It is expensive to collect positive keyword training data, while it is easy to find abundant non-keyword data.
    - Additionally, we do need a large amount of diverse negative training data to prevent false alarms, especially due to phrases similar to the keyword or due to various environment noises.

MINING EFFECTIVE NEGATIVE TRAINING SAMPLES FOR KEYWORD SPOTTING

- E2E architecture
  - Negative sampling
    - we strategically down-sample negative frames to keep data in check between the two classes.
    - For each negative utterance in a mini-batch, we select the most difficult frame with the top positive posterior probability computed by the current model. This frame is put into a collection I.
    - Then, we mask Δ neighboring frames (both left and right neighbors) of the selected hardest frame. These masked frames are not selected, as they are assumed to be acoustically similar to the selected frame.
    - After processing all the negative utterances in a mini-batch, we rank all negative frames in I by their posterior probabilities and select the top rP frames for training the negative class, thereby keeping the data ratio between these two classes to be under r.

MINING EFFECTIVE NEGATIVE TRAINING SAMPLES FOR KEYWORD SPOTTING

- E2E architecture
  - TR (Trigger Region) frame
    - TR usually comes from automatic forced alignment by existing acoustic models, which may not be accurate.
      - a Kaldi HMM-TDNN acoustic model trained on general Mandarin speech data.
    - In later epochs, we relax the TR constraint to select the single highest posterior frame from anywhere in the positive utterance since the model now is better trained.

MINING EFFECTIVE NEGATIVE TRAINING SAMPLES FOR KEYWORD SPOTTING

- E2E architecture
  - Results

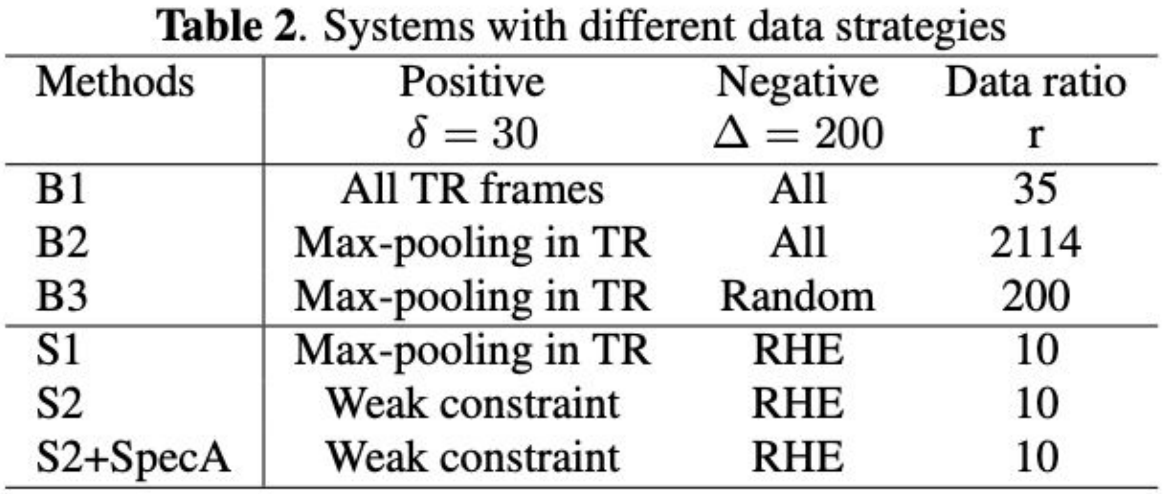


Table 2. Systems with different data strategies

Methods	Positive $\delta = 30$	Negative $\Delta = 200$	Data ratio r
B1	All TR frames	All	35
B2	Max-pooling in TR	All	2114
B3	Max-pooling in TR	Random	200
S1	Max-pooling in TR	RHE	10
S2	Weak constraint	RHE	10
S2+SpecA	Weak constraint	RHE	10

MINING EFFECTIVE NEGATIVE TRAINING SAMPLES FOR KEYWORD SPOTTING

- E2E architecture
  - Results
  - The **FRR** is expressed as a percentage of situations in which are user gets a false negative result.
  - False Alarm rate is false positive rate

