# Summary

# Combining TopN (N-Best)

## Binary Switch Model

In this approach, the classifier will make a binary decision about whether the corresponding NLU is correct or not. The binary choice can serve as a binary switch to control whether to pick up the NLU or not.

In this method, all of NLU used the top ASR [It is not straightforward to get which ASR the NLU comes from. For example, the second NLU is not necessarily coming from the second ASR.]

### Result

For training, I used all SLU candidates (43055 SLU in total; on average, one user turn has 3.69 SLU). For each SLU, the following feature is exacted:

* The top ASR
* The Acts from System and from User

When prediction, if the current SLU is classified as “0” (not correct), it will be ignored. If it is “1”, the SLU confidence score for the slots and values identified in this SLU is added. Then, as the same as the *baseline*, the top slot-value will be the answer.

The results are shown in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Accuracy | L2 |
| method | test | Joint\_Goals | Joint\_Goals |
| baseline | dstc2\_train | 0.582289 | 0.810131 |
| baseline | dstc2\_dev | 0.501043 | 0.960854 |
|  |  |  |  |
| HWUbaseline | dstc2\_train | 0.73196 | 0.451223 |
| HWUbaseline | dstc2\_dev | 0.623045 | 0.601356 |
|  |  |  |  |
| 2waymodel\_actngram | dstc2\_train | 0.673652 | 0.652696 |
| 2waymodel\_actngram | dstc2\_dev | 0.587591 | 0.824818 |
|  |  |  |  |
| binaryswitch\_actngram | dstc2\_train | 0.502499 | 0.96989 |
| binaryswitch\_actngram | dstc2\_dev | 0.450209 | 1.065232 |

Observation:

* The *binaryswitch* model is the worst. It is even worse than the *baseline*.

There are two reasons that make the *binaryswitch* model bad:

* Issue 1: The binaryswitch is not good when the mind can change. For the baseline, after each turn, the confidence score will be normalized; in this way, if there are new slot-values, the score for the old one will decrease after moralization. And it includes all possible slot-values, no matter correct one and in-correct one. Thus, the previous scores will be diluted with the turns going. However, the binaryswitch model considered only the correct one and it ignores most of the SLU. Thus, the scores are less diluted compared to the baseline, which makes it hard to change the goal. For example, last turn, “inform(food=chinese)” has score 0.97; next turn, “inform(food=mexican)” has score 0.8. In this case, the goal will be Chinese food because it has a higher confidence score. Thus, only when a new goal has a higher confidence score, the goal could be changed.
* Issue 2: When training, all the SLU candidates are included, which makes the training data set noisy when compared to using just the top one. Therefore, the classifier performance will be worse.

## New Binary Switch Models

Thus, I change the binary switch model in two ways, before the testing, I want to know what the topline is for this approach, first: Change the way for confidence score; Change the training data

### Change the way for confidence Score

* No Change: Leave the confidence score alone
* Always One: always use 1.0 as the confidence score. In this way, it ignores the history because the new one is always the top one
* DecayModel: the confidence will become half for the following turn. For example, for a score 0.8, it will become to 0.4 for the following turns.

The results are shown below:

Observations:

* Always One > DecayModel > No Change

From the observation, I used the “always one” model for testing.

The result is shown below:

* The model is trained on all candidate
* Feature: Act + Unigram
* Model: SVM

Observations:

* It suffers from the *Issue 2.* An evidence is when topK = 0 (Consider only the top 1), the performance is worse than the naïve 2waymodel (accuracy = 0.67)
* The performance drops with increasing the number of topK. It means the classifier doesn’t do well on the other SLU. The reason is possible I used the same ASR for all SLU.

### Change the Training Data

To tackle the *issue 2,* I changed the training data:

* All: Use all candidates as training
* Top 3: Use the top 3 in each turn as training
* Top 1: Use only the top 1 for each turn as training

## Results:

All based on “always One”.

Although, training on the Top1 improves the performance, however, it still drops with the increase of TopK. And it is no better than HWUBaseline.

## First Correct Model

Another natural extension of the binaryswitch model is to consider only the first correct SLU and ignore the rest.

The result on this model is:

Now, Training on top1 is the best.

The best result is when topK=2, which has accuracy 0.705 on the training and 0.597 on the testing.

