The strategic use of ambiguity in dialogue

Ella I. Eisemann

21.10.2019

Structure

First, the model will be shown, then the testing and then the plotting.

RSA-Model

The developed RSA-Modelling Framework is implemented in the following pages.

The simple listener function determines the hypothetical listener's object choice given the objects to choose from and its preferences, determining P(obj | utt, listener's object preferences).

```r
    knitr::opts_chunk$set(fig.width=4, fig.height = 3)

simpleListener <-
  function(utterance,
           mapUttToObjProbs,
           listenerObjectPreferences) {
    objPosterior <-
      mapUttToObjProbs[utterance, ] * (listenerObjectPreferences + 1e-100)
    if (sum(objPosterior) == 0) {
      return(objPosterior)
    }
    return(objPosterior / sum(objPosterior))
  }
```

The simple pragmatic speaker considers all "imaginable" (i.e. implemented) preference distributions over objects of the listener. Starting with a prior assumption over the possible listener's preferences ("preferencesPrior"). This prior has one value for each feature value possible. Only the priors for the target feature values > 0 because only they are relevant for the task. It then infers the posterior over these preferences given the listener makes a particular object choice. The priors in the beginning have a uniform distribution but with each trial the priors are the previous posterior preference presumptions. This leads to a Bayesian learning process. "utterance" is an index referring to one of the relevant utterances ("relevantUtterances) which are all present feature values in the current objects i.e. P(listener's feature value preferences | utterance, object choice by the listener, prior over preferences).

```r
simplePragmaticSpeaker <-
  function(utterance,
           obj,
           preferencesPriorAll,
           relevantUtterances,
           currentObjects,
           mapUttToObjProbs,
           objectPreferenceSoftPriors) {
    preferencesPrior <- preferencesPriorAll[relevantUtterances]
    prefPost <- rep(0, length(relevantUtterances) + 1)
    for (pref in c(1:length(preferencesPrior))) {
      # prior over the preferences the speaker is interested in
      if (preferencesPrior[pref] > 0) {
        pp <-
          simpleListener(utterance,
                         mapUttToObjProbs,
                         objectPreferenceSoftPriors[[pref]])
        prefPost[pref] <- pp[obj] * preferencesPrior[pref]
      }
    }
    for (pos in c(1:length(relevantUtterances))) {
      preferencesPriorAll[relevantUtterances[pos]] <- prefPost[pos]
    }
    if (sum(preferencesPriorAll) == 0) {
      # no evidence for any preferences... -> no inference
      return(preferencesPriorAll)
    }
    return(preferencesPriorAll / sum(preferencesPriorAll))
  }
```

To know how well the model did, the evaluation number similar to the one used in the experiment is calculated. The best is 3 and the worst is 0. This number is calculated by comparing the simulated preferences the choosing listener has with the model or human predictions. To not compare absolute or relative numbers, as they might vary in quite big ranges, the hierarchy is compared. This results in having truthfully predicted which preferences the listener has (evaluation number = 3) in contrast to not having a clue about them (evaluation number = 0).

```
evaluate <-
  function(allUtterancePref,
           preferencesPrior,
           targetFeature) {
    index <- targetFeature * 3
    indices <- c(index - 2, index - 1, index)
    tarFeaPref <- allUtterancePref[indices,]
    if (length(preferencesPrior) > 3) {
      tarFeaPrefPrior <- preferencesPrior[indices]
    } else {
      tarFeaPrefPrior <- preferencesPrior
    }
    prefRank <-
      order(as.numeric(tarFeaPref[, 3]))
    prefPriorRank <-
      order(tarFeaPrefPrior)
    if (identical(prefRank, prefPriorRank)) {
      evalNum <- 3
    } else if (identical(prefPriorRank, c(prefRank[1], prefRank[3], prefRank[2])) ||
               identical(prefPriorRank, c(prefRank[2], prefRank[1], prefRank[3]))) {
      evalNum <- 2
    } else if (identical(prefPriorRank, c(prefRank[2], prefRank[3], prefRank[1])) ||
               identical(prefPriorRank, c(prefRank[3], prefRank[1], prefRank[2]))) {
      evalNum <- 1
    } else if (identical(prefPriorRank, c(prefRank[3], prefRank[2], prefRank[1]))) {
      evalNum <- 0
    }
    return(evalNum)
  }
```

Evaluation and Model testing

The model has some determining factors for how it performs. These are the not-obey-instant and the soft-preferences-value. The not-obey-instant is a value between 0 and 1, with 0 the listener follows always it's preferences and 1 not at all. The soft-preferences-value manipulates how strong the listener's preferences are. With the value being 0 the listener has absolute preferences and absolute non-preferences. Augmenting this value leads to a uniform-prior model.

```r
notObeyInst <- 0
softPrefValue <- 1
```

