WIDM @ NTCIR-14 STC-3 Task:
Dialogue Quality and Nugget Detection for Short Text Conversation (STC-3) based on Hierarchical Multi-Stack Model with Memory Enhance Structure

NATIONAL CENTRAL UNIVERSITY, TAOYUAN, TAIWAN

AUTHORS: HSIANG-EN CHERNG AND CHIA-HUI CHANG

PRESENTER: HSIANG-EN CHERNG (SEAN)
Outline

1. Introduction
2. Dialogue Quality (DQ) Subtask
3. Nugget Detection (ND) Subtask
4. Conclusion
Introduction

Task Overview – DQ Subtask
Task Overview – ND Subtask
Contribution
Task Overview – DQ Subtask

Goal of DQ

- DQ aims to evaluate the quality of a dialogue by three measures (scale: -2, -1, 0, 1, 2)
  1) A-score: Task Accomplishment
  2) E-score: Dialogue Effectiveness
  3) S-score: Customer Satisfaction of the dialogue

Why DQ

- To build good task-oriented dialogue systems, we need good ways to evaluate them
- You cannot improve dialogue systems if you cannot measure, DQ provides 3 measures
Task Overview – ND Subtask

Goal of ND

- ND subtask aims to classify the nugget of utterances in a dialogue
- ND is similar to dialogue act (DA) labeling problem

Why ND

- Nuggets may serve as useful features for automatically estimating dialogue quality
- ND may help us diagnose a dialogue closely (why it failed, where it failed)
- Experiences from ND may help us design effectively and efficiently helpdesk systems

Nugget: purpose or motivation
Contribution

1. We proposed and compared several DNN models based on
   ◦ Hierarchical multi-stack CNN for sentence and dialog representation
   ◦ BERT for sentence representation

2. We compared the models with or without memory enhance

3. We compared simple BERT model with BERT + complex structures model

4. In both DQ and ND, our models result in the best performance comparing with organizer baseline models

BERT: An pre-train model based on multiple bi-directional transformer blocks (Devlin, J., Chang, M, W., Lee, K., Toutanova, K. 2018)
Dialogue Quality (DQ) Subtask

Model

Experiments
Memory enhanced multi-stack gated CNN (MeHGCNN)

Embedding layer
- 100 dimensions Word2Vec

Utterance layer
- 2-stack gated CNN learning sentence representation

Context layer
- 1-stack gated CNN learning context information

Memory layer (Memory Network)
- Further capture long-range context features

Output layer
- Output DQ distribution by softmax
3 techniques we used in our models

1. Multi-stack structure
2. Gating mechanism
3. Memory enhance (memory network)
Multi-stack

Multi-stack structure

- Hierarchically capture rich n-gram information
- Window size k and # stacks m can capture m(k-1)+1 words features
Gating mechanism & Memory Enhance Structure

Gating mechanism
- Widely used in LSTM and GRU to control the gates of memory states
- The idea of gated CNN is to learn whether to keep or drop a feature generated by CNN

Memory enhance structure
- LSTM are not good at capturing very long-range context features
- Memory network is applied to our models to get detail context features by self-attention
Utterance Layer: 2-stack Gated CNN

Utterance layer (UL)

- $l = 1$
- $X_i^l = [w_{i,1}, w_{i,2}, ..., w_{i,n}]$
- $ulA_i^l = \text{ConvA}(X_i^l)$
- $ulB_i^l = \text{ConvB}(X_i^l)$
- $ulC_i^l = ulA_i^l \odot \sigma(ulB_i^l)$
- $X_i^{l-1} = ulC_i^l$
- $ul_i = [\text{maxpool}(ulC_i^l), \text{speaker}_i, \text{nugget}_i]$

Apply max-pooling to the output of the last stack
Context Layer: 1-stack Gated CNN

- Context layer (CL)
  - Conduct the same operations as UL but no additional features
  - $c_l A_i = \text{ConvA}([ul_{i-1}, ul_i, ul_{i+1}])$
  - $c_l B_i = \text{ConvB}([ul_{i-1}, ul_i, ul_{i+1}])$
  - $c_l C_i = c_l A_i \odot \sigma(c_l B_i)$
  - $c_l = \text{maxpool}(c_l C_i)$