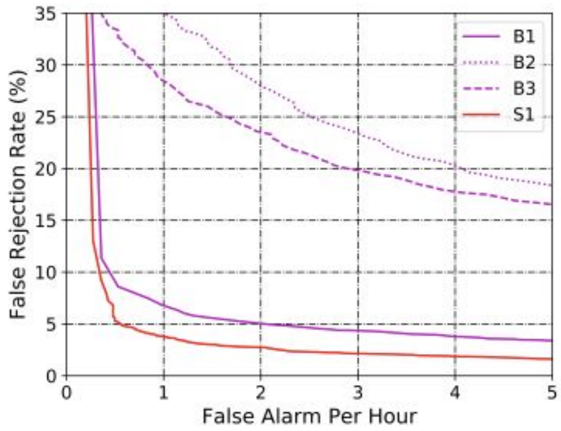


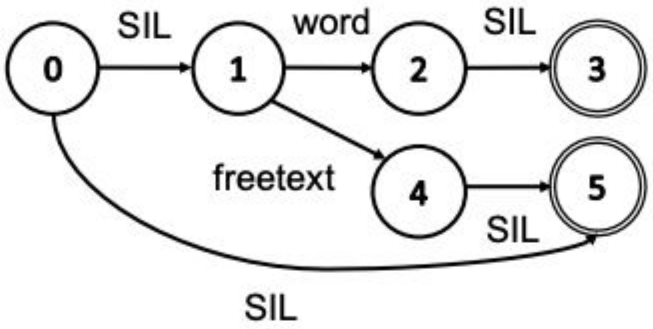
Fig. 1. DET curves on "Hi Xiaowen" with GRU.

THE ALIGNMENT-FREE LF-MMI SYSTEM

- Ln and L are the subword truth sequence and a competing hypothesis sequence respectively, and On is the input audio

$$\mathcal{F}_{\text{LF-MMI}} = \sum_{n=1}^{N} \log P(L_n|\mathbf{O}_n) = \sum_{n=1}^{N} \log \frac{P(\mathbf{O}_n|L_n)P(L_n)}{\sum_L P(\mathbf{O}_n|L)P(L)}$$

- Competing hypothesis generated from FST shown on right
- One path containing the word HMM corresponds to positive recordings , and the other two correspond to negative recordings (other speech/non-speech and silence).

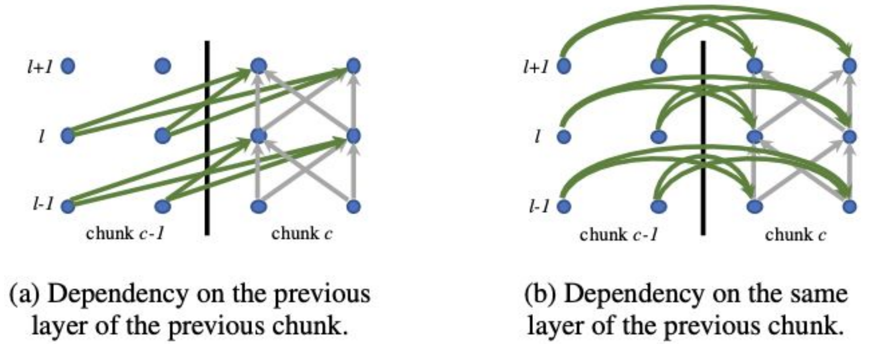


Wake Word Detection with Alignment-Free Lattice-Free MMI

- LF-MMI Loss
- Acoustic Modelling
  - Factorised TDNN with skip connections
- Viterbi Decoding
  - Start and end state same to form loop
  - a wake word is found, we just stop decoding and trigger the system; otherwise continue the decoding process.

Streaming transformers

- The whole input sequence is segmented into several equal-length chunks
- the hidden state from the previous chunk are cached to extend keys and values from the current chunk, providing extra history to be attended to.
- SG is stop gradient operation in equation on right



l+1
l
l-1
chunk c-1
chunk c
(a) Dependency on the previous layer of the previous chunk.
(b) Dependency on the same layer of the previous chunk.