# Method Classifier

The system and user can have same dialog acts. However, only the dialog acts in the user turn affect “method”. Thus, it is better to differentiate them.

In addition, only the name of dialog act doesn’t help a lot to identify “method”. For example, “inform(food=chinese)” has the same act name as “inform(name=X)”. However, method for “inform(food=chinese)” should be “by constraints”; the other is “by name”. Therefore, it might be better to include the slot name into the feature.

So, for the method classifier:

* Add slot-name into feature
* Differentiate System and User Acts.

For example, “inform(area=north)” in the system are different from “inform(area=north)” in the user. The feature for this example, will be “system\_inform\_area” and “user\_inform\_area”.

Result:

* “MindChangeMethodClassifier\_actngramWithSlotName” is the new model.
* It is slightly better on the “dev”; slightly worse on the “train”. It means that the generality for the new model is better.

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Method\_accuracy | Method\_l2 |
| HWUbaseline | dstc2\_train | 0.8880084 | 0.1650613 |
| HWUbaseline | dstc2\_dev | 0.8603207 | 0.2168224 |
|  |  |  |  |
| 2waymodel\_actngram | dstc2\_train | 0.8902726 | 0.2194548 |
| 2waymodel\_actngram | dstc2\_dev | 0.8644594 | 0.2710812 |
|  |  |  |  |
| MindChangeMethodClassifier\_actngram | dstc2\_train | 0.9420883 | 0.1158234 |
| MindChangeMethodClassifier\_actngram | dstc2\_dev | 0.8942059 | 0.2115882 |
|  |  |  |  |
| MindChangeMethodClassifier\_actngramWithSlotName | dstc2\_train | 0.9416529 | 0.1166942 |
| MindChangeMethodClassifier\_actngramWithSlotName | dstc2\_dev | 0.8993792 | 0.2012416 |

# Build my Own SLU: From ASR directly to Goals

First Observations:

* All the slot-value pairs appear in the given ontology, except “dontcare”.
* Only 4 slots are in the goals: area, food, name, pricerange
* Number of Possible slot values are limited

|  |  |  |  |
| --- | --- | --- | --- |
| Slot | | number of values | value examples |
| Goals | Area | 5 | centre,north,west,south,east |
| food | 91 | catalan,chinese,christmas,corsica,creative,crossover,cuban |
| name | 113 | ali baba,anatolia,ask,backstreet bistro,bangkok city,bedouin |
| pricerange | 3 | cheap,moderate,expensive |

Thus, we can build classifiers to test whether there is such slot in this turn; Then, for “area” and “pricerange”, just build 3-way classifier and 5-way classifier; For “food” and “name”, we can we can use un-supervised method to find the possible values according to the ontology [like edit distance] (where ASR error correction applies). Actually, we can combine the two steps into one classification. Take area as an example, it is a 7-way classifier: ***Area.No***, ***Area.Yes.Dontcare***, Area.Yes.Centre, Area.Yes.North, Area.Yes.West, Area.Yes.South, Area.Yes.East.

For food and name, only the first step will perform: binary classification.

# MindChang Model for Goals: make it independent

However, it has the similar issue as “method”: depending on the previous turn.

The MindChange model will make the goals independent with the previous one. It relies on the fact that only the changed goals matter. If the annotated goals don’t change, we can assume the user didn’t say new information in the current turn; thus, we can assume there are no goals in the turn.

Following this idea, one rule is used to recovery the current method label from the annotation:

* If the goal is only in the current turn and not in the previous one, use the labelled one as golden standard; Or else, assume the goal is not in the current turn.

## Results:

Only the top ASR and top NLU are considered.