The output of context layer for utterance $i$ is $c_l i$
Memory Layer

Memory layer (ML)

1) Both input memory ($I_i$) and output memory ($O_i$) are generated by BI-GRU from $cl_i$

- **Input Memory**
  - $\tilde{I}_i = \overline{GRU}(cl_i, h_{i-1})$
  - $\hat{I}_i = \overline{GRU}(cl_i, h_{i+1})$
  - $I_i = \tanh(\tilde{I}_i + \hat{I}_i)$

- **Output Memory**
  - $\tilde{O}_i = \overline{GRU}(cl_i, h_{i-1})$
  - $\hat{O}_i = \overline{GRU}(cl_i, h_{i+1})$
  - $O_i = \tanh(\tilde{O}_i + \hat{O}_i)$
Memory Layer (cont.)

- Memory layer (ML)
  2) **Attention weight** is the inner product between $cl_i$ and $I_i$ followed by softmax
     \[ w_i = \frac{\exp(cl_i \cdot I_i)}{\sum_{i' = 1}^{k} \exp(cl_{i'} \cdot I_{i'})} \]

  3) The output of memory layer for $cl_i$ is the addition between **weighted sum of $O_i$ and $cl_i$**
     \[ ml_i = \sum_{i' = 1}^{k} w_{i'} \cdot O_{i'} + cl_i \]
Output Layer

- **Output layer**
  - Flatten all utterances vectors
    - \( ml = [ml_1, ml_2, ..., ml_k] \)
  - Apply a fully-connected layer with softmax to output the score distribution as
    - \( fc = mlW_{fc} + b_{fc} \)
    - \( P(score|dialogue) = \frac{\exp(fc_i)}{\sum_{i'=1}^{5} \exp(fc_{i'})} \)
  - Dimension of \( P(score|u_i) \) is 1x5 since the scale of scores are -2, -1, 0, 1, 2
Dialogue Quality (DQ) Subtask

Model

Experiments
Data

Customer helpdesk dialogues

- Annotators: 19 students from Waseda university
- Validation data is randomly selected 20% from training data

Preprocessing

- Remove all full-shape characters
- Remove all half-shape characters except A-Za-z!"#$%&()*+,-./:;<=>?@[\]^_`{|}~ ‘
- Tokenize by NLTK toolkit (Edward Loper and Steven Bird. 2002)

<table>
<thead>
<tr>
<th>Data</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td># Dialogues</td>
<td>1,672</td>
<td>390</td>
</tr>
<tr>
<td># Utterances</td>
<td>8,672</td>
<td>1,755</td>
</tr>
</tbody>
</table>
Word Embedding

Embedding parameter
- Dimension: 100
- Tool: genism
- Method: skip-gram
- Window size: 5

STC-3 DQ&ND data
- Customer helpdesk dialogues
- Including train data and test data

<table>
<thead>
<tr>
<th>Data source</th>
<th># words</th>
</tr>
</thead>
<tbody>
<tr>
<td>text8(wiki)</td>
<td>17,005,208</td>
</tr>
<tr>
<td>STC-3 DQ&amp;ND</td>
<td>339,410</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>17,344,618</strong></td>
</tr>
</tbody>
</table>
## Hyper parameters of DQ

<table>
<thead>
<tr>
<th>Hyper parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>40</td>
</tr>
<tr>
<td>Epochs</td>
<td>50</td>
</tr>
<tr>
<td>Early stopping</td>
<td>3</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam optimizer</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0005</td>
</tr>
</tbody>
</table>

### Multi-stack CNN of UL
- # convolutional layers: 2
- # Filter: [512, 1024]
- Kernel size: 2 & 2

### Multi-stack CNN of CL
- # convolutional layers: 1
- # Filter: [1024]
Result of DQ Subtask

- **MeHGCNN**: Our proposed model
- **MeGCBERT**: Replace embedding and utterance layer of MeHGCNN with BERT
- **BL-BERT**: Simple BERT model with only BERT and output layer

