# Summary: beat the baseline

## Longer history features for CRF

Instead of just use the current unigram as features, I add following features to CRF

* The word in last sentence
* The word in the sentence before the last sentence
* Last label

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_enrich\_more\_goals | dstc2\_train | 0.711 | 0.577 |
| 2waymodel\_enrich\_more\_goals | dstc2\_dev | 0.561 | 0.878 |
|  |  |  |  |
| HWUbaseline | dstc2\_train | 0.732 | 0.451 |
| HWUbaseline | dstc2\_dev | 0.623 | 0.601 |
|  |  |  |  |
| 2waymodel\_goals\_crf | dstc2\_train | 0.718 | 0.564 |
| 2waymodel\_goals\_crf | dstc2\_dev | 0.560 | 0.881 |
|  |  |  |  |
| 2waymodel\_goals\_crf2 | dstc2\_train | 0.728 | 0.543 |
| 2waymodel\_goals\_crf2 | dstc2\_dev | 0.620 | 0.759 |

It turns out this CRF does improve the performance. It is close to the HWUBaseline now.

## Just replace the goals (pricerange and area) with NBest Model

The NBest model leverages the Ngram in the ASR. First of all, it is good to capture the SLU errros on “pricerange” and “area” because there are only a limited number of words in these two slots. At the same time, it is easy to be overfitting on the goal of “food” and “name”, because, many of the food or name just appears once and it cannot deal with out of vocabulary (OOV) problem in the test set.

Thus, it is reasonable to just rely on the NBest model for “pricerange” and “area”. The other two goals rely on the feature\_enrich model.

In addition, in the old model, for “food”, if it is predicted as “Yes”, I will just pick up the first available SLU with a food slot. However, I didn’t consider the situation when the slot is “food=dontcare”. Thus, after taking care of this small issue, the performance increased a little bit.

### *Result*

* 2waymodel\_goals\_nbest\_price\_area: NBest “price” and “area”; Enrich “food” and “name”
* 2waymodel\_goals\_nbest\_price\_area\_dontcare: fixed the “dontcare” issue

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_enrich\_more\_goals | dstc2\_train | 0.711 | 0.577 |
| 2waymodel\_enrich\_more\_goals | dstc2\_dev | 0.561 | 0.878 |
|  |  |  |  |
| HWUbaseline | dstc2\_train | 0.732 | 0.451 |
| HWUbaseline | dstc2\_dev | 0.623 | 0.601 |
|  |  |  |  |
| 2waymodel\_goals\_nbest\_price\_area | dstc2\_train | 0.749 | 0.501 |
| 2waymodel\_goals\_nbest\_price\_area | dstc2\_dev | 0.648 | 0.705 |
|  |  |  |  |
| 2waymodel\_goals\_nbest\_price\_area\_dontcare | dstc2\_train | 0.749 | 0.502 |
| 2waymodel\_goals\_nbest\_price\_area\_dontcare | dstc2\_dev | 0.650 | 0.700 |

This is the first model which beats the HWUbaseline.

## Combine HWUBaseline and N-Best Model: Leveraging both SLU and ASR

The HWUBaseline considers only the SLU and the N-Best considers only the ASR. Therefore, combining them might get a better performance.

#### Use the food, name results from HWUBaseline and Use the pricerange and area result from N-Best

After a closing analysis, I suspect the HWUBaseline is better at prediction the food and name slot. Thus, instead of using the feature\_enrich model, I just use the label given by the HWUBaseline. For example, if the food slot is given by the HWUBaseline, I will convert it to “Yes” so that it fits in the classifier framework.

### *Results:*

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| HWUbaseline | dstc2\_train | 0.732 | 0.451 |
| HWUbaseline | dstc2\_dev | 0.623 | 0.601 |
|  |  |  |  |
| 2waymodel\_goals\_nbest\_price\_area\_dontcare\_hwu | dstc2\_train | 0.780 | 0.439 |
| 2waymodel\_goals\_nbest\_price\_area\_dontcare\_hwu | dstc2\_dev | 0.664 | 0.671 |

### Observations:

* It works very well, which is much better than the HWUbaseline.
* It means two things
  + HWUbaseline did a good job on “food”, “name” slots
  + NBest model did a better job on “pricerange” and “area” slots

Base on theser observations, we can do one-step further for the “food” slots by just used the labels directly from HWUbaseline instead of using a binary classifier model. [Errors from “name” are very little.]