Here the model gets tested with the data from the experiment. It allows direct comparison of the performance of the model with the human behavior. The data to feed the model is the data which was also fed to the participants in the experiment. For each combination of objects, utterance and previous trials the model adjusts its posterior presumptions which then become the next prior presumptions.

```r
for (worker in c(0:totalWorker)) {
  for (block in c(1:totalBlock)) {
    blockdata <-
      subset(inputData,
             blockNr == block - 1 &
               workerid == unique(inputData$workerid)[worker + 1])
    targetFeatureNum <- blockdata$targetFeatureNum[1]
    preferencesPrior <- getPreferencesPrior(targetFeatureNum)
    preferencesPriorIndices <- which(preferencesPrior != 0)
    allUtterancePref <-
      getAllUtterancePref(
        c(
          blockdata$simPreference0[1],
          blockdata$simPreference1[1],
          blockdata$simPreference2[1]
        )
      )
    ambiguousUtteranceCount <- 0
    for (trial in c(1:maxTrialNum)) {
      row <- row + 1
      currentObjects <-
        c(blockdata$orderObjNum1[trial],
          blockdata$orderObjNum2[trial],
          blockdata$orderObjNum3[trial])
      allPresentFeaValues <- determineAllFeaValues(currentObjects)
      inputData$allPresentFeaValues[row] <-
        toString(allPresentFeaValues)
      relevantUtterances <- determineValidUtterances(currentObjects)
      utteranceGeneral <- as.integer(blockdata$utteranceNum[trial])
      utterance <- which(relevantUtterances == utteranceGeneral)
      ambiguous <- isAmbiguous(allPresentFeaValues,
                               utteranceGeneral,
                               currentObjects,
                               targetFeatureNum)
      inputData$ambiguous[row] <-
        ambiguous
      ambigRatio <- countAmbigUttRatio(allPresentFeaValues,
                                       currentObjects,
                                       targetFeatureNum)
      inputData$ambigRatio[row] <- ambigRatio
      if (ambiguous) {
        ambiguousUtteranceCount <- ambiguousUtteranceCount + 1
      }
      inputData$ambiguousUtteranceCount[row] <-
        ambiguousUtteranceCount
      mapObjToUtt <-
        determineObjectToUtterancesMapping(currentObjects)
      mapUttToObjProbs <-
        determineUtteranceToObjectProbabilities(relevantUtterances,
                                                currentObjects,
                                                mapObjToUtt,
                                                notObeyInst)
      mapUttToPref <-
        getMapUttToPref(relevantUtterances, allObjects, allUtterancePref)
      objectPreferenceSoftPriors <-
        getObjectPreferencePriors(
          relevantUtterances,
          currentObjects,
          softPrefValue,
          mapUttToObjProbs,
          mapUttToPref
        )
      mapUttToObjToPref <-
        getMapUttToObjToPref(
          currentObjects,
          targetFeatureNum,
          relevantUtterances,
          allUtterancePref,
          allObjects,
          mapUttToPref
        )
      obj <-
        which(currentObjects == blockdata$simulatedAnswerObjNum[trial])
      preferencesPrior <-
        simplePragmaticSpeaker(
          utterance,
          obj,
          preferencesPrior,
          relevantUtterances,
          currentObjects,
          mapUttToObjProbs,
          objectPreferenceSoftPriors
        )
      inputData$preferencesPrior1[row] <-
        preferencesPrior[preferencesPriorIndices[1]]
      inputData$preferencesPrior2[row] <-
        preferencesPrior[preferencesPriorIndices[2]]
      inputData$preferencesPrior3[row] <-
        preferencesPrior[preferencesPriorIndices[3]]
      evalNumModel <-
        evaluate(allUtterancePref, preferencesPrior, targetFeatureNum)
      inputData$evalNumModel[row] <- evalNumModel
```

```
      humanResponse <-
        c(
          blockdata$normResponse0[trial],
          blockdata$normResponse1[trial],
          blockdata$normResponse2[trial]
        )
      evalNum <-
        evaluate(allUtterancePref, humanResponse, targetFeatureNum)
      inputData$evalNum[row] <- evalNum
    }
  }
}
```

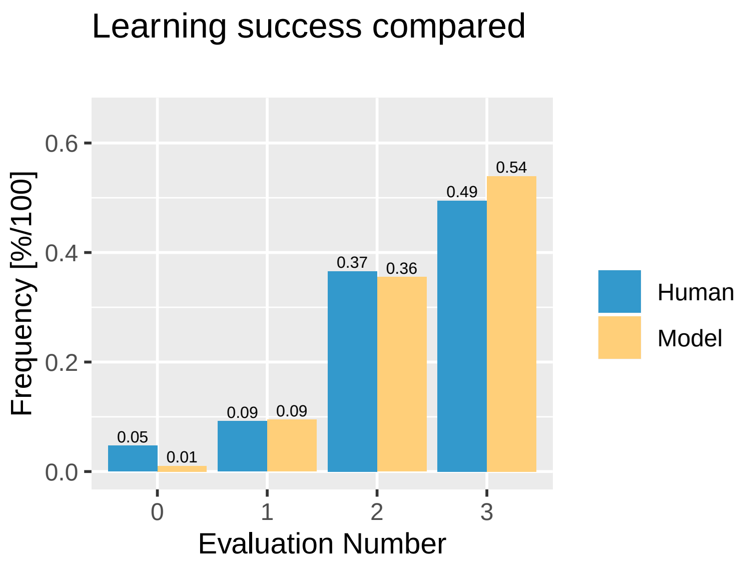
For each trial the utterance chosen by the participant is evaluated if it is ambiguous for the target feature or not. This shows which participants use ambiguity to detect information.

```
inputData$ambiguousUtteranceCount <- as.factor(inputData$ambiguousUtteranceCount)
ambiguityUsed <- matrix(nrow = totalWorker + 1, ncol = 3)
for (worker in c(0:totalWorker)) {
  ambiguityUsed[worker + 1, 1] <-
    unique(inputData$workerid)[worker + 1]
  ambiguityUsed[worker + 1, 2] <-
    round(sum(inputData$ambiguous[which(inputData$workerid == unique(inputData$workerid)[worker + 1])])
  ambiguityUsed[worker + 1, 3] <-
    sum(inputData$ambiguous[which(inputData$workerid == unique(inputData$workerid)[worker + 1])])
}
ambiguousWorker <-
  subset(ambiguityUsed, ambiguityUsed[, 2] > quantile(ambiguityUsed[, 2], 0.75))[, 1]
inputDataAmbiguous <-
  subset(inputData, workerid %in% ambiguousWorker)
nonAmbiguousWorker <-
  subset(ambiguityUsed, ambiguityUsed[, 2] < quantile(ambiguityUsed[, 2], 0.25))[, 1]
inputDataNonAmbiguous <-
  subset(inputData, workerid %in% nonAmbiguousWorker)
```