<table>
<thead>
<tr>
<th>Model</th>
<th>(A-score)</th>
<th>(E-score)</th>
<th>(S-score)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMD</td>
<td>RSNOD</td>
<td>NMD</td>
</tr>
<tr>
<td>Organizer baselines</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BL-uniform</td>
<td>0.1677</td>
<td>0.2478</td>
<td>0.1580</td>
</tr>
<tr>
<td>BL-popularity</td>
<td>0.1855</td>
<td>0.2532</td>
<td>0.1950</td>
</tr>
<tr>
<td>BL-lstm</td>
<td>0.0896</td>
<td>0.1320</td>
<td>0.0824</td>
</tr>
<tr>
<td>BL-BERT</td>
<td>0.0934</td>
<td>0.1379</td>
<td>0.0881</td>
</tr>
<tr>
<td>MeHGCNN</td>
<td>0.0862</td>
<td>0.1307</td>
<td>0.0814</td>
</tr>
<tr>
<td>MeGCBERT</td>
<td><strong>0.0823</strong></td>
<td><strong>0.1255</strong></td>
<td><strong>0.0791</strong></td>
</tr>
</tbody>
</table>

Ours
Ablation of MeGCBERT for DQ

Gating mechanism & Memory enhance
- Well improve A-score & S-score
- A little improvement in E-score

Adding Nugget features
- Well improve A-score
- A little improvement in E-score

<table>
<thead>
<tr>
<th>Model</th>
<th>(A-score) NMD</th>
<th>(A-score) RSNOD</th>
<th>(E-score) NMD</th>
<th>(E-score) RSNOD</th>
<th>(S-score) NMD</th>
<th>(S-score) RSNOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeGCBERT</td>
<td>0.0823</td>
<td>0.1255</td>
<td>0.0791</td>
<td>0.1202</td>
<td>0.0758</td>
<td>0.1245</td>
</tr>
<tr>
<td>W/o gating mechanism</td>
<td>0.0885</td>
<td>0.1322</td>
<td>0.0813</td>
<td>0.1214</td>
<td>0.0815</td>
<td>0.1289</td>
</tr>
<tr>
<td>W/o memory enhance</td>
<td>0.0913</td>
<td>0.1364</td>
<td>0.0808</td>
<td>0.1235</td>
<td>0.0799</td>
<td>0.1273</td>
</tr>
<tr>
<td>W/o nugget features</td>
<td>0.0963</td>
<td>0.1388</td>
<td>0.0802</td>
<td>0.1204</td>
<td>0.0774</td>
<td>0.1247</td>
</tr>
</tbody>
</table>
Nugget Detection (ND) Subtask

Model

Experiments
Hierarchical multi-stack CNN with LSTM (HCNN-LSTM)

Embedding layer
- 100 dimensions Word2Vec

Utterance layer
- Apply 3-stack CNN to learn sentence representation

Context layer
- Apply 2-stack BI-LSTM to learn context information between utterances

Output layer
- Output the nugget distribution by softmax
Utterance Layer: 3-stack CNN

➢ Utterance layer (UL)
  - $l = 1$
  - $X_i^l = [w_{(i,1)}, w_{(i,2)}, \ldots, w_{(i,n)}]$
  - $ulA_i^l = ConvA(X_i^l)$
  - $ulB_i^l = ConvB(X_i^l)$
  - $ulC_i^l = [ulA_i^l, ulB_i^l]$
  - $X_i^{l-1} = ulC_i^l$
  - $ul_i = [\text{maxpool}(ulC_i^l), \text{speaker}_i]$

Filter size: 2\&3 for convA\&convB

\textit{if } l \leq 3

2019/6/27
Context Layer: 2-stack BI-LSTM & Output Layer

- **Context Layer (CL)**
  - $c_l^i = LSTM(ul_i, h_{i-1})$
  - $c_l^i = LSTM(ul_i, h_{i+1})$
  - $c_l^i = \tanh(c_l^i + c_l^i)$
  - $ul_i = c_l^i$
  - $cl_i = c_l^i$

- **Output layer**
  - $P(\text{nugget}|u_i) = \frac{\exp(Wc_l^i)}{\sum_{l'=1}^{k} \exp(c_l'^i)}$
Nugget Detection (ND) Subtask

