Supplementary materials for *WordSeg: Standardizing unsupervised word form segmentation from text*

alejandrina cristia

10/31/2018

Contents

Correlations across token, type, and boundary F-scores 2
Deep dive: F-scores, precision, and recall 3
Corpus size effects 16

Corpus size effects with all children **24**

```
read.table("derived/all.txt",header=T)->res


summary(res)
```

```
##  token_precision    token_recall      token_fscore     type_precision
##  Min.   :0.01138   Min.   :0.00943   Min.   :0.01660   Min.   :0.03727
##  1st Qu.:0.17947   1st Qu.:0.09529   1st Qu.:0.08513   1st Qu.:0.12028
##  Median :0.39270   Median :0.27925   Median :0.30200   Median :0.21050
##  Mean   :0.38344   Mean   :0.32892   Mean   :0.33744   Mean   :0.26987
##  3rd Qu.:0.62348   3rd Qu.:0.51615   3rd Qu.:0.58173   3rd Qu.:0.44240
##  Max.   :0.74500   Max.   :0.84090   Max.   :0.76980   Max.   :0.63660
##
##   type_recall        type_fscore       boundary_all_precision
##  Min.   :0.002901   Min.   :0.005587   Min.   :0.3330
##  1st Qu.:0.122725   1st Qu.:0.124175   1st Qu.:0.7797
##  Median :0.346450   Median :0.258300   Median :0.8875
##  Mean   :0.331580   Mean   :0.287308   Mean   :0.8324
##  3rd Qu.:0.521300   3rd Qu.:0.472500   3rd Qu.:0.9796
##  Max.   :0.701400   Max.   :0.667400   Max.   :1.0000
##
##  boundary_all_recall boundary_all_fscore boundary_noedge_precision
##  Min.   :0.2295      Min.   :0.3734      Min.   :0.2329
##  1st Qu.:0.5472      1st Qu.:0.5375      1st Qu.:0.5858
##  Median :0.7036      Median :0.7185      Median :0.7941
##  Mean   :0.6847      Mean   :0.7013      Mean   :0.7367
##  3rd Qu.:0.8547      3rd Qu.:0.8630      3rd Qu.:0.9189
##  Max.   :1.0000      Max.   :0.9302      Max.   :1.0000
##                                          NA's   :148
##  boundary_noedge_recall boundary_noedge_fscore adjusted_rand_index
##  Min.   :0.0000         Min.   :0.0000         Min.   :-0.0000016
##  1st Qu.:0.3338         1st Qu.:0.3645         1st Qu.: 0.0956350
##  Median :0.5219         Median :0.5552         Median : 0.3216500
##  Mean   :0.5357         Mean   :0.5147         Mean   : 0.3205623
##  3rd Qu.:0.7697         3rd Qu.:0.7911         3rd Qu.: 0.4981000
##  Max.   :1.0000         Max.   :0.9013         Max.   : 0.8452000
##
```

```
##                       thisres       clean_name            algo
##  eth01-phone-ag          :   1   eth01  :  16   ag          :148
##  eth01-phone-baseline-00:   1   eth02  :  16   baseline-00:148
##  eth01-phone-baseline-05:   1   eth04  :  16   baseline-05:148
##  eth01-phone-baseline-10:   1   eth17  :  16   baseline-10:148
##  eth01-phone-dibs        :   1   eth21  :  16   dibs        :148
##  eth01-phone-puddle      :   1   eth24  :  16   puddle      :148
##  (Other)                 :1178   (Other):1088   (Other)     :296
##        unit     nphone_tokens    nphone_types   nsyllable_tokens
##  phone   :592   Min.   : 4061   Min.   :38.00   Min.   :1664
##  syllable:592   1st Qu.: 8923   1st Qu.:39.00   1st Qu.:3547
##                 Median :11880   Median :40.00   Median :4748
##                 Mean   :11463   Mean   :39.62   Mean   :4581
##                 3rd Qu.:13638   3rd Qu.:40.00   3rd Qu.:5443
##                 Max.   :19141   Max.   :40.00   Max.   :7667
##
##  nsyllable_types  nwords_tokens   nwords_types   nword_hapax
##  Min.   : 302.0   Min.   :1333   Min.   : 271   Min.   : 83.0
##  1st Qu.: 586.0   1st Qu.:2983   1st Qu.: 548   1st Qu.:236.0
##  Median : 715.5   Median :3889   Median : 682   Median :290.0
##  Mean   : 688.5   Mean   :3720   Mean   : 671   Mean   :293.1
##  3rd Qu.: 782.0   3rd Qu.:4450   3rd Qu.: 772   3rd Qu.:330.0
##  Max.   :1111.0   Max.   :6280   Max.   :1160   Max.   :568.0
##
##      mattr           entropy        nutts_single_word     nutts
##  Min.   :0.6948   Min.   :0.01331   Min.   : 35.0     Min.   : 301.0
##  1st Qu.:0.8727   1st Qu.:0.01622   1st Qu.: 68.0     1st Qu.: 538.0
##  Median :0.8975   Median :0.01776   Median : 91.0     Median : 722.5
##  Mean   :0.8869   Mean   :0.01784   Mean   :102.8     Mean   : 700.3
##  3rd Qu.:0.9131   3rd Qu.:0.01925   3rd Qu.:120.0     3rd Qu.: 842.0
##  Max.   :0.9336   Max.   :0.02347   Max.   :378.0     Max.   :1214.0
##
##  child
##  eth:336
##  lil:192
##  nai:480
##  vio:128
##  wil: 48
##
##
```

```
dim(res)
```

