# A NEURAL NETWORK BASED RANKING FRAMEWORK TO IMPROVE ASR WITH NLU RELATED KNOWLEDGE DEPLOYED

*Zhengyu Zhou[1], Xuchen Song[2], Rami Botros[3], Lin Zhao[1]*

[1] Bosch Research and Technology Center North America, Sunnyvale, CA, USA;
[2] ByteDance AI Lab, Palo Alto, CA, USA; [3] Google Inc., Mountain View, CA, USA

## INTRODUCTION

Previous efforts to improve Automatic Speech Recognition (ASR):

- ➢ Various directions explored: end-to-end training, language modeling, etc.
- ➢ One popular choice: post-processing hypotheses generated by ASR
  - Mainly due to the convenience of applying additional knowledge
  - Methods include confusion networks, re-ranking ASR hypotheses with various language/discriminative models, pairwise classification using SVM or neural network encoder based classifier, etc.
- ➢ From the aspect of knowledge usage, limited linguistic knowledge is used
  - Natural Language Understanding (NLU) information, such as slots and intents, are typically not used in efforts to improve ASR.

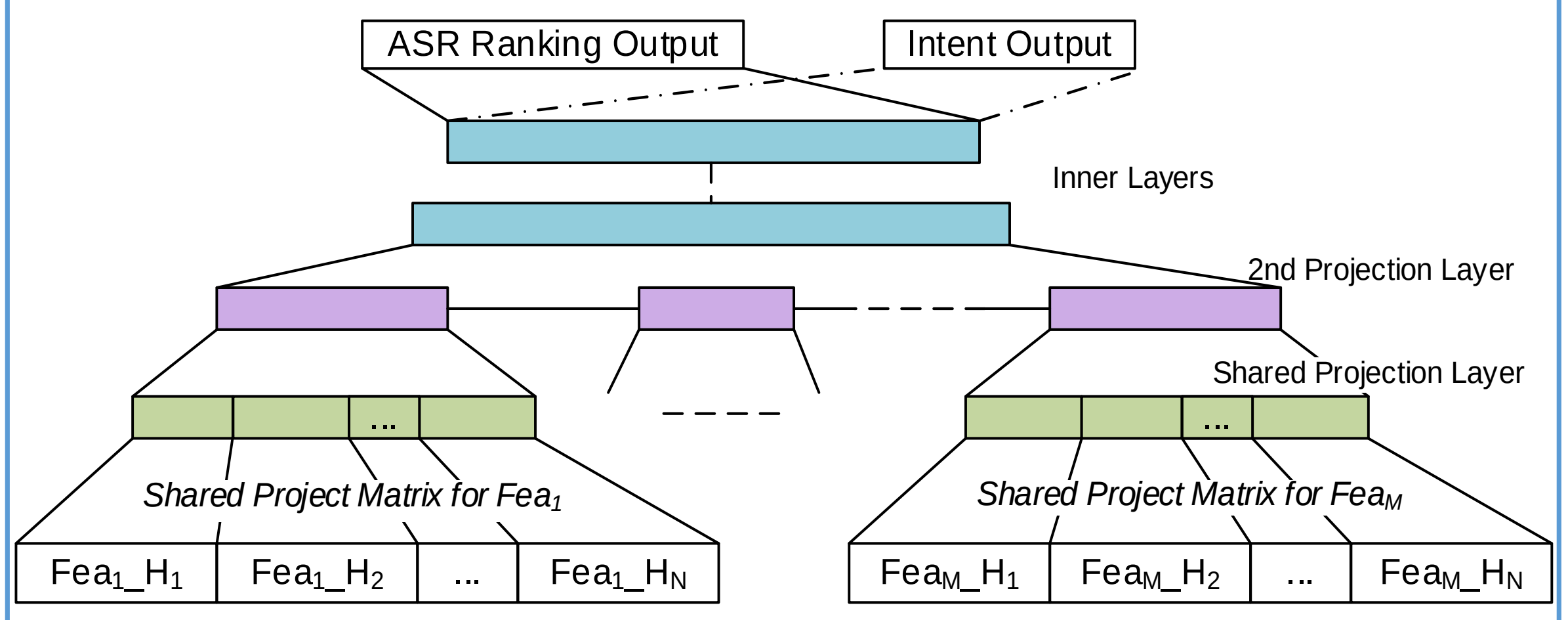
Contribution of this work:

- ➢ Propose a new framework that uses neural network to simultaneously rank multiple hypotheses generated by ASR for a speech utterance.
  - The framework uses all competing hypotheses as input, and predicts the ranking of them (based on scores of corresponding output nodes).
- ➢ NLU knowledge is deployed in the framework to facilitate the ranking
  - In addition to ASR features, slot/intent relevant features are also used, e.g., triggers capturing long-distance constraints between words/slots; BLSTM feature representing intent-sensitive sentence embedding.
  - The framework can be jointly trained with intent detection
- ➢ Novel soft target values that capture ranking distribution are also proposed for the training of the framework, and found effective.
- ➢ The framework simultaneously generates new ASR and NLU results.