Model

Experiments
## Hyper parameters of ND

<table>
<thead>
<tr>
<th>Hyper parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>30</td>
</tr>
<tr>
<td>Epochs</td>
<td>50</td>
</tr>
<tr>
<td>Early stopping</td>
<td>3</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam optimizer</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0005</td>
</tr>
<tr>
<td>Multi-stack CNN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• # convolutional layers: 3</td>
</tr>
<tr>
<td></td>
<td>• # Filter: [256, 512, 1024]</td>
</tr>
<tr>
<td></td>
<td>• Kernel size: 2 &amp; 3</td>
</tr>
<tr>
<td>Multi-stack BI-LSTM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• # BI-LSTM layers: 2</td>
</tr>
<tr>
<td></td>
<td>• # hidden units: [1024, 1024]</td>
</tr>
<tr>
<td></td>
<td>• Activation function of concatenation: tanh</td>
</tr>
</tbody>
</table>
Result of ND Subtask

- **HCNN-LSTM**: Our proposed model
- **BERT-LSTM**: Replace the embedding layer and utterance layer of HCNN-LSTM with BERT
- **BL-BERT**: Simple BERT + Output layer model

- BERT-LSTM outperforms all other models
- HCNN-LSTM outperforms NTCIR baselines in JSD
- Context layer is important for BERT
  - JSD drop 0.012 and RNSS drop 0.024 without context layer

<table>
<thead>
<tr>
<th>Model</th>
<th>JSD</th>
<th>RNSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL-uniform</td>
<td>0.2304</td>
<td>0.3708</td>
</tr>
<tr>
<td>BL-popularity</td>
<td>0.1665</td>
<td>0.2653</td>
</tr>
<tr>
<td>BL-lstm</td>
<td>0.0248</td>
<td>0.0952</td>
</tr>
<tr>
<td>BL-BERT</td>
<td>0.0341</td>
<td>0.1171</td>
</tr>
<tr>
<td>HCNN-LSTM</td>
<td>0.0246</td>
<td>0.0962</td>
</tr>
<tr>
<td>BERT-LSTM</td>
<td>0.0228</td>
<td>0.0933</td>
</tr>
</tbody>
</table>

Organizer baselines

Ours
Ablation & Complex Structure Experiments

Left table shows that both UL and CL are important for ND subtask

Right table shows that both gating mechanism and memory enhance structure doesn’t improve the performance

- Since the less training data, complex structure might cause overfitting

<table>
<thead>
<tr>
<th>Model</th>
<th>JSD</th>
<th>RNSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-LSTM</td>
<td>0.0228</td>
<td>0.0933</td>
</tr>
<tr>
<td>W/o CL multi-stack</td>
<td>0.0246</td>
<td>0.0951</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>JSD</th>
<th>RNSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-LSTM</td>
<td>0.0228</td>
<td>0.0933</td>
</tr>
<tr>
<td>W/ gating mechanism</td>
<td>0.0244</td>
<td>0.0960</td>
</tr>
<tr>
<td>W/ memory enhance</td>
<td>0.0234</td>
<td>0.0941</td>
</tr>
</tbody>
</table>
Learning Curve of Different Training Data Size for ND

For ND subtask

- Both JSD and RNSS reduce when adding % of training data until 100%
- The tendency shows our model could perform better if there is more training data
- We do not apply a complex model for ND since the lack of training data
Conclusion
Conclusion

1. We propose two hierarchical models for DQ and ND subtasks
2. We compare the models w/ & w/o gating mechanism & memory enhance
   ◦ Both improve the performance of DQ subtask
   ◦ But drop the performance of ND subtask
3. Data for ND might be insufficient which cause overfitting in complex models
4. We compare sentence representation between BERT and word2vec
5. Our models outperform other organizer baseline models in ND & DQ subtasks
Nugget Types for ND

CNUG0: Customer trigger
  ◦ Problem stated

CNUG*: Customer goal
  ◦ Solution confirmed

CNUG: Customer regular
  ◦ Contains info that leads to solution

CNaN: Customer Not-a-Nugget
  ◦ Does not contain info that leads to solution

HNUG*: Helpdesk goal
  ◦ Solution stated

HNUG: Helpdesk regular
  ◦ contains info that leads to solution

HNaN: Helpdesk Not-a-Nugget
  ◦ Does not contain info that leads to solution
**Example of ND**

C: I copied a picture from my PC to my mobile phone, but it kind of looks fuzzy on the phone. How can I solve this? P.S. I’m no good at computers and mobile phones.