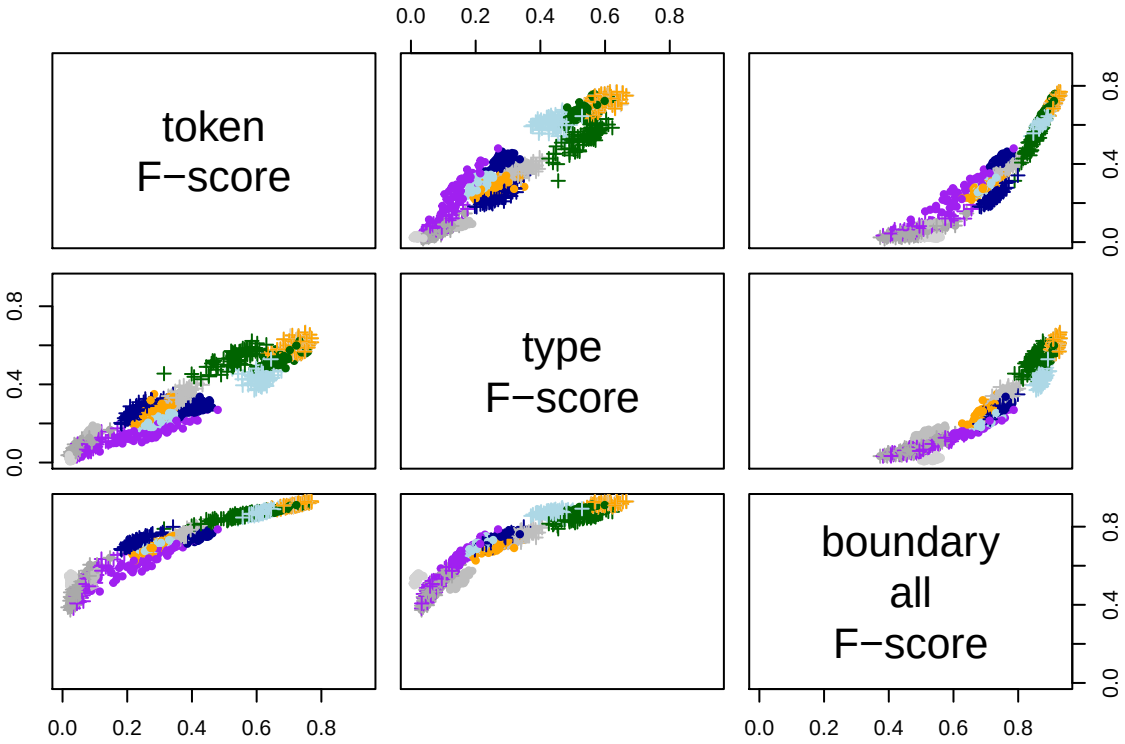
```
## [1] 1184   29
```

Correlations across token, type, and boundary F-scores

The inscriptions in the diagonal refer to the boundary axes (e.g., the second cell in the top row shows point as a function of type in the x-axis and token in the y-axis, whereas the inverse is true for the first cell in the middle row). Each point is the performance of a segmentation experiment on one of the 74 transcripts, using either phones (circles) or syllables (crosses) in combination with one of the 8 algorithms (color - please see Figure 4 for algorithm-color mapping).

Checking for whether different outcomes are independent.

```
subselcol=c("token_fscore","type_fscore","boundary_all_fscore")
sublabs=gsub("_","\n",subselcol)
sublabs=gsub("fs","F-s",sublabs)
pchs=c(3,15:18,0,7)


pairs(res[,subselcol],col =  mycols[as.numeric(as.factor(res$algo))],xlim=range(res[,subselcol]),ylim=ra
```




```
cor(res[,subselcol],method="p")
```

```
##                     token_fscore type_fscore boundary_all_fscore
## token_fscore           1.0000000   0.9638697           0.9649478
## type_fscore            0.9638697   1.0000000           0.9521331
## boundary_all_fscore    0.9649478   0.9521331           1.0000000
```

```
cor(res[,subselcol],method="s")
```

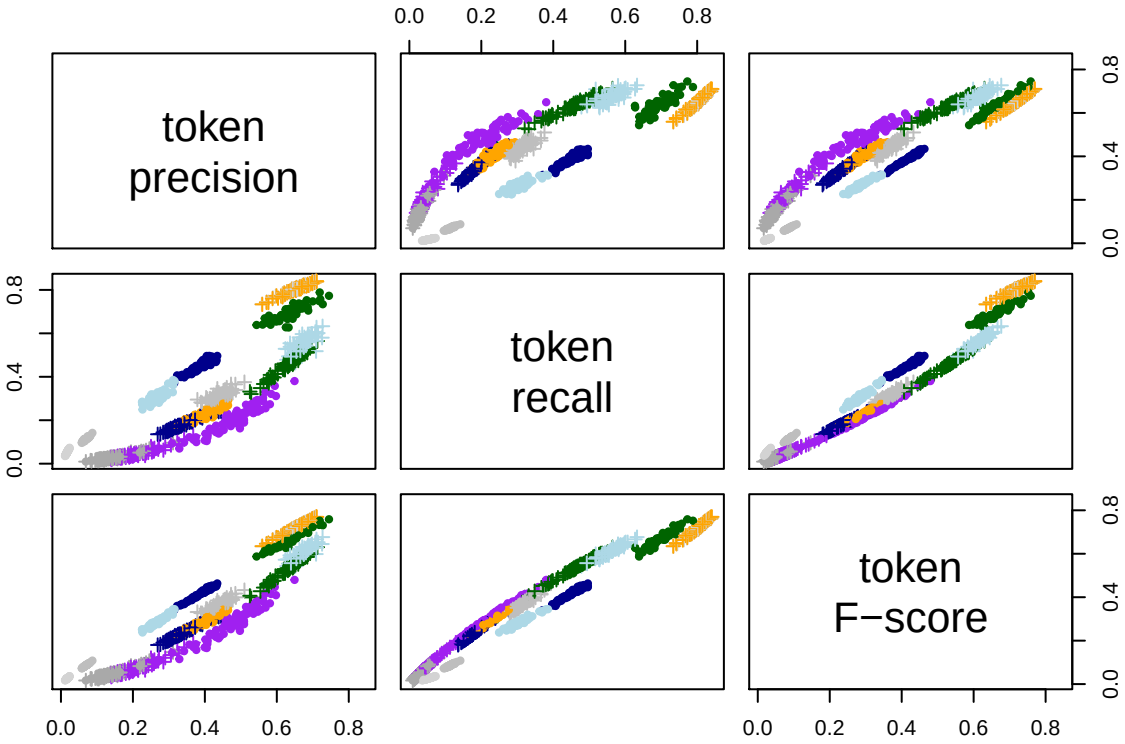
```
##                     token_fscore type_fscore boundary_all_fscore
## token_fscore           1.0000000   0.9603214           0.9601371
## type_fscore            0.9603214   1.0000000           0.9590580
## boundary_all_fscore    0.9601371   0.9590580           1.0000000
```

Deep dive: F-scores, precision, and recall

Correlations across across F-scores, precision, and recall within token, type, and boundary scores.

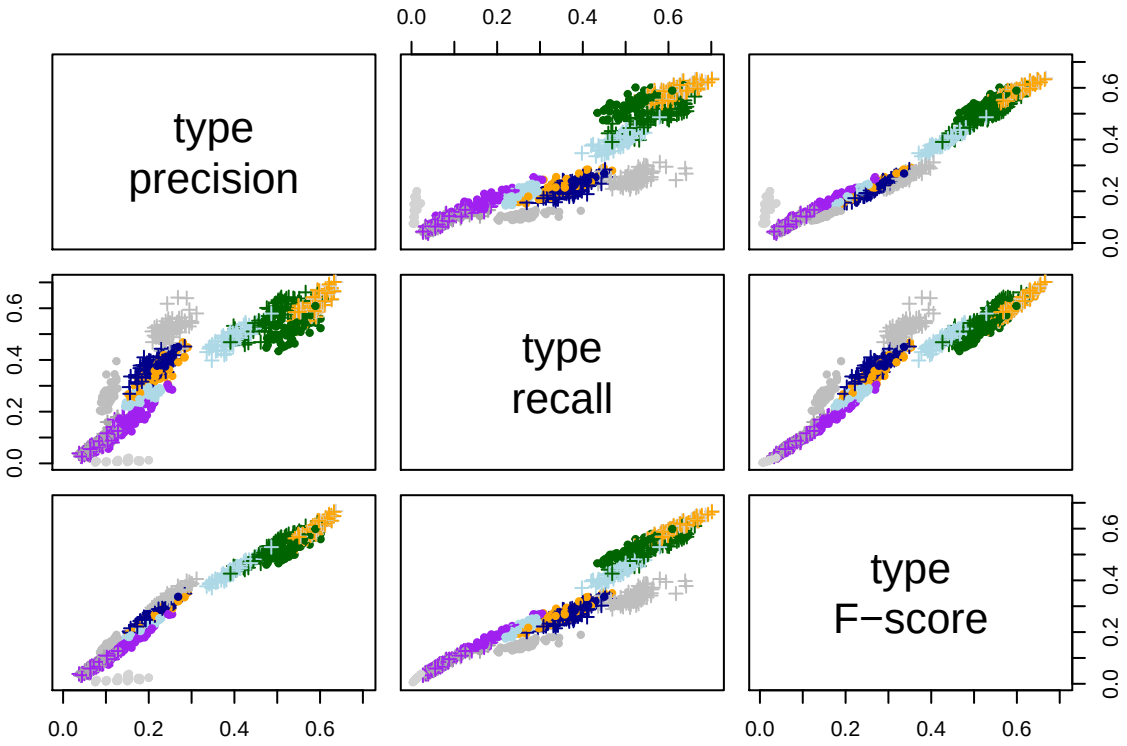
```
for(thisout in c("token_","type_","boundary_")){
  subselcol=colnames(res)[grep(thisout,colnames(res))]
  subselcol=subselcol[grep("noedge",subselcol,invert=T)]
  sublabs=gsub("_","\n",subselcol)
  sublabs=gsub("fs","F-s",sublabs)
```

```
  pairs(res[,subselcol],col =  mycols[as.numeric(as.factor(res$algo))],xlim=range(res[,subselcol]),ylim=

 cat("\n\n\n")
  print("Spearman correlations")
  print(kable(round(cor(res[,subselcol],method="s"),3)))
 cat("\n\n\\pagebreak\n")

}
```