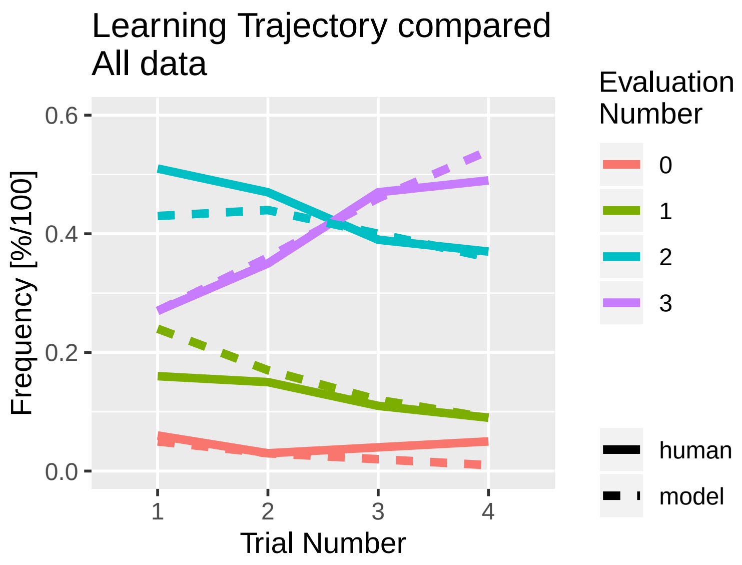
The experimental human data was cleaned for language and assurance that the task was done accurately. Only participants which specified that their native language was English were kept. Two participants with different native languages (Italian and Urdu) were excluded. This happened to avoid problems and hassles related to language barriers. Also, as the study scrutinized a psycho-linguistic phenomenon, possible interference with other native languages as English were tried to minimize. Three participants who answered to the question "Did you read the instructions and do you think you did the HIT correctly? - Yes, No or Confused" with "No" or "Confused" were excluded too. In these cases the data cannot be considered in the evaluation. (HIT is an abbreviation for Human Intelligence Task and is to be used here synonymously with "experiment".)