H: Please synchronise your PC and phone using iTunes first, and then upload your picture.

C: I’d done the synchronization but did not upload it with XXX Mobile Assistant. I managed to do so by following your advice. You are a real expert, thank you!

H: You are very welcome. If you have any problems using XXX Mobile Phone Software, please contact us again, or visit XXX.com.
Measures of DQ

A-score: Task Accomplishment
- Has the problem been solved? To what extent?

E-score: Dialogue Effectiveness
- Do the utterers interact effectively to solve the problem efficiently?

S-score: Customer Satisfaction of the dialogue
- Not of the product/service or the company

Scale: -2, -1, 0, 1, 2
Related Work

Short Text Conversation (STC)

Word Embedding to BERT
Short Text Conversation (STC)

Traditional machine learning methods
- Hidden Markov Model (Stolcke et al. 2006)
- Naïve Bayes (Lendvai and Geertzen 2007)

Deep learning methods
- CNN based & RNN based models (Lee, J., Y., Dernoncourt, F. 2016)
- Recurrent convolutional neural networks (Blunsom, P., Kalchbrenner, N. 2013)
- LSTM + CRF model (Huang, Z., Xu, W., Yu, K. 2015; Ma, X., Hovy, E. 2016)
- Hierarchical CNN + CNN / Bi-LSTM (Liu, Y., Han, K., Tan, Z., Lei, Y. 2017)
- Hierarchical encoder with CRF (Kumar, H., Agarwal, A., Dasgupta, R., Joshi, S., Kumar, A. 2018)
Word Embedding to BERT

Word embedding
- Word2Vec (Mikolov, T., Chen, K., Corrado, G., Dean, J. 2013)
- Our proposed models apply word2vec with skip-gram algorithm

BERT (Bidirectional Encoder Representations from Transformers)
- An pre-train model based on multiple bi-directional transformer blocks
- Redefines the state of the art for 11 natural language processing tasks
- BERT (Devlin, J., Chang, M, W., Lee, K., Toutanova, K. 2018)

Transformer
- Constructed by self-attention and feed-forward neural networks (without any CNN, RNN)
Two Evaluation Measures of DQ

Normalized Match Distance (NMD)

Cross-Bins Measures

Definition

- There are 2 normalized distributions \( (p, q) \)
- \( cp(i) = \sum_{k=0}^{i} p(k) \)
- \( cq(i) = \sum_{k=0}^{i} q(k) \)
- \( MD(p, q) = \sum_{i}|cp(i) - cq(i)| \)
- \( NMD(p, q) = \frac{MD(p, q)}{\text{length}-1} \)

Example

- \( p = [0,0,1] \)
- \( q_1 = [0.2,0.8,0], q_2 = [0.8,0.2,0] \)
- \( cp = [0,0,1] \)
- \( cq_1 = [0.2,1,1], cq_2 = [0.8,1,1] \)
- \( NMD(p, q_1) = \frac{0.2+1+0}{3-1} = 0.6 \)
- \( NMD(p, q_2) = \frac{0.8+1+0}{3-1} = 0.9 \)
- \( q_1 \) is better than \( q_2 \)
Two Evaluation Measures of DQ (cont.)

Root Symmetric Normalized Order-Aware Divergence (RSNOD)

Cross-Bins Measures consider the distance between a pair of bins

Definition

- There are 2 normalized distributions \((p, q)\)
- \(DW(i) = \sum_j|i - j|(p(j) - q(j))^2\)
- \(OD(p, q) = \frac{1}{|B^*|}\sum_{i \in B^*}DW(i), \ B^* = \{i|p(i) > 0\}\)
- \(SymmetricOD(p, q) = \frac{OD(p, q) + OD(q, p)}{2}\)
- \(RSNOD(p, q) = \sqrt{\frac{SymmetricOD(p, q)}{\text{length}-1}}\)

DW: Distance-Weighted sum of squares
OD: Order-Aware Divergence
Two Evaluation Measures of ND

**Jensen-Shannon divergence (JSD)**

Evaluate the similarity between 2 normalized distributions

**Definition**

- If there are 2 normalized distributions \((p, q)\)
- Define \(m = \frac{1}{2}[p(+q)]\) (element-wise addition)
- \(JSD = \frac{1}{2}[KL(p, m) + KL(q, m)]\) with log base = 2
- \(0 \leq JSD \leq 1\)

The lower \(JSD\) means the similar distributions
Two Evaluation Measures of ND (cont.)