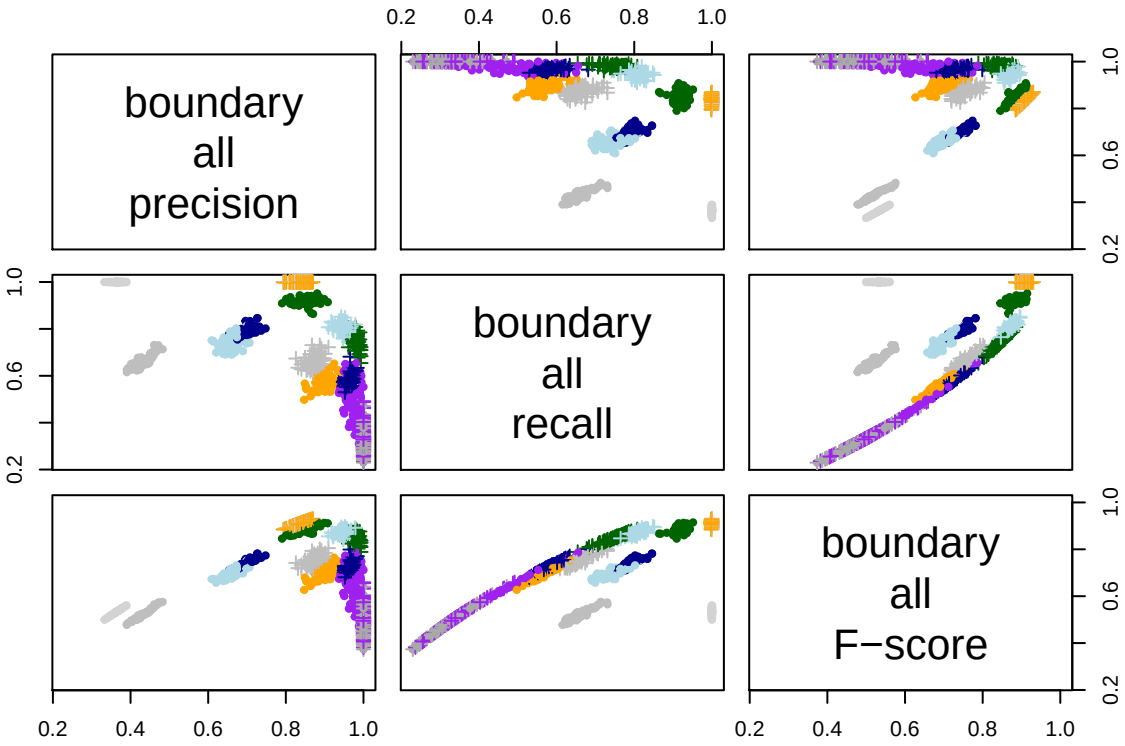
[1] "Spearman correlations"

	token_precision	token_recall	token_fscore
token_precision	1.000	0.886	0.941
token_recall	0.886	1.000	0.980
token_fscore	0.941	0.980	1.000



[1] “Spearman correlations”

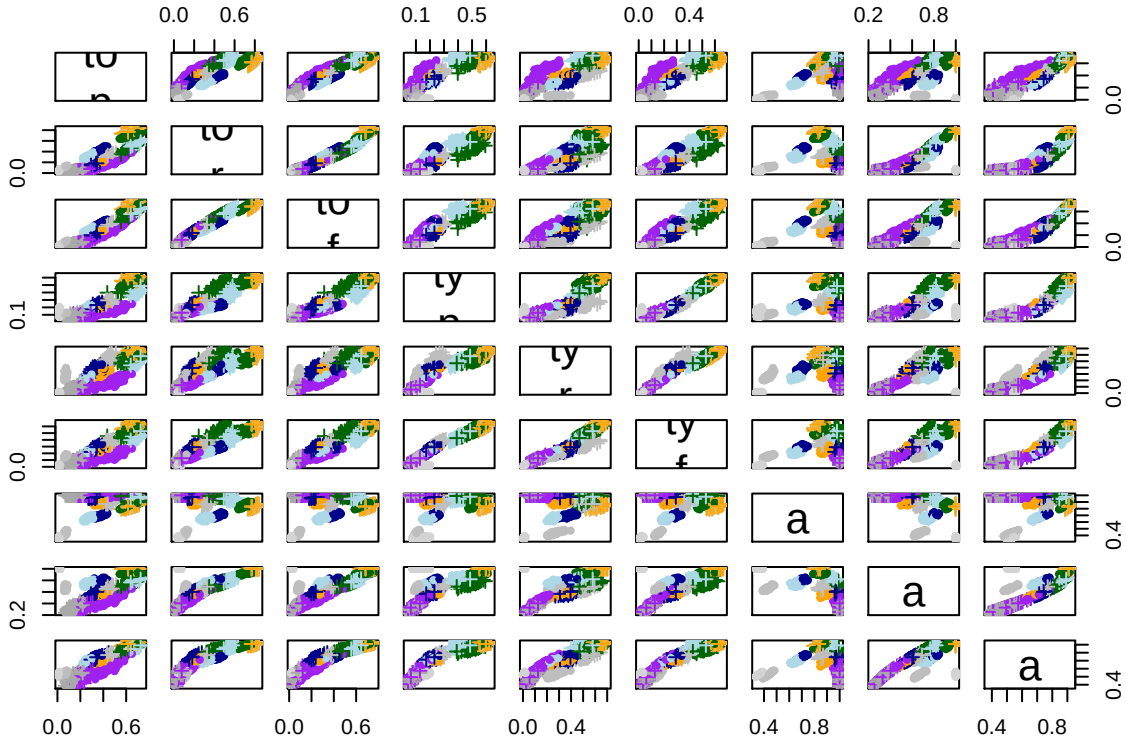
	type_precision	type_recall	type_fscore
type_precision	1.000	0.927	0.960
type_recall	0.927	1.000	0.984
type_fscore	0.960	0.984	1.000