Fig. 1: Two different type of nodes dependency when computing self-attention in streaming Transformers. The figures use 3-layer networks with 2 chunks (delimited by the thick vertical line in each sub-figure) of size 2 as examples. The grey arcs illustrate the nodes dependency within the current chunk, while the green arcs show the dependency from the current chunk to the previous one.

$$\tilde{\mathbf{K}}_c = [\mathrm{SG}(\mathbf{K}_{c-1}); \mathbf{K}_c], \quad \tilde{\mathbf{V}}_c = [\mathrm{SG}(\mathbf{V}_{c-1}); \mathbf{V}_c]$$

Modification to Streaming transformers

- also allow a chunk to “look-ahead” to the next chunk to get future context when making predictions from the current chunk.
- For the right context, the gradient in back-propagation does not just stop at Kc+1 and Vc+1, but rather go all the way down to the input within the chunk c + 1.
- Can do potentially same for left cache - no state caching
- Use embedding matrix for positional embedding
- Can do something clever in computation
- In the 0- th row, we keep those corresponding to the relative positions in the range [−lk + lq, lq − 1]; in the i-th row, the range is left shifted by 1 from the one in the (i − 1)-th row; finally in the (lq − 1)-th row, the range would be [−lk + 1, 0].

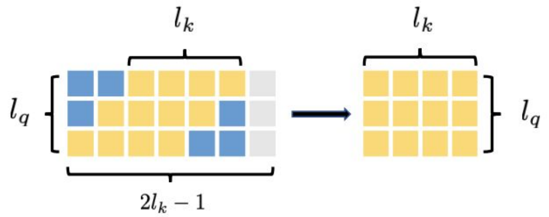


Fig. 2: The process of selecting relevant cells from the matrix $\mathbf{M} \in \mathbb{R}^{l_q \times (2l_k - 1)}$ (left) and rearranging them into $\mathbf{M}' \in \mathbb{R}^{l_q \times l_k}$ (right). The relevant cells are in yellow, and others are unselected. Note that the position of yellow block in one row of $\mathbf{M}$ is left shifted by 1 cell from the yellow block in the row above.

Results

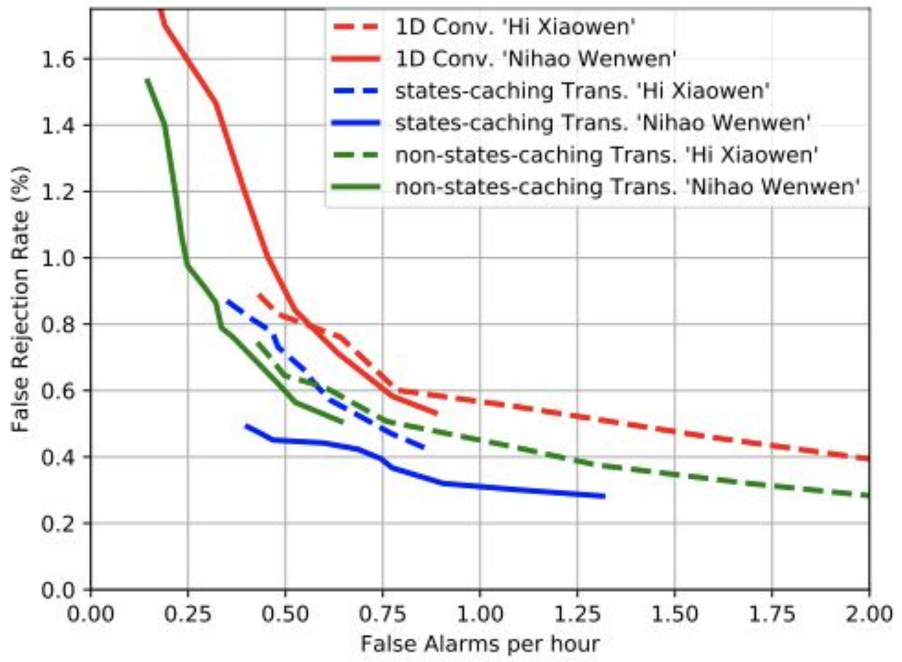


Fig. 3: DET curves for the baseline 1D convolution network and our two proposed streaming Transformers.