**Root Normalized Sum of Squared Errors (RNSS)**

Evaluate the similarity between 2 normalized distributions

**Definition**

- If there are 2 normalized distributions \((p, q)\)
- \[ RNSS = \sqrt{\frac{\sum_i (p_i - q_i)^2}{2}} \]
- \(0 \leq RNSS \leq 1\)

The lower \(RNSS\) means the similar distributions
ND as a traditional sequence labeling problem

ND subtasks take label probability distribution as label
- We could only apply softmax layer instead of CRF layer

We consider the ND subtask as a traditional sequence labeling problem
- Convert the label distribution to one-hot labeling
- Solve the ND subtask by CRF instead of softmax
- Evaluate the performance by precision / recall / f1-score
Preprocessing

Distribution labels -> one-hot labels
- Choice the nugget with highest probability as label

For labels with 2 highest probability nuggets
- Create 2 one-hot labels for both nuggets as golden answers

<table>
<thead>
<tr>
<th>Nugget</th>
<th>CNUG*</th>
<th>CNUG</th>
<th>CNaN</th>
<th>CNUG0</th>
<th>HNUG*</th>
<th>HNUG</th>
<th>HNaN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Label</td>
<td>0.158</td>
<td>0.421</td>
<td>0.421</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>One-hot Label 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>One-hot Label 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
ND as sequence labeling Performance

HCNN-BERT outperform HCNN-skipGram in accuracy, macro P and macro F.

Accuracy is much more higher than macro P/R/F

- Some nugget types are difficult to correctly recognized.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Macro P</th>
<th>Macro R</th>
<th>Macro F</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCNN-skipGram</td>
<td>88.8%</td>
<td>75.6%</td>
<td>74.8%</td>
<td>75.2%</td>
</tr>
<tr>
<td>HCNN-BERT</td>
<td>89.9%</td>
<td>83.4%</td>
<td>74.6%</td>
<td>78.7%</td>
</tr>
</tbody>
</table>
Confusion Matrix

Rows: prediction / Columns: Label

Nugget pairs that are easily confused

- [CNUG, CNaN]
- [CNUG*, CNUG]
- [CNUG*, CNaN]
- [HNUG*, HNUG]
- [HNUG, HNaN]

We doubt that whether these pairs are also confused by human
Confusion of Human Annotation

The table shows the avg probability difference of 2 highest nugget of utterances

- The higher difference means the higher probability to confused by human

The easily confused nugget pairs of models

- [CNUG, CNaN]
- [CNUG*, CNUG]
- [CNUG*, CNaN]
- [HNUG*, HNU]
- [HNUG, HNaN]

Are also with confused by human

<table>
<thead>
<tr>
<th>Nugget pair</th>
<th>Avg prob diff</th>
<th># Pairs</th>
<th>Pct %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNUG0, CNUG*</td>
<td>0.842</td>
<td>13</td>
<td>0%</td>
</tr>
<tr>
<td>CNUG0, CNUG</td>
<td>0.731</td>
<td>242</td>
<td>3%</td>
</tr>
<tr>
<td>CNUG0, CNaN</td>
<td>0.696</td>
<td>1,508</td>
<td>22%</td>
</tr>
<tr>
<td>CNUG*, CNUG</td>
<td>0.348</td>
<td>232</td>
<td>3%</td>
</tr>
<tr>
<td>CNUG*, CNaN</td>
<td>0.339</td>
<td>36</td>
<td>1%</td>
</tr>
<tr>
<td>CNUG, CNaN</td>
<td>0.455</td>
<td>1,793</td>
<td>26%</td>
</tr>
<tr>
<td>HNU*, HNU</td>
<td>0.307</td>
<td>865</td>
<td>13%</td>
</tr>
<tr>
<td>HNU*, HNaN</td>
<td>0.118</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>HNU*, HNaN</td>
<td>0.401</td>
<td>2,220</td>
<td>32%</td>
</tr>
</tbody>
</table>