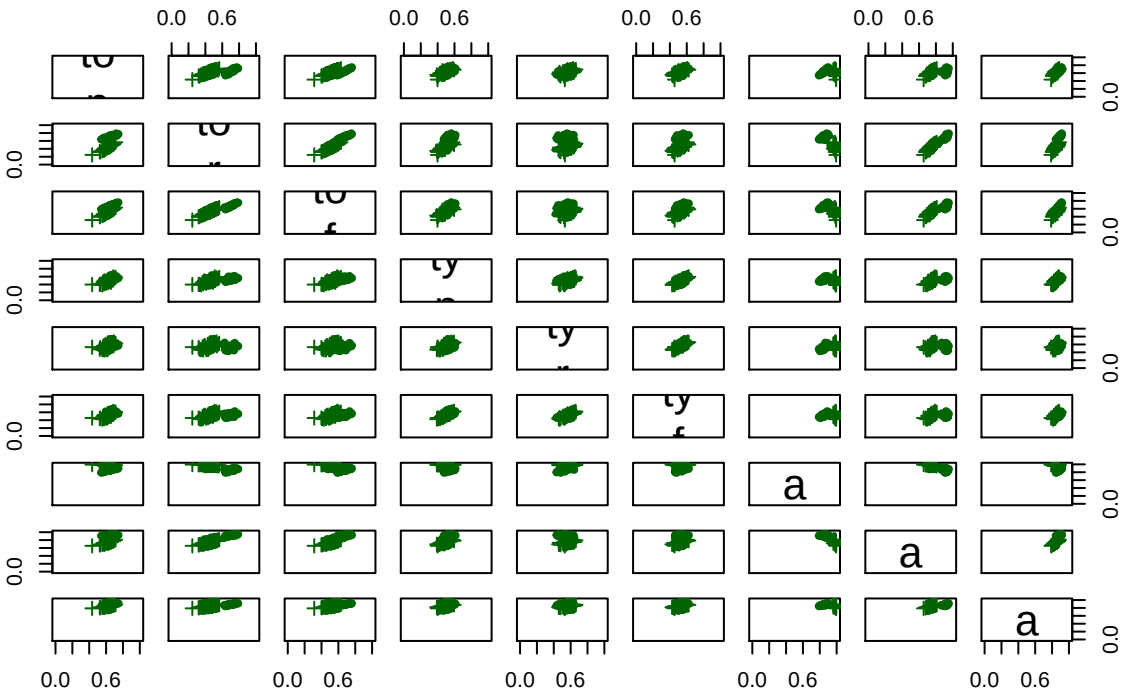
[1] “Spearman correlations”

	boundary_all_precision	boundary_all_recall	boundary_all_fscore
boundary_all_precision	1.000	-0.711	-0.235
boundary_all_recall	-0.711	1.000	0.715
boundary_all_fscore	-0.235	0.715	1.000

```
temp<-res

colnames(temp)<-gsub("token","to",colnames(temp))
colnames(temp)<-gsub("type","ty",colnames(temp))
colnames(temp)<-gsub("boundary","b",colnames(temp))
colnames(temp)<-gsub("all","a",colnames(temp))
colnames(temp)<-gsub("noedge","ne",colnames(temp))
colnames(temp)<-gsub("precision","p",colnames(temp))
colnames(temp)<-gsub("reca","r",colnames(temp))
colnames(temp)<-gsub("fscore","f",colnames(temp))
xx=colnames(temp)
selcol<-c(xx[grep("to_",xx)],xx[grep("ty_",xx)],xx[grep("b_a",xx)])
labs=gsub("_","\n",selcol)

pairs(temp[,selcol],col =  mycols[as.numeric(as.factor(temp$algo))], pch=ifelse(temp$unit=="phone",20,3)
```




```
for(i in 1:length(levels(temp$algo))){
  thisalgo=levels(temp$algo)[i]
  pairs(temp[res$algo==thisalgo,selcol],col = mycols[i], pch=ifelse(res$unit[temp$algo==thisalgo]=="phon
      xlim=range(temp[,selcol]),ylim=range(temp[,selcol]))
 cat("\n\n\n")
  print("Spearman correlations")
  print(kable(round(cor(temp[temp$algo==thisalgo,selcol],method="s"),3)))
 cat("\n\n\\pagebreak\n")

}
```

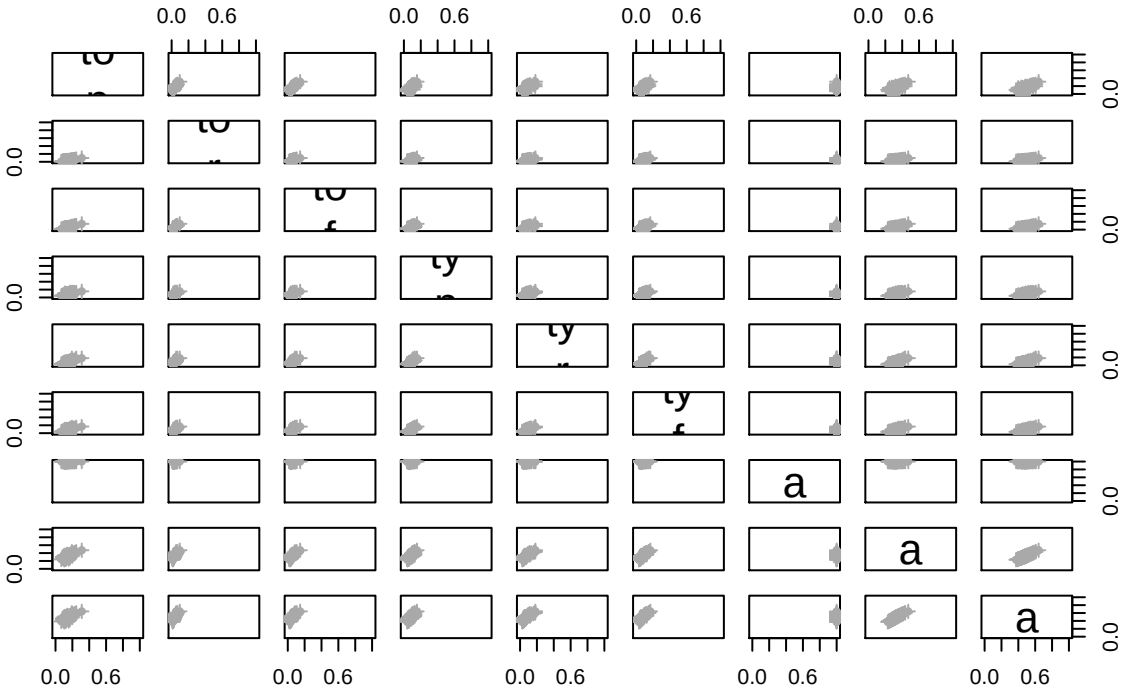


[1] “Spearman correlations”