## Proposed Framework Structure

We propose a new framework, which a deep feedforward neural network, to simultaneously rank multiple ASR hypotheses for a speech utterance.

- Input: features from $N$ ($N$=10 in this study) competing hypotheses.
- Output: ranking result, optionally together with intent detection result



*Input: M* types of features ($Fea_1, Fea_2, \ldots, Fea_M$) extracted from *N* hypotheses ($H_1, H_2, \ldots, H_N$)

Note: For a feature type that extracts hundreds or more features per hypothesis, two projection layers are used to handle those features. In case that a feature type only generates one or a few features per hypothesis, such as the confidence score feature, related features are directly feed into inner layers, concatenated with 2nd project layer.

In the output layer, the major part is "ASR Ranking Output"

- contains $N$ output nodes, corresponding to the $N$ input hypotheses.
- Softmax activation is used to generate the output values, and the hypotheses are then ranked based on the values accordingly.
- To effectively train the framework, instead of one-hot target, we propose the following soft target values :

$$\text{target}_i = \frac{e^{-d_i}}{\sum_{i=0}^{n} e^{-d_i}}$$

where $d_i$ is the Levenshtein distance of the $i$th hypothesis from the reference

- The target distribution reserves the ranking information, generating a higher score for an output node if the corresponding input hypothesis contains less ASR errors.
- By minimizing the Kullback-Leibler Divergence loss, the output distribution approximates the target distribution.

The output layer optionally includes an "intent output" part for joint training

- It could be beneficial to jointly train the ranking task and intent detection, since intent information may help distinguish among the hypotheses.
- "Intent" related output nodes are corresponding to possible intents, assigned with one-hot target values, and trained with cross-entropy loss.
- For joint training, we back-propagate the costs from both the ASR ranking part and the intent related part to the lower layers.

## Features

Four types of features are extracted per hypothesis in this study:

➢ Trigger Feature

- Trigger features are used to model long/flexible-distance constraints
- Trigger: a pair of linguistic units (i.e., word/slot) that are significantly correlated in a same sentence. Example: "play" → <song name>
- Selection of triggers: Given an in-domain text corpus, use slots to replace corresponding text (e.g., using <song name> to replace "Poker Face"), calculate mutual information (MI) scores of all possible trigger pairs *A*→*B* based on the corpus as follows [Rosenfeld, 1994]. The top *n* trigger pairs with highest MI scores are then selected as trigger features.

$$MI(A:B) = P(A,B)\log\frac{P(B \mid A)}{P(B)} + P(A,\overline{B})\log\frac{P(\overline{B} \mid A)}{P(\overline{B})} + P(\overline{A},B)\log\frac{P(B \mid \overline{A})}{P(B)} + P(\overline{A},\overline{B})\log\frac{P(\overline{B} \mid \overline{A})}{P(\overline{B})}$$

- Extraction of trigger features: Use a standalone NLU module (see next figure) to detect the slots in each hypothesis. The value of a trigger feature is 1 if the trigger pair appears in the hypothesis, and 0 otherwise.
- Trigger feature captures the dependencies between the two units no matter how far they may be apart in a sentence.

➢ BOW Feature

- Bag-of-words (BOW) features, previously used in feedforward neural network language modeling [Kazuki, 2015], are also adopted. Given a dictionary, a vector of BOW features is calculated for each hypothesis as:

$$bow_{decay} = \sum_{i=0}^{K} \gamma^{i} \vec{w_i}$$

where $K$ is the number of words in the hypothesis, $w_i$ is the one-hot representation of the $i$-th word in the hypothesis, $r$ is a decaying factor.