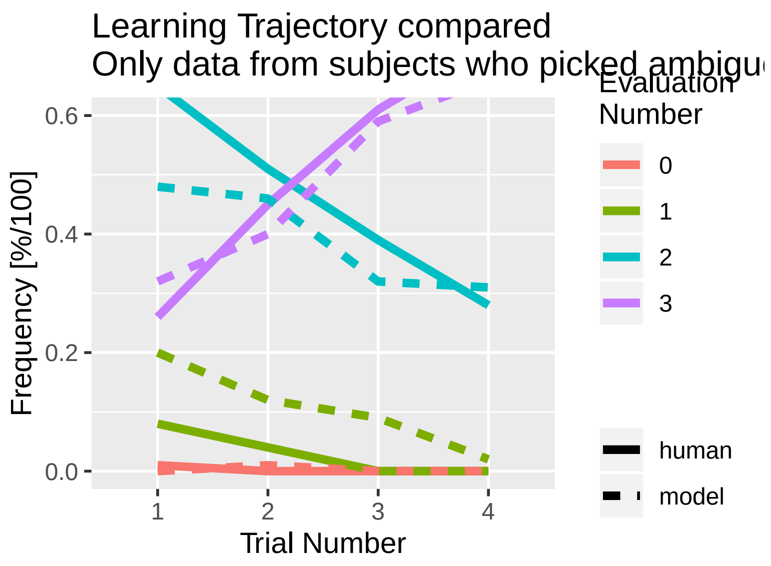
Plotting

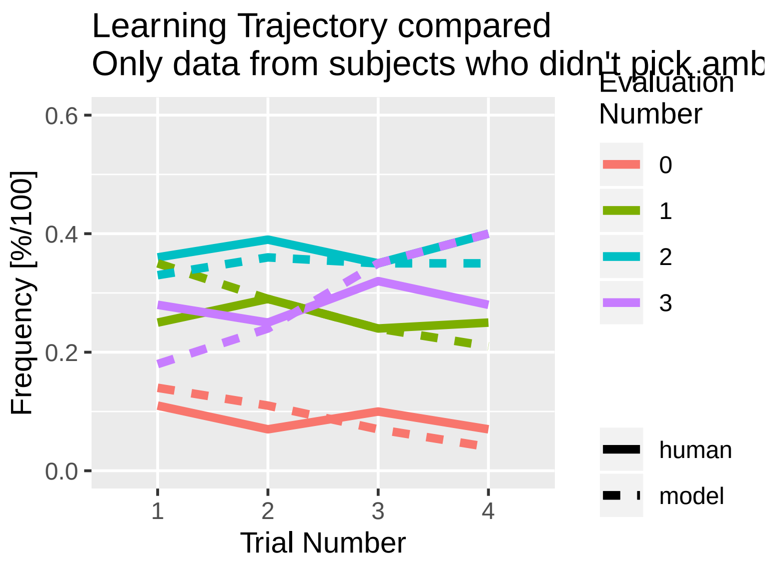
This plot shows the success rate of the model compared with the success rate of the human participants. Only the last (4the) trials in a block were considered. The success is rated in 4 levels. An "Evaluation Number" of "0" reflects an inverted guessed ranking compared with the real hierarchy of the preferences of the choosing listener, which means the guessing went totally wrong. A "3" instead reflects the right detection. These results shows that around half of the cases both participants and model detect the real hierarchy correctly. Also in the other cases (Evaluation Number 0, 1, 2) the model predictions approximate the human results accurately. All the "human" bars add up to 100% and likewise the "model" bars.

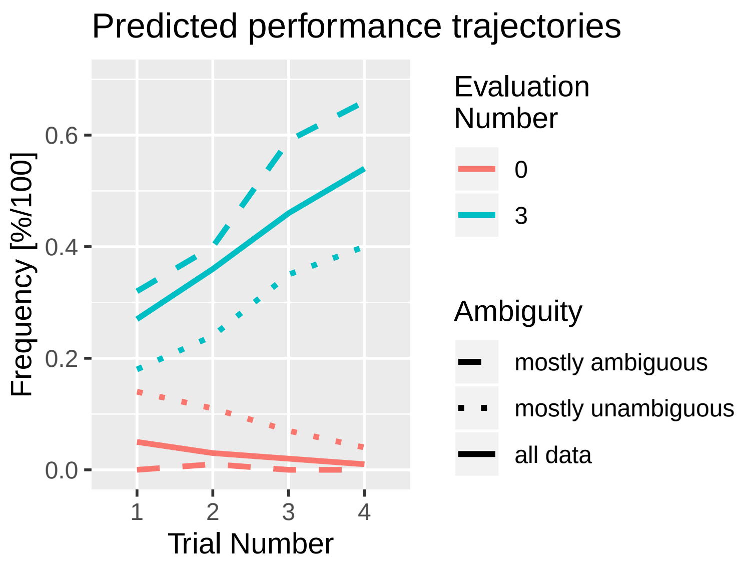


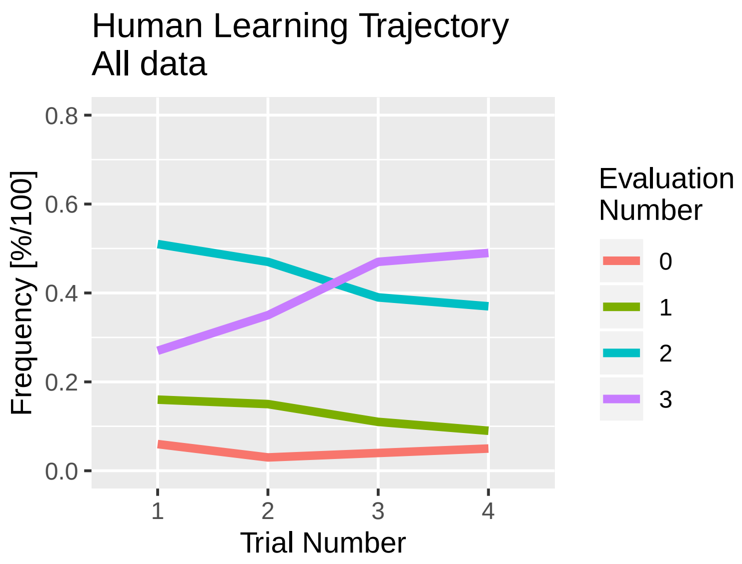
The three next plots show the learning trajectory of the experiment compared with the one of the model. Also in this case the success is rated in 4 levels as described before - the Evaluation Number. The plots are constituted out of 4 lines for each source of data (human data: normal line, model data: dotted line). Each line of one quartet stands for the frequency in percent of one Evaluation Number value over the course of all four trials. The plots differ in which data from is showed. Also in these plots it becomes clear that the model predictions match neatly to the human data.