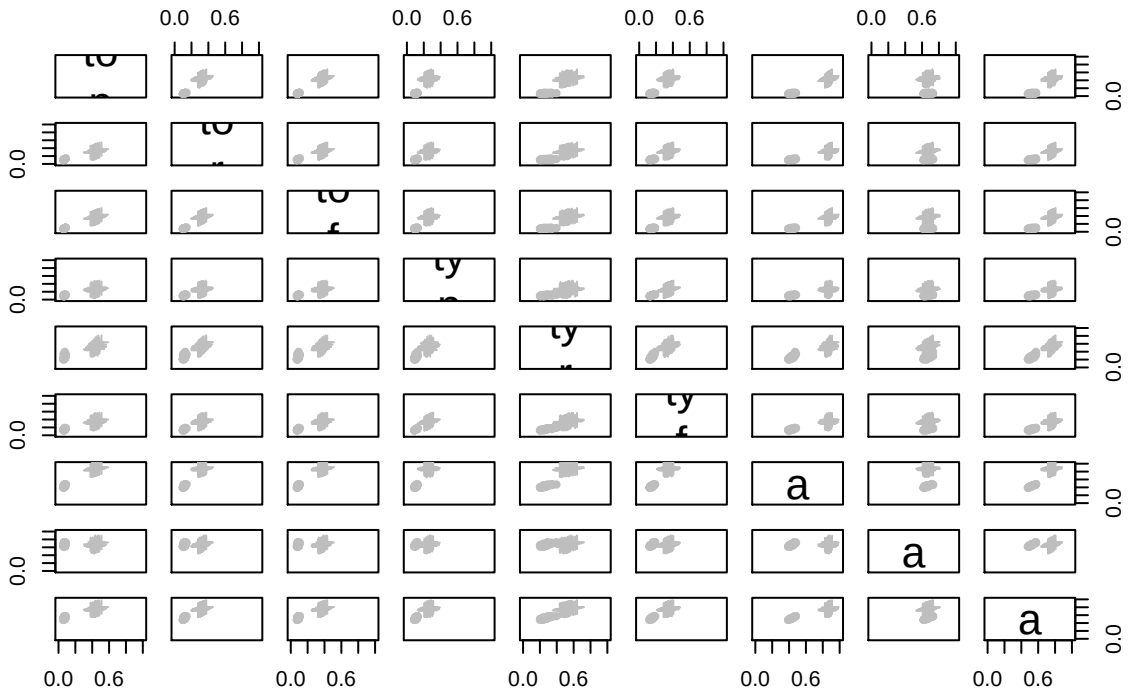
	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.506	0.546	0.686	0.501	0.722	0.064	0.370	0.601
to_r	0.506	1.000	0.993	0.713	-0.253	0.196	-0.740	0.963	0.951
to_f	0.546	0.993	1.000	0.732	-0.216	0.233	-0.707	0.937	0.961
ty_p	0.686	0.713	0.732	1.000	0.244	0.704	-0.312	0.668	0.804
ty_r	0.501	-0.253	-0.216	0.244	1.000	0.843	0.585	-0.315	-0.091
ty_f	0.722	0.196	0.233	0.704	0.843	1.000	0.244	0.129	0.357
b_a_p	0.064	-0.740	-0.707	-0.312	0.585	0.244	1.000	-0.788	-0.587
b_a_r	0.370	0.963	0.937	0.668	-0.315	0.129	-0.788	1.000	0.915
b_a_f	0.601	0.951	0.961	0.804	-0.091	0.357	-0.587	0.915	1.000

[1] “Spearman correlations”

```
## Warning in cor(temp[temp$algo == thisalgo, selcol], method = "s"): the
## standard deviation is zero
```



	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.942	0.953	0.694	0.628	0.664	NA	0.622	0.624
to_r	0.942	1.000	0.999	0.804	0.780	0.804	NA	0.824	0.825
to_f	0.953	0.999	1.000	0.793	0.766	0.791	NA	0.806	0.808
ty_p	0.694	0.804	0.793	1.000	0.937	0.970	NA	0.828	0.828
ty_r	0.628	0.780	0.766	0.937	1.000	0.991	NA	0.891	0.891
ty_f	0.664	0.804	0.791	0.970	0.991	1.000	NA	0.883	0.884
b_a_p	NA	NA	NA	NA	NA	NA	1	NA	NA
b_a_r	0.622	0.824	0.806	0.828	0.891	0.883	NA	1.000	1.000
b_a_f	0.624	0.825	0.808	0.828	0.891	0.884	NA	1.000	1.000

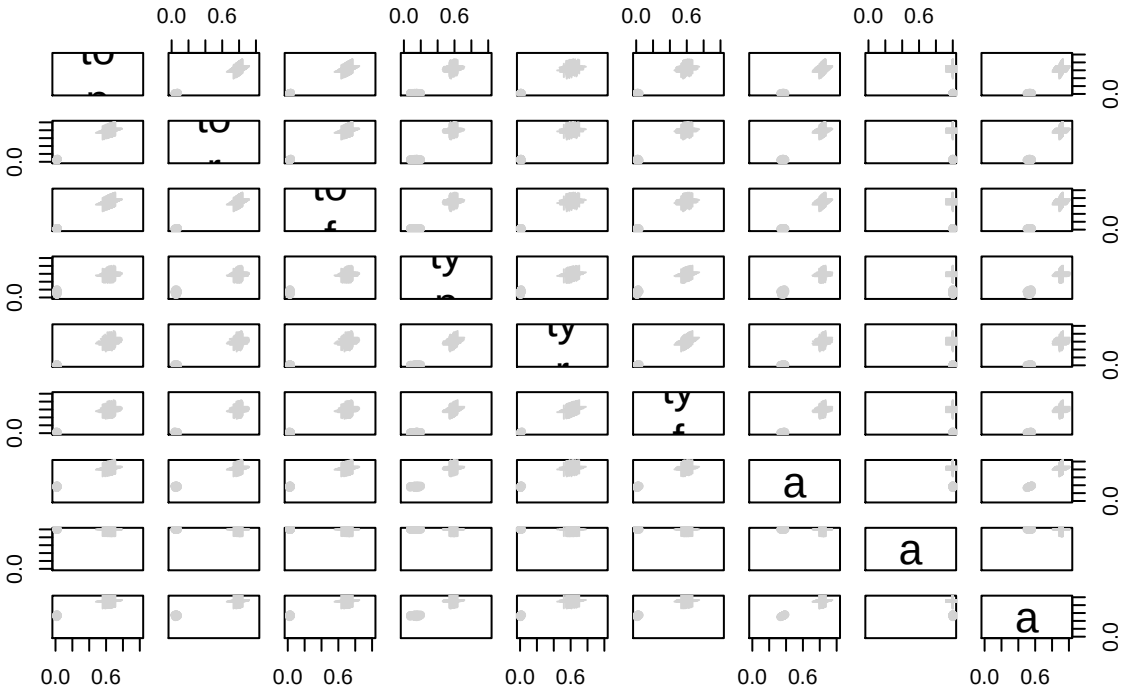


[1] “Spearman correlations”