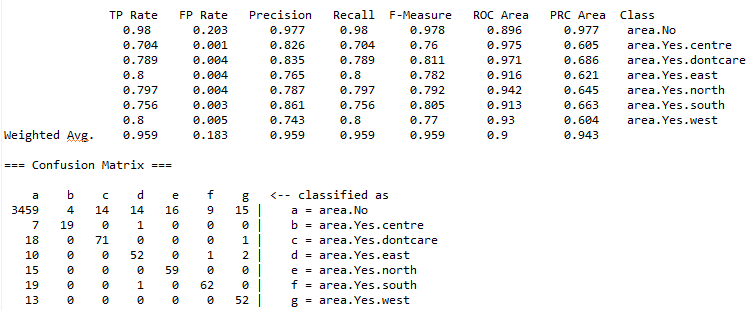
Feature Sets:

* Act+Unigram
* Unigram, Act with Name (shown in the “method” session)
* Enrich
  + Unigram, Act with Name
  + Word count
  + ASR Score
  + SLU Score
  + Speech rate
  + Cosin(System Transcription, ASR)
  + Cosin(ASR\_t, ASR\_{t-1})

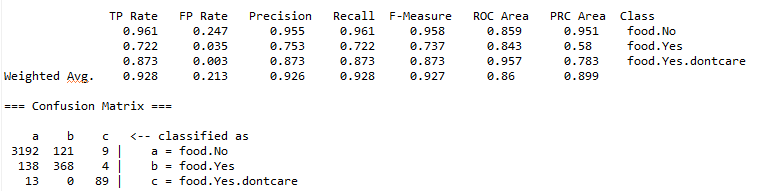
Model: SVM

### Classification (Enrich Feature Set, Dev data set):

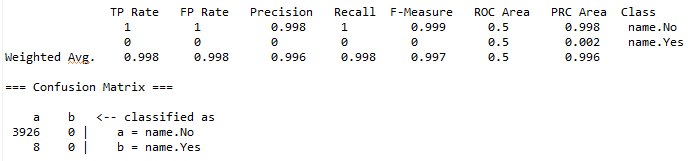
#### Area



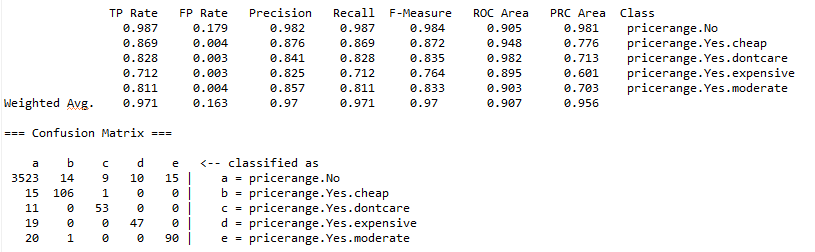
#### Food



#### Name



#### PriceRange



### Combined Result:

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_actngram\_goals | dstc2\_train | 0.521 | 0.958 |
| 2waymodel\_actngram\_goals | dstc2\_dev | 0.423 | 1.154 |
|  |  |  |  |
| 2waymodel\_actWithNamengram\_goals | dstc2\_train | 0.712 | 0.575 |
| 2waymodel\_actWithNamengram\_goals | dstc2\_dev | 0.555 | 0.889 |
|  |  |  |  |
| 2waymodel\_enrich\_goals | dstc2\_train | 0.714 | 0.572 |
| 2waymodel\_enrich\_goals | dstc2\_dev | 0.558 | 0.884 |

## Topline

1. To test the limitation of this method, the topline is computed as following:

* Use the correct label for each of the four slots

1. In addition, I also tested how ASR influences the performance. Now, I trained the model with the correct transcription instead of ASR. And test on the correct transcription, the result is shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_actngram\_goals\_topline | dstc2\_train | 0.817 | 0.366 |
| 2waymodel\_actngram\_goals\_topline | dstc2\_dev | 0.770 | 0.459 |
|  |  |  |  |
| 2waymodel\_enrich\_goals\_trans | dstc2\_train | 0.804 | 0.392 |
| 2waymodel\_enrich\_goals\_trans | dstc2\_dev | 0.711 | 0.577 |

## Observation:

* The topline for this method is better than the topline for the 2waymodel (0.68751). It is also better than topline1 (0.73196, if the goal appears in the SLU, it is correct) because it considers the ASR errors. But it is worse than topline 2(0.85322, if the goal appears in the top ASR, it is correct)
* If the ASR is correct, the performance will be much better.

# Conclusion:

The best result I got (Based on Training):

The goals are got with First correct model with training on top1;

The Request model is based on multi-label classification model.

The Method is based on MindChange Classification model with ActNgram features.



In addition, the Roc.V2.05 is still 0.