The new result is:

* 2waymodel\_goals\_nbest\_hwu\_food: Nbest “price”, “name” and “area”; HWUbaseline “food”
* 2waymodel\_goals\_crf2\_hwu\_food: CRF2 “price”, “name” and “area”; HWUbaseline “food”

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_goals\_nbest\_hwu\_food | dstc2\_train | 0.783 | 0.433 |
| 2waymodel\_goals\_nbest\_hwu\_food | dstc2\_dev | 0.679 | 0.643 |
|  |  |  |  |
| 2waymodel\_goals\_crf2\_hwu\_food | dstc2\_train | 0.764 | 0.473 |
| 2waymodel\_goals\_crf2\_hwu\_food | dstc2\_dev | 0.639 | 0.722 |

In this case, we don’t need to find what is the actual food it is when prediction the food slot as “food.Yes”.

This is the best model I have now.

#### Voting with HWUBaseline, Baseline\_Focus, and N-Best

Another straightforward idea to combine they is to just use voting.

I tried two combinations for the voting

* 2waymodel\_goals\_voting: voting on all the four goals
* 2waymodel\_goals\_nbest\_price\_area\_food\_name\_voting: voting only on the “food” and “name” and use the nbest model result for “price” and “area”

### *Result:*

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_goals\_nbest\_price\_area\_food\_name\_voting | dstc2\_train | 0.781 | 0.439 |
| 2waymodel\_goals\_nbest\_price\_area\_food\_name\_voting | dstc2\_dev | 0.675 | 0.650 |
|  |  |  |  |
| 2waymodel\_goals\_voting | dstc2\_train | 0.736 | 0.529 |
| 2waymodel\_goals\_voting | dstc2\_dev | 0.659 | 0.682 |

Both are better than the HWUbaseline.

#### Add them as features to do the prediction

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_goals\_bybrid | dstc2\_train | 0.780 | 0.440 |
| 2waymodel\_goals\_bybrid | dstc2\_dev | 0.657 | 0.685 |

## Class-based Model for “food” goal

|  |  |  |  |
| --- | --- | --- | --- |
| method | test | Joint\_Goals\_accuracy | Joint\_Goals\_l2 |
| 2waymodel\_goals\_nbest\_foodclass | dstc2\_train | 0.720 | 0.559 |
| 2waymodel\_goals\_nbest\_foodclass | dstc2\_dev | 0.629 | 0.741 |

## Different Classifiers for different dialog acts

Inspired by the Short Answer Assessment Task, using different classifiers for different questions might help. Now, the number of questions asked by the system is limited. Thus, it is possible to build one classifier for one question.

## Explorer Results on Test set

### Average length of turns

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | # of turns | |  | # of calls | |  | average length | |
| dstc2\_train | DM0 | DM1 |  | DM0 | DM1 |  | DM0 | DM1 |
| SR0 | 2075 | 3591 |  | 302 | 435 |  | 6.87 | 8.26 |
| SR1 | 2865 | 3146 |  | 443 | 432 |  | 6.47 | 7.28 |
|  |  |  |  |  |  |  |  |  |
| dstc2\_dev | DM0 | DM1 |  | DM0 | DM1 |  | DM0 | DM1 |
| SR0 | 593 | 1220 |  | 81 | 140 |  | 7.32 | 8.71 |
| SR1 | 938 | 1183 |  | 141 | 144 |  | 6.65 | 8.22 |
|  |  |  |  |  |  |  |  |  |
| dstc2\_test | DM2 |  |  | DM2 |  |  | DM2 |  |
| SR0 | 5105 |  |  | 542 |  |  | 9.42 |  |
| SR1 | 4785 |  |  | 575 |  |  | 8.32 |  |

As shown in the table above, the length of a dialog in Test set is much longer than the training and dev. After taking a look at the data, I found there are more repetitions in the dialog. The system can ask a same question again and again.

## Current Possible Submissions

* First-correct model
* BinarySwitch Model
* NBest model
* Voting Model
* Nbest + HWUBasline [best]