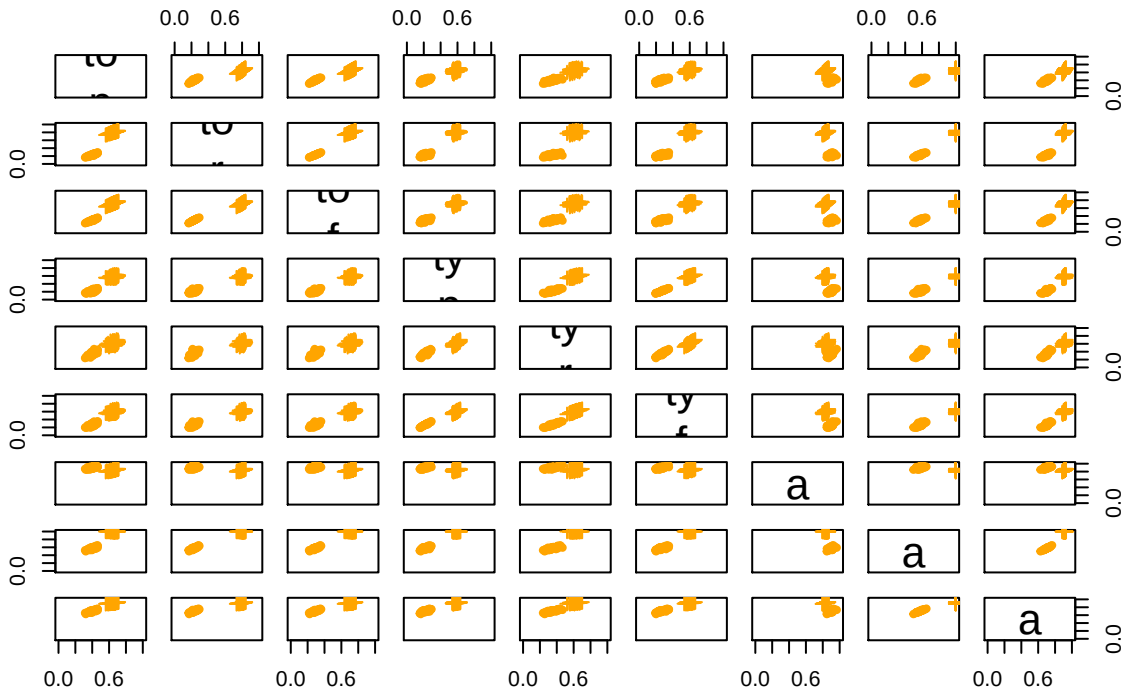
	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.976	0.992	0.878	0.895	0.897	0.934	0.200	0.904
to_r	0.976	1.000	0.994	0.905	0.906	0.918	0.908	0.314	0.935
to_f	0.992	0.994	1.000	0.894	0.907	0.912	0.921	0.262	0.924
ty_p	0.878	0.905	0.894	1.000	0.896	0.988	0.900	0.332	0.928
ty_r	0.895	0.906	0.907	0.896	1.000	0.944	0.903	0.291	0.922
ty_f	0.897	0.918	0.912	0.988	0.944	1.000	0.920	0.350	0.945
b_a_p	0.934	0.908	0.921	0.900	0.903	0.920	1.000	0.304	0.955
b_a_r	0.200	0.314	0.262	0.332	0.291	0.350	0.304	1.000	0.461
b_a_f	0.904	0.935	0.924	0.928	0.922	0.945	0.955	0.461	1.000

[1] “Spearman correlations”

```
## Warning in cor(temp[temp$algo == thisalgo, selcol], method = "s"): the
## standard deviation is zero
```

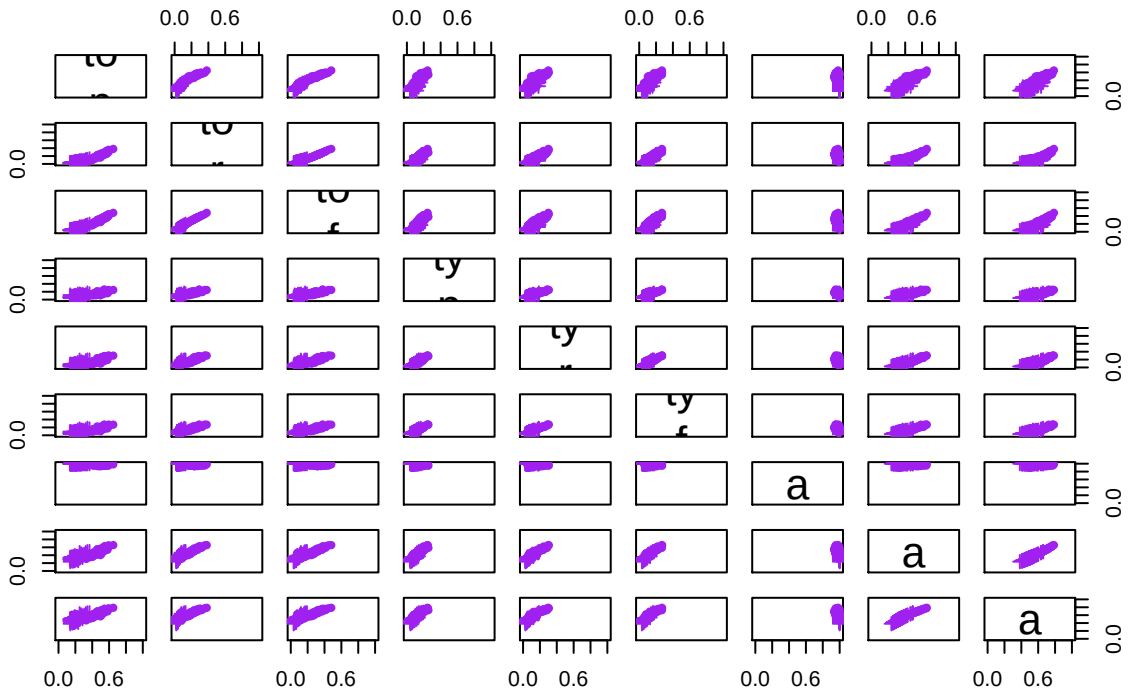


	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.996	1.000	0.809	0.805	0.806	0.876	NA	0.876
to_r	0.996	1.000	0.998	0.798	0.792	0.793	0.855	NA	0.855
to_f	1.000	0.998	1.000	0.806	0.801	0.802	0.871	NA	0.871
ty_p	0.809	0.798	0.806	1.000	0.930	0.948	0.845	NA	0.845
ty_r	0.805	0.792	0.801	0.930	1.000	0.995	0.895	NA	0.895
ty_f	0.806	0.793	0.802	0.948	0.995	1.000	0.895	NA	0.895
b_a_p	0.876	0.855	0.871	0.845	0.895	0.895	1.000	NA	1.000
b_a_r	NA	NA	NA	NA	NA	NA	NA	1	NA
b_a_f	0.876	0.855	0.871	0.845	0.895	0.895	1.000	NA	1.000