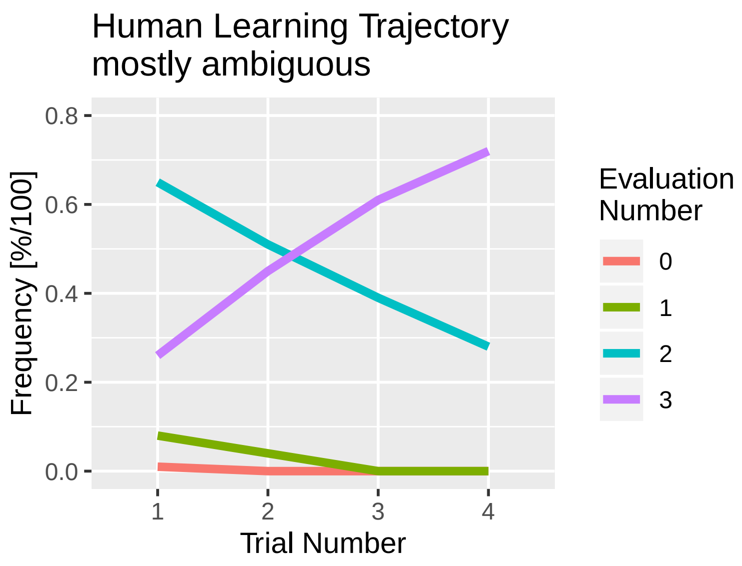


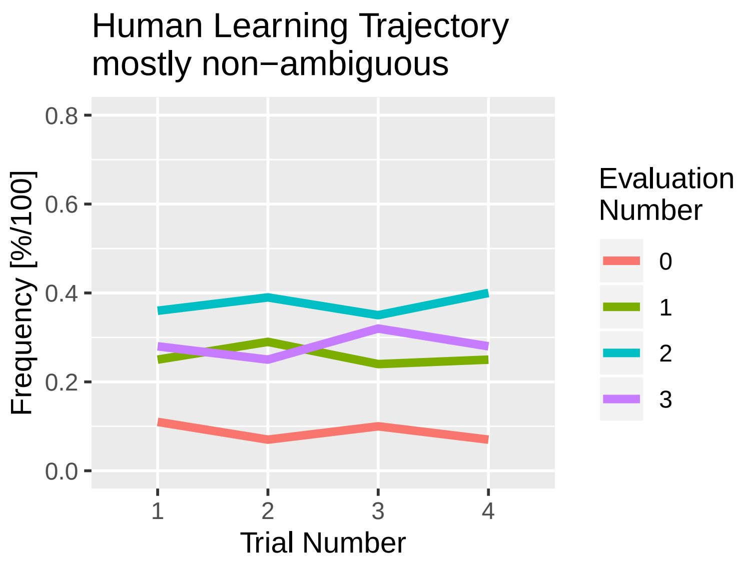

Learning Trajectory compared
Only data from subjects who picked ambigu
Evaluation Number
0
1
2
3
human
model
Frequency [%/100]
0.0
0.2
0.4
0.6
1
2
3
4
Trial Number


Learning Trajectory compared
Only data from subjects who didn't pick amb
Evaluation Number
0
1
2
3
human
model
Frequency [%/100]
0.0
0.2
0.4
0.6
1
2
3
4
Trial Number

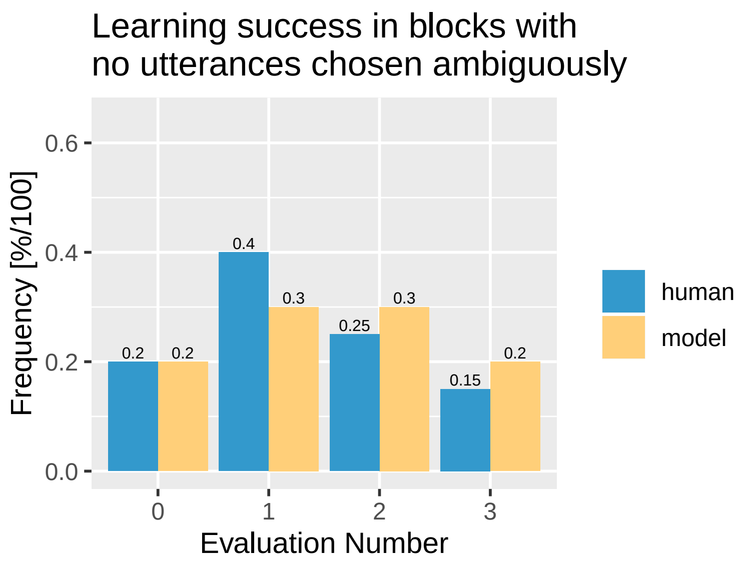

Predicted performance trajectories
Frequency [%/100]
0.0
0.2
0.4
0.6
1
2
3
4
Trial Number
Evaluation Number
0
3
Ambiguity
mostly ambiguous
mostly unambiguous
all data


Human Learning Trajectory
All data
Frequency [%/100]
0.0
0.2
0.4
0.6
0.8
1
2
3
4
Trial Number
Evaluation Number
0
1
2
3