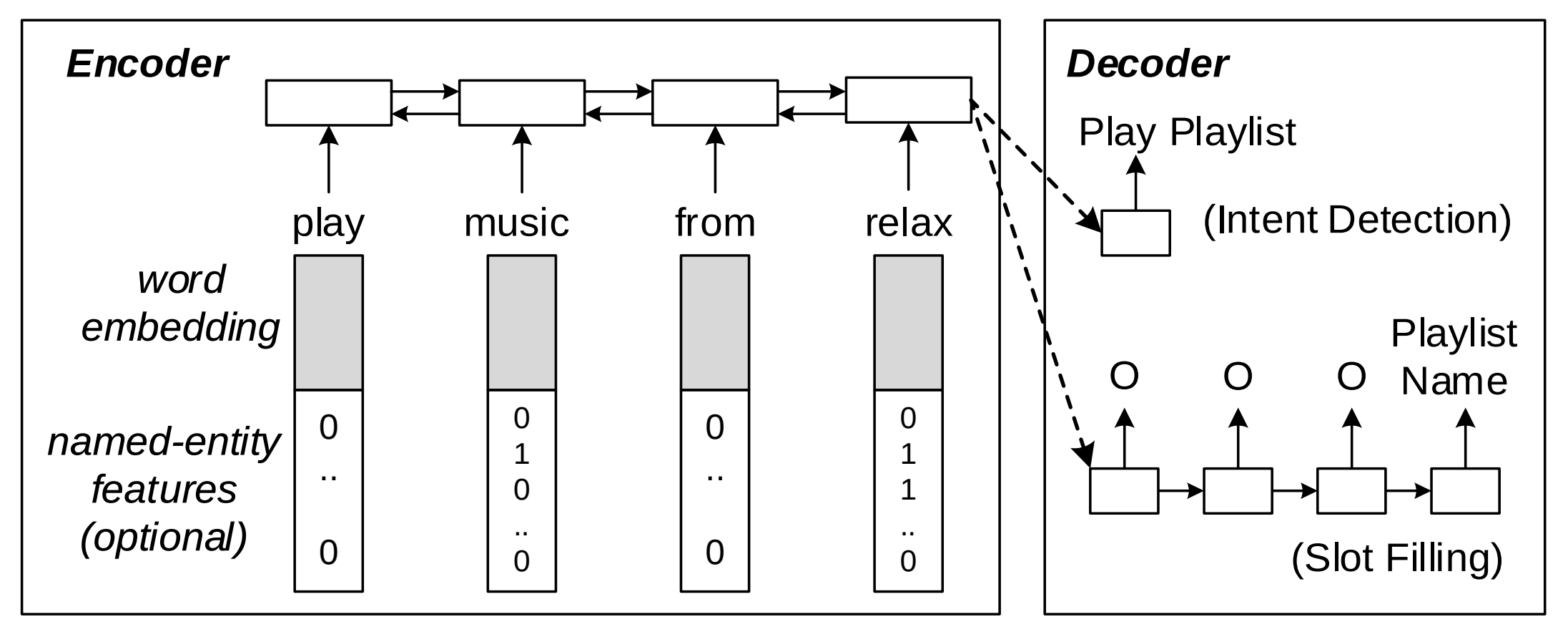
➢ BLSTM Feature

- In the extraction of trigger features, the standalone NLU module is used to process each hypothesis. It utilizes bidirectional LSTM RNN to encode the hypothesis, where the last states of both the forward and backward RNN cover information of entire hypothesis. We concatenate the two last states into a *sentence embedding vector*, referred to as the *BLSTM features*.
- Since the NLU module is a joint model of intent detection and slot filling, the BLSTM features are intent-sensitive.

➢ Confidence Feature

- Sentence-level confidence score assigned by ASR engine to each hypothesis is also used as feature, and directly fed into the inner layers.
- Note that various ASR engines may produce confidence scores using different distributions. When the input hypotheses are generated by different ASR engines, we apply a linear regression method [Zhou, 2018] proposed previously to align the confidence scores into a same space, and then use the *aligned score* as the *confidence feature*.

## A Standalone NLU Module



A standalone NLU module developed for feature extraction (and evaluation)

- Adopt a LSTM based encoder-decoder structure [Liu & Lane, 2016]
- Jointly models slot filling and intent detection

We propose a method to improve performance: Feeding in not only word embedding but also named-entity features, when name lists are available.

- Aim: Use name type information to facilitate learning, especially when having limited training data and many names are unseen or rarely occur.
- Each value of the named-entity vector is corresponding to one name list, set as 1 if the input word is part of a name in that list and 0 otherwise.
- Example: The word "relax" is both a song name and playlist name, shown in the named-entity vector. The NLU module may identify "relax" as a playlist name even if the name "relax" is unseen in the training data.

## Experiments

➢ ***Dataset 1***: internal in-car infotainment corpus (driver assistant topics)

- Training/tuning/testing sets: 9166, 1975 and 2080 utterances respectively.
- Speech recorded in car with relatively low noise conditions from multiple speakers with balanced gender.
- Decoded by three complementary ASR engines (general ASR engine, plus grammar based and statistical LM based domain-specific engines) Top-best hypothesis from each engine fed into the framework to rank.
- Most names involved in this dataset are from 16 name lists, some of which are large (e.g., the song list has 5232 entries). |
- 40 slot labels (including phone number, frequency, etc. which have no predefined list) were created following IOB schema [Ramshaw, 1999]
- 89 distinct intents (e.g., "tune radio frequency") are used.