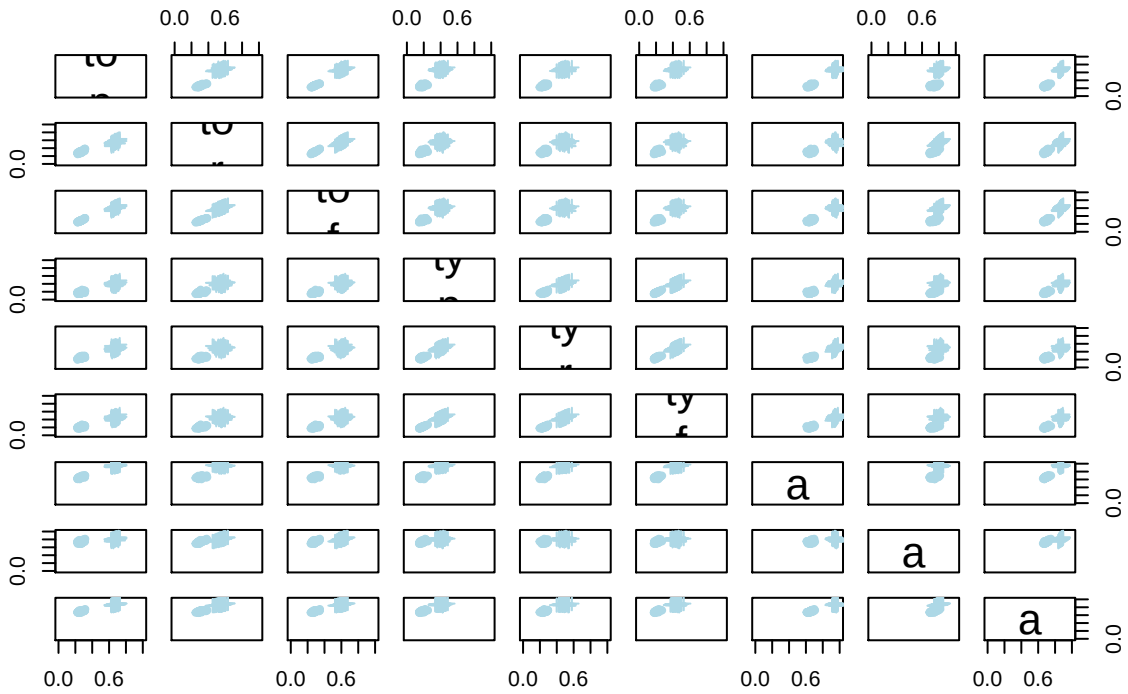
[1] “Spearman correlations”

	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.985	0.993	0.882	0.900	0.891	-0.530	0.840	0.966
to_r	0.985	1.000	0.998	0.862	0.883	0.870	-0.574	0.855	0.959
to_f	0.993	0.998	1.000	0.873	0.894	0.882	-0.558	0.851	0.966
ty_p	0.882	0.862	0.873	1.000	0.953	0.989	-0.566	0.841	0.913
ty_r	0.900	0.883	0.894	0.953	1.000	0.983	-0.562	0.830	0.933
ty_f	0.891	0.870	0.882	0.989	0.983	1.000	-0.559	0.834	0.925
b_a_p	-0.530	-0.574	-0.558	-0.566	-0.562	-0.559	1.000	-0.677	-0.512
b_a_r	0.840	0.855	0.851	0.841	0.830	0.834	-0.677	1.000	0.870
b_a_f	0.966	0.959	0.966	0.913	0.933	0.925	-0.512	0.870	1.000



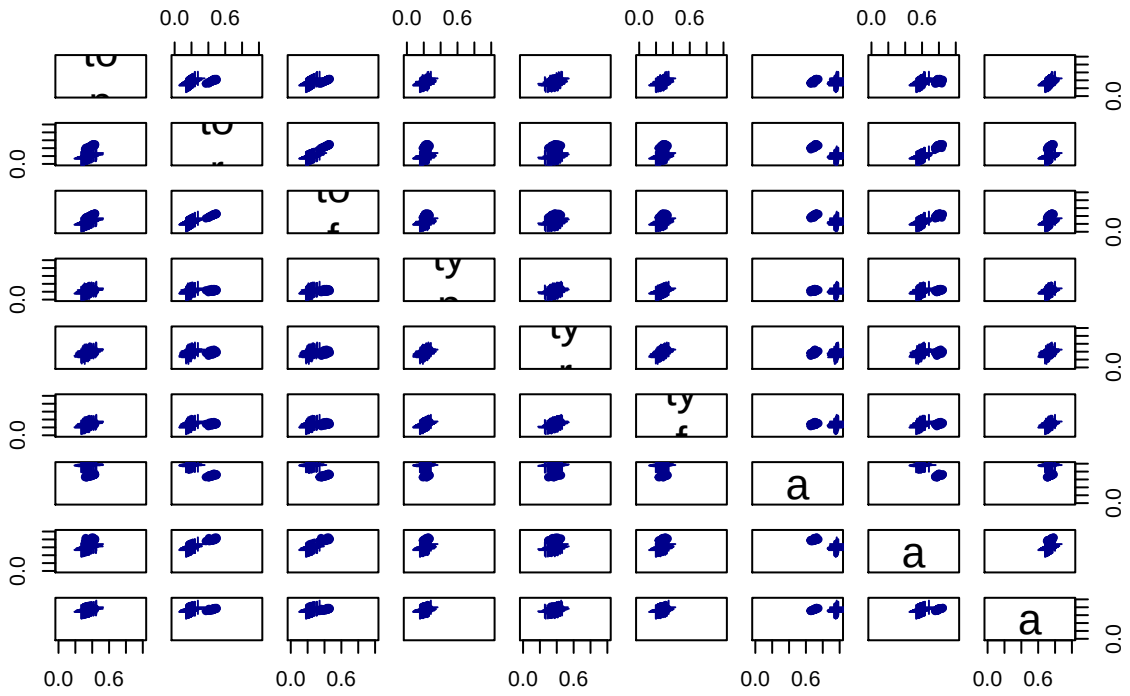
[1] “Spearman correlations”

	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.985	0.988	0.887	0.876	0.885	-0.802	0.899	0.897
to_r	0.985	1.000	1.000	0.926	0.927	0.932	-0.822	0.951	0.948
to_f	0.988	1.000	1.000	0.924	0.923	0.929	-0.819	0.947	0.945
ty_p	0.887	0.926	0.924	1.000	0.971	0.988	-0.758	0.952	0.952
ty_r	0.876	0.927	0.923	0.971	1.000	0.995	-0.729	0.975	0.976
ty_f	0.885	0.932	0.929	0.988	0.995	1.000	-0.744	0.973	0.974
b_a_p	-0.802	-0.822	-0.819	-0.758	-0.729	-0.744	1.000	-0.752	-0.740
b_a_r	0.899	0.951	0.947	0.952	0.975	0.973	-0.752	1.000	1.000
b_a_f	0.897	0.948	0.945	0.952	0.976	0.974	-0.740	1.000	1.000



[1] “Spearman correlations”

	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.945	0.980	0.829	0.860	0.849	0.884	0.847	0.936
to_r	0.945	1.000	0.988	0.820	0.800	0.817	0.780	0.948	0.949
to_f	0.980	0.988	1.000	0.824	0.830	0.833	0.828	0.912	0.955
ty_p	0.829	0.820	0.824	1.000	0.936	0.989	0.870	0.824	0.887
ty_r	0.860	0.800	0.830	0.936	1.000	0.974	0.894	0.743	0.857
ty_f	0.849	0.817	0.833	0.989	0.974	1.000	0.892	0.795	0.883
b_a_p	0.884	0.780	0.828	0.870	0.894	0.892	1.000	0.707	0.870
b_a_r	0.847	0.948	0.912	0.824	0.743	0.795	0.707	1.000	0.942
b_a_f	0.936	0.949	0.955	0.887	0.857	0.883	0.870	0.942	1.000



[1] “Spearman correlations”

	to_p	to_r	to_f	ty_p	ty_r	ty_f	b_a_p	b_a_r	b_a_f
to_p	1.000	0.900	0.912	0.664	0.487	0.654	-0.305	0.799	0.880
to_r	0.900	1.000	0.996	0.601	0.318	0.555	-0.612	0.947	0.789
to_f	0.912	0.996	1.000	0.597	0.326	0.556	-0.599	0.931	0.788
ty_p	0.664	0.601	0.597	1.000	0.699	0.974	-0.126	0.644	0.759
ty_r	0.487	0.318	0.326	0.699	1.000	0.829	0.238	0.323	0.624
ty_f	0.654	0.555	0.556	0.974	0.829	1.000	-0.025	0.588	0.766
b_a_p	-0.305	-0.612	-0.599	-0.126	0.238	-0.025	1.000	-0.629	-0.122
b_a_r	0.799	0.947	0.931	0.644	0.323	0.588	-0.629	1.000	0.812
b_a_f	0.880	0.789	0.788	0.759	0.624	0.766	-0.122	0.812	1.000

Everything is correlated. So using token F-score only as outcome for all the following analyses.

