# Summary

# Method Classifier

## Take advantage of the previous tag

### Topline: use the correct tag when prediction

### Online Model

Use the previous predicted tag as the feature. Thus, the prediction step is: 1) predict the first instance by assuming the previous tag is “none”; 2) predict the next one by adding the previous predicted tag

### MindChange Model:

The current method prediction depends on the previous one only if the current method is unknown. Thus, if the current method cannot be predicted, treat it as “none”. In this way, the history can be ignored. At the same time, when predicting, the previous method will be used if the predicted method is “none”. [We can treat this model is to just annotate the method based on the current turn.]

Two rules are used to recovery the current method label from the annotation:

* When the current labelled method is ***different*** from the previous one, use the labelled method as golden standard because the user changed his mind.
* When the current labelled method is the ***same*** as the previous one, if the top SLU predicts the same method, used the labelled one; If the top SLU predicts the wrong method, use “none” as the golden method.

## Intrinsic Results (The 5-way Method Classifier Performance)

Use SVM with features Act+Unigram. The “recall” here is the “accuracy” since it is weighted.



## Extrinsic Results (Overall Method Tracking Performance in the real Challenge)

* The MindChange model got the best results, both on the accuracy and the L2. [Topline is just for reference]



# Request Classifier

For “request”, only 8 things are “requestable”, so we can build a classifier for each one by asking “Is address requested?”, “Is phone requested?”, etc.

This is not simple 8-way classification problem. Instead, it is multi-label classification because the prediction is no longer one label but several labels. Two requests can happen in one turn.

The toolkit I use for the multi-label classification is [Mulan](http://mulan.sourceforge.net/index.html) [1].

## Results:

The results for requested is the best. It is much better than the previous models.

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Requested\_Accuracy | Requested\_L2 |
| HWUbaseline | dstc2\_train | 0.9165429 | 0.1402343 |
| HWUbaseline | dstc2\_dev | 0.9026325 | 0.1642966 |
|  |  |  |  |
| Firstcorrect (topK=0) | dstc2\_train | 0.9118163 | 0.1763673 |
| Firstcorrect (topK=0) | dstc2\_dev | 0.9080418 | 0.1839163 |
|  |  |  |  |
| RequestClassifier\_actngram | dstc2\_train | 0.9779878 | 0.0440243 |
| RequestClassifier\_actngram | dstc2\_dev | 0.9635774 | 0.0728453 |

Why it is good:

* The fact of this approach is to build a new model for “request”
* It ignores the request slot in SLU but directly gets results from both the system dialog acts and the input dialog acts, and from the ASR also
* It considered the ASR errors
  + i.e.,

User: “phone number”

ASR: “i don't the number”

For this method, request(phone)=1. However, SLU cannot recognize it.

* It can get multiple requests
  + i.e.

“could i have the address and phone number” -> request(addr) = 1, request(phone)=1

However, for SLU, only request(addr) is recognizied.

# 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.

Therefore, 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

Not Finished

## Bayes Rules Model?

TODO

# Reference

[1] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, I. Vlahavas (2011) "Mulan: A Java Library for Multi-Label Learning", Journal of Machine Learning Research, 12, pp. 2411-2414.