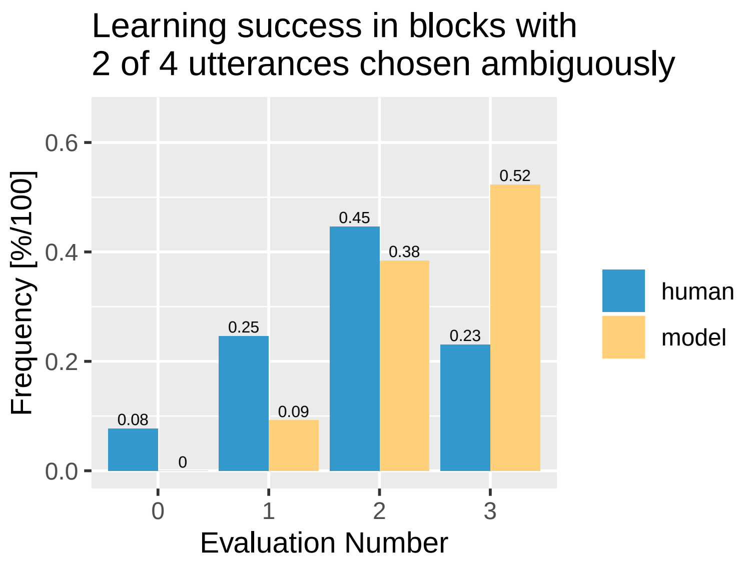




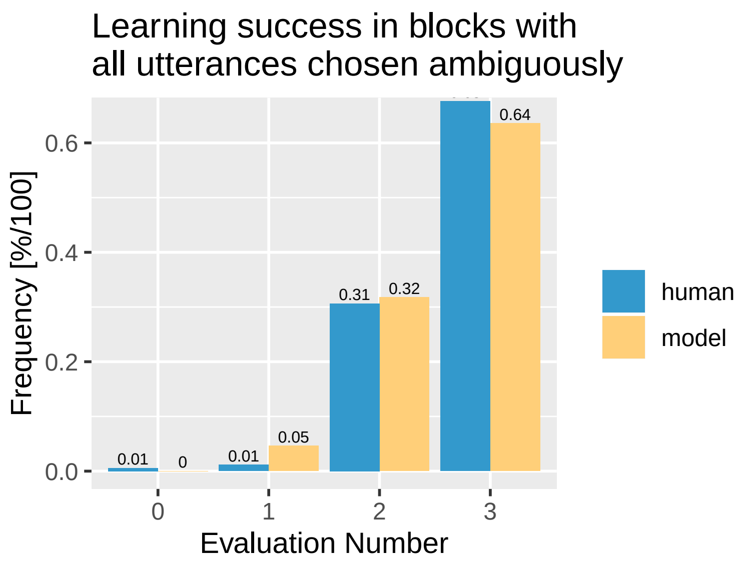
In this plot displays clearly the correlation between high ambiguity use and high preference detection success. Both for the human data and for the modelling results this correlation exists. The conclusion is that learning works best (and only) with using ambiguous utterances.



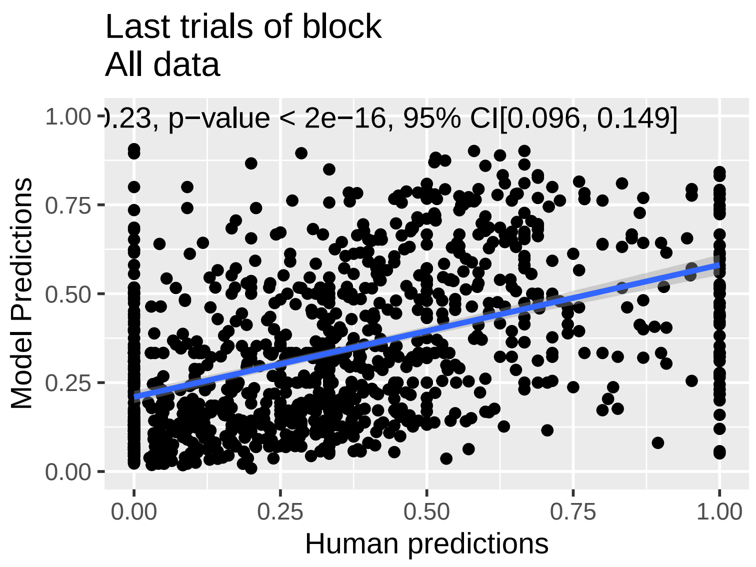
Learning success in blocks with no utterances chosen ambiguously
Frequency [%/100]
0.0
0.2
0.4
0.6
0.2
0.2
0.4
0.3
0.25
0.3
0.15
0.2
0
1
2
3
Evaluation Number
human
model