➢ ***Dataset 2***: public ATIS corpus (airline travel information)

- Training/tuning/testing sets: 4497, 491 and 893 sentences respectively.
- Speech utterances synthesized with noises and reverberation added
- Decode by one ASR engine: state-of-the-art Google cloud ASR, which generates 10 hypotheses at maximum per utterance.
- There are 127 slot labels and 18 distinct intents in total.

➢ For both datasets, we develop and evaluate the systems in similar ways.

- NLU module: Trained with references, using GloVe word embedding. Since ATIS data provides no name list, the named-entity vector is applied to infotainment data only, based on given name lists.
- Trigger features: 850 trigger features are utilized for both corpora. Triggers selected based on training references for ATIS data, and on an additional text set of 24,836 in-domain sentences for infotainment data.
- BOW features: Dictionary is defined as the 90% most frequent words in training references, along with an entry for out-of-vocabulary words.
- Confidence alignment is only needed and applied for infotainment data.

➢ Ranking Framework Implementation

- Accept 10 hypotheses to rank. Void the space of extra hypotheses as 0.
- Each shared projection matrix projects to a space of 50 nodes, leading to a layer of size 50*10*3 (500 nodes per trigger/BOW/BLSTM feature)
- 2nd projection layer size: 200*3 for infotainment data, 100*3 for ATIS.
- 10 confidence features + 2nd projection layer → Input layers.
- Inner layers: 4 layers (500, 200, 100 and 100 nodes, respectively) for infotainment data, and 3 layers (200, 100 and 50 nodes, respectively) for ATIS, with ReLU activation and batch normalization applied per layer.
- Adam optimization used to train the model in batches.
- Early stopping is conducted when the loss on tuning data fails to improve for the last 30 iterations. The model performing the best on tuning data is used in evaluation.

## Results on In-Car Infotainment Data

| | WER% | Intent Error% | Slot F1 |
|---|---|---|---|
| Oracle hypo. +NLU | 3.87 | 7.84 | 84.51 |
| Top-scored hypo. +NLU | 7.56 | 11.37 | 79.73 |
| ASR-alone Framework | 7.05 | 10.49 | 79.82 |
| Joint Framework | **6.69** | **10.00** | **80.50** |

- *Oracle hypo.*: the hypothesis with the lowest word error rate (WER)
- *Top-scored hypo.*: the hypothesis with highest (aligned) confidence
- *ASR-alone Framework*: framework with only ASR ranking output
- *Joint Framework*: jointly trained with ASR ranking and intent detection

- ➢ For either framework, when it predicts the top-ranked hypothesis, it also retrieves that hypothesis' NLU results (obtained in feature extraction). In this way, ASR and NLU results are obtained simultaneously.
- ➢ Adopting the proposed soft target values for ranking is found important. For example, in the joint framework, when replacing the soft target values with one-hot values, the WER obtained rises to 7.21%.
- ➢ All the four types of features are found beneficial in this case. For example, removing the trigger features (NLU relevant) from the joint framework increases the WER to 7.28%.
- ➢ For the standalone NLU module, feeding in named-entity features is beneficial, e.g., reducing the intent error from 9.12% to 5.17% and raising slot filling F1 from 64.55 to 90.68 on testing references.

## Results on ATIS data

| | WER% | Intent Error% | Slot F1 |
|---|---|---|---|
| Oracle hypo. +NLU | 1.24 | 2.58 | 94.05 |
| Top-scored hypo. +NLU | 6.53 | 3.92 | 90.89 |
| ASR-alone Framework | **5.10** | **3.25** | **92.50** |
| Joint Framework | 5.51 | 3.36 | 91.93 |

- ➢ On ATIS, all types of features are beneficial except the BLSTM feature (possibly because competing hypotheses are generated by one engine and are highly similar in sentence embedding). The BLSTM feature is thus removed from the ranking frameworks. Results are shown above.
- ➢ ASR-alone framework achieves the best performance, obtaining 21.9% relative reduction in WER over state-of-the-art Google ASR.
- ➢ Joint training with intent brings no improvement in this case, also due to the similarity among hypotheses, which typically bear the same intent.
- ➢ Adopting the soft target values is also critical. Using one-hot target values instead in the ASR-alone framework leads to a WER of 6.68%.

## Summary

- We proposed a new framework to rank competing hypotheses for a speech utterance, shown effective no matter the hypotheses are generated by multiple ASR engines or one engine.
- The framework ranks hypotheses based on not only ASR but also NLU related features, generating new ASR and NLU results at the same time.
- Novel soft target value is proposed to effectively train the framework.
- Joint training the framework with intent detection is found beneficial when different types of engines are used in decoding.
- For the standalone NLU module, incorporating name type information improves NLU performance when predefined name lists are available.
- In future, further improvements can be made in various directions, such as introducing new features and improving the network design.