Corpus size effects

Corpus size can be measured in number of phones, words, utterances. To decide which of the three we'll use in further analyses, we inspect the proportion of variance explained by each.

```
for(thispred in c("nphone_tokens","nsyllable_tokens","nwords_tokens","nutts")){
  myfit=lm(res$token_fscore~res$algo*res$unit*res[,thispred] + (1/res$clean_name))
  print(thispred)
  print(summary(myfit)$adj.r.squared)
}
```

```
## [1] "nphone_tokens"
## [1] 0.9825155
## [1] "nsyllable_tokens"
## [1] 0.9825153
## [1] "nwords_tokens"
## [1] 0.9825074
## [1] "nutts"
## [1] 0.9845529
```

The best is number of utterances. In the following analyses, we investigate the association between performance and corpus length for all algorithms.

```
tab=NULL
for(thisalgo in 1:length(levels(res$algo))) {
  phonelm=summary(lm(token_fscore~nutts + (1/clean_name),data=res,subset=c(res$algo==levels(res$algo)[t
  syllm=summary(lm(token_fscore~nutts + (1/clean_name),data=res,subset=c(res$algo==levels(res$algo)[this
    print(phonelm)
    print(syllm)
    tab=rbind(tab,rbind(
      cbind(levels(res$algo)[thisalgo],"phone",phonelm$adj.r.squared,phonelm$coefficients[2],phonelm$co
      cbind(levels(res$algo)[thisalgo],"syll",syllm$adj.r.squared,syllm$coefficients[2],syllm$coefficie
      ))
}
```

```
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.093483 -0.019655 -0.002183  0.022129  0.088475
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.913e-01  1.451e-02  47.637   <2e-16 ***
## nutts       -3.496e-05  1.992e-05  -1.755   0.0836 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03433 on 72 degrees of freedom
```

```
## Multiple R-squared:  0.04101,    Adjusted R-squared:  0.02769
## F-statistic: 3.079 on 1 and 72 DF,  p-value: 0.08357
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.20781 -0.01792  0.01050  0.03071  0.07389
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.516e-01  2.113e-02  21.367  < 2e-16 ***
## nutts       1.098e-04  2.902e-05   3.784 0.000316 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04999 on 72 degrees of freedom
## Multiple R-squared:  0.1659, Adjusted R-squared:  0.1543
## F-statistic: 14.32 on 1 and 72 DF,  p-value: 0.0003163
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.029525 -0.014699 -0.005416  0.006913  0.081860
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.386e-02  9.139e-03   2.611  0.01098 *
## nutts       3.384e-05  1.255e-05   2.697  0.00872 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02162 on 72 degrees of freedom
## Multiple R-squared:  0.09173,    Adjusted R-squared:  0.07912
## F-statistic: 7.272 on 1 and 72 DF,  p-value: 0.008716
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.029525 -0.014699 -0.005416  0.006913  0.081860
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.386e-02  9.139e-03   2.611  0.01098 *
## nutts       3.384e-05  1.255e-05   2.697  0.00872 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02162 on 72 degrees of freedom
## Multiple R-squared:  0.09173,    Adjusted R-squared:  0.07912
## F-statistic: 7.272 on 1 and 72 DF,  p-value: 0.008716
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0146152 -0.0038991  0.0002055  0.0045024  0.0222451
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.034e-02  2.805e-03  32.210   <2e-16 ***
## nutts       -4.677e-06  3.851e-06  -1.215    0.228
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.006635 on 72 degrees of freedom
## Multiple R-squared:  0.02008,    Adjusted R-squared:  0.006471
## F-statistic: 1.475 on 1 and 72 DF,  p-value: 0.2285
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.04031 -0.01625 -0.00265  0.01420  0.06717
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.687e-01  9.419e-03  39.142   <2e-16 ***
## nutts       -3.937e-06  1.293e-05  -0.304    0.762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02228 on 72 degrees of freedom
## Multiple R-squared:  0.001285,   Adjusted R-squared:  -0.01259
## F-statistic: 0.09265 on 1 and 72 DF,  p-value: 0.7617
##
```

```
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0096174 -0.0031686 -0.0004815  0.0036266  0.0089051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.898e-02  1.875e-03  15.457   <2e-16 ***
## nutts       -3.389e-06  2.574e-06  -1.317    0.192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.004435 on 72 degrees of freedom
## Multiple R-squared:  0.02352,    Adjusted R-squared:  0.009954
## F-statistic: 1.734 on 1 and 72 DF,  p-value: 0.1921
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.082630 -0.017568  0.001671  0.020347  0.044088
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.497e-01  1.183e-02  63.377   <2e-16 ***
## nutts       -4.054e-05  1.624e-05  -2.496   0.0149 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02798 on 72 degrees of freedom
## Multiple R-squared:  0.07964,    Adjusted R-squared:  0.06685
## F-statistic:  6.23 on 1 and 72 DF,  p-value: 0.01485
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059908 -0.016353 -0.003011  0.019939  0.069713
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 2.471e-01  1.179e-02  20.961  < 2e-16 ***
## nutts       6.316e-05  1.619e-05   3.901 0.000213 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02789 on 72 degrees of freedom
## Multiple R-squared:  0.1745, Adjusted R-squared:  0.163
## F-statistic: 15.22 on 1 and 72 DF,  p-value: 0.0002128
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.082506 -0.017528  0.001122  0.021415  0.043608
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.502e-01  1.187e-02  63.177   <2e-16 ***
## nutts       -4.051e-05  1.630e-05  -2.485   0.0153 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02809 on 72 degrees of freedom
## Multiple R-squared:  0.07897,    Adjusted R-squared:  0.06617
## F-statistic: 6.173 on 1 and 72 DF,  p-value: 0.0153
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.111561 -0.041755 -0.008378  0.036409  0.192382
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.601e-02  2.516e-02   2.624   0.0106 *
## nutts       3.023e-04  3.454e-05   8.751 6.07e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05951 on 72 degrees of freedom
## Multiple R-squared:  0.5154, Adjusted R-squared:  0.5087
## F-statistic: 76.59 on 1 and 72 DF,  p-value: 6.065e-13
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
```

```
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.042370 -0.018230 -0.007365  0.009266  0.109527
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.231e-02  1.233e-02   2.621  0.01069 *
## nutts       5.911e-