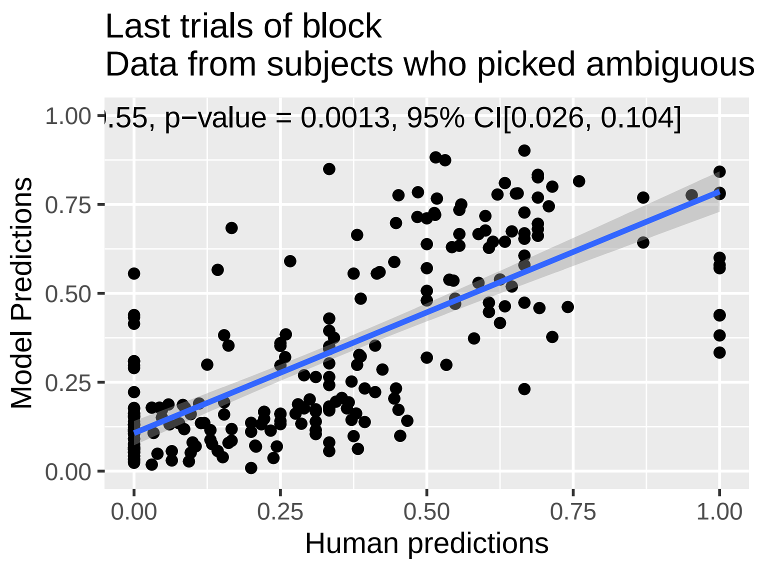


Learning success in blocks with 2 of 4 utterances chosen ambiguously
Frequency [%/100]
0.0
0.2
0.4
0.6
0.08
0
0.25
0.09
0.45
0.38
0.23
0.52
0
1
2
3
Evaluation Number
human
model

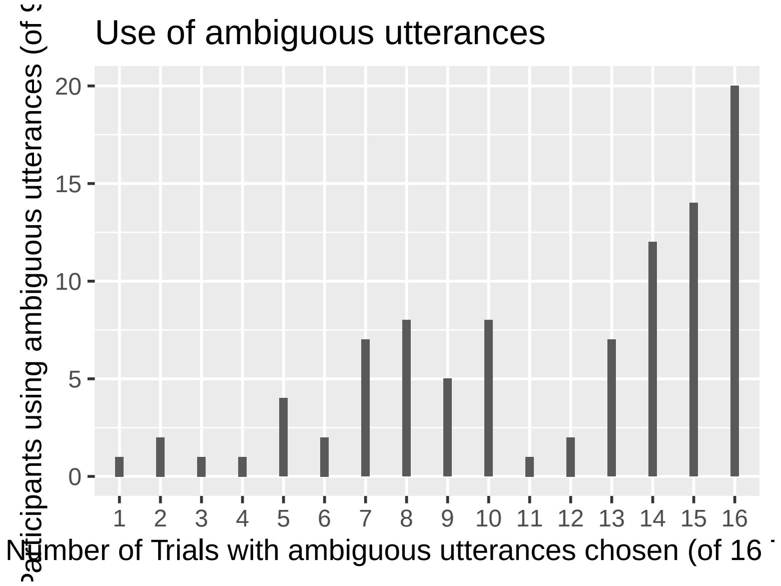


In the following scatter plots the correlation between the raw normalized values of the experiment compared with the model data is shown. For each comparison between the model predictions and the human value entered for one target feature value in one trial one point is drawn.





In this plot all the distribution over how many ambiguous utterances participants picked.



```r
summary(inputData$ambigRatio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2000  0.2500  0.5000  0.4365  0.5000  1.0000
```

```r
summary(as.factor(inputData$ambigRatio))
```

```
##               0.2              0.25               0.5 0.666666666666667
##               235               423               544               238
##                 1
##                80
```

```r
model <- clmm(evalNum ~ Answer.time_in_minutes + (1|workerid), data = inputDataCondensed)
summary(model)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
##
```

```
## formula: evalNum ~ Answer.time_in_minutes + (1 | workerid)
## data:    inputDataCondensed
##
##  link  threshold nobs logLik  AIC    niter    max.grad cond.H
##  logit flexible  380  -397.29 804.58 198(599) 4.48e-05 2.2e+03
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  workerid (Intercept) 1.189    1.091
## Number of groups:  workerid 95
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## Answer.time_in_minutes  0.02682    0.03966   0.676    0.499
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 0|1  -3.2797     0.4730  -6.934
## 1|2  -1.9690     0.4248  -4.635
## 2|3   0.2836     0.4038   0.702
```

```
model <- clmm(ambiguousUtteranceCount ~ Answer.time_in_minutes + (1|workerid), data = inputDataCondense
summary(model)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
##
## formula: ambiguousUtteranceCount ~ Answer.time_in_minutes + (1 | workerid)
## data:    inputDataCondensed
##
##  link  threshold nobs logLik  AIC    niter     max.grad cond.H
##  logit flexible  380  -454.64 921.28 280(1984) 2.09e-04 2.9e+03
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  workerid (Intercept) 5.519    2.349
## Number of groups:  workerid 95
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## Answer.time_in_minutes  0.11577    0.07335   1.578    0.115
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 0|1  -3.7867     0.7548  -5.017
## 1|2  -1.9541     0.7221  -2.706
## 2|3  -0.2893     0.7222  -0.401
## 3|4   1.4195     0.7323   1.938
```

```
model <- clmm(evalNum ~ blockNr + (1|workerid), data = inputDataCondensed)
summary(model)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
##
## formula: evalNum ~ blockNr + (1 | workerid)
## data:    inputDataCondensed
```

```
##
##  link  threshold nobs logLik  AIC    niter     max.grad cond.H
##  logit flexible  380  -396.09 806.18 362(1089) 3.56e-04 2.9e+01
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  workerid (Intercept) 1.268    1.126
## Number of groups:  workerid 95
##
## Coefficients:
##          Estimate Std. Error z value Pr(>|z|)
## blockNr1 -0.01079    0.29918  -0.036    0.971
## blockNr2  0.39922    0.29888   1.336    0.182
## blockNr3  0.28472    0.29743   0.957    0.338
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 0|1  -3.3983     0.3511  -9.679
## 1|2  -2.0672     0.2821  -7.328
## 2|3   0.2078     0.2472   0.841
```

```
model <- clmm(ambiguousUtteranceCount ~ blockNr + (1|workerid), data = inputDataCondensed)
summary(model)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
##
## formula: ambiguousUtteranceCount ~ blockNr + (1 | workerid)
## data:    inputDataCondensed
##
##  link  threshold nobs logLik  AIC    niter     max.grad cond.H
##  logit flexible  380  -448.07 912.14 599(5781) 3.76e+00 1.2e+04
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  workerid (Intercept) 6.17     2.484
## Number of groups:  workerid 95
##
## Coefficients:
##          Estimate Std. Error z value Pr(>|z|)
## blockNr1 0.788371   0.003540   222.7   <2e-16 ***
## blockNr2 0.790134   0.003616   218.5   <2e-16 ***
## blockNr3 1.207806   0.003523   342.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##      Estimate Std. Error   z value
## 0|1 -4.326530   0.003737 -1157.801
## 1|2 -2.442536   0.003357  -727.635
## 2|3 -0.730159   0.175745    -4.155
## 3|4  1.059430   0.219201     4.833
```

```
model <- clmm(ambiguousUtteranceCount ~ certainty + (1|workerid), data = inputDataCondensed)
summary(model)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
##
## formula: ambiguousUtteranceCount ~ certainty + (1 | workerid)
## data:    inputDataCondensed
##
##  link  threshold nobs logLik  AIC    niter     max.grad cond.H
##  logit flexible  380  -448.84 909.67 288(2846) 3.96e+00 1.7e+04
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  workerid (Intercept) 5.796    2.407
## Number of groups:  workerid 95
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## certainty   2.3298     0.3762   6.193  5.9e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##      Estimate Std. Error  z value
## 0|1 -3.178553   0.003720 -854.474
## 1|2 -1.332360   0.003722 -357.931
## 2|3  0.381858   0.191933    1.990
## 3|4  2.156956   0.250287    8.618
```

```r
model <- clmm(evalNum ~ certainty + (1|workerid), data = inputDataCondensed)
summary(model)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
##
## formula: evalNum ~ certainty + (1 | workerid)
## data:    inputDataCondensed
##
##  link  threshold nobs logLik  AIC    niter    max.grad cond.H
##  logit flexible  380  -387.46 784.92 203(612) 3.69e-06 7.4e+01
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  workerid (Intercept) 1.336    1.156
## Number of groups:  workerid 95
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## certainty   2.4671     0.5639   4.375 1.22e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 0|1  -1.7885     0.4791  -3.733
## 1|2  -0.4164     0.4483  -0.929
## 2|3   1.9446     0.4662   4.171
```

```r
lmLP1 <- lm(HRelativeFreq ~ MRelativeFreq, data = LP1)
summary(lmLP1)
```

```
##
## Call:
## lm(formula = HRelativeFreq ~ MRelativeFreq, data = LP1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.081370 -0.010603 -0.002137  0.012537  0.081060
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.004439   0.016054   0.276    0.786
## MRelativeFreq 0.987213   0.052668  18.744 2.59e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03693 on 14 degrees of freedom
## Multiple R-squared:  0.9617, Adjusted R-squared:  0.9589
## F-statistic: 351.3 on 1 and 14 DF,  p-value: 2.585e-11
```

```r
lmLP2 <- lm(HRelativeFreq ~ MRelativeFreq, data = LP2)
summary(lmLP2)
```

```
##
## Call:
## lm(formula = HRelativeFreq ~ MRelativeFreq, data = LP2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.11337 -0.04597  0.01516  0.03894  0.13139
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.03894    0.02420  -1.609     0.13
## MRelativeFreq  1.16157    0.07318  15.873   2.4e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06378 on 14 degrees of freedom
## Multiple R-squared:  0.9474, Adjusted R-squared:  0.9436
## F-statistic:   252 on 1 and 14 DF,  p-value: 2.402e-10
```

```r
lmLP3 <- lm(HRelativeFreq ~ MRelativeFreq, data = LP3)
summary(lmLP3)
```

```
##
## Call:
## lm(formula = HRelativeFreq ~ MRelativeFreq, data = LP3)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.095948 -0.022452  0.003628  0.035789  0.088628
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.04036    0.03317   1.217    0.244
## MRelativeFreq  0.83898    0.12100   6.934 6.94e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05377 on 14 degrees of freedom
## Multiple R-squared:  0.7745, Adjusted R-squared:  0.7584
## F-statistic: 48.08 on 1 and 14 DF,  p-value: 6.939e-06
```

Thanks for reading until here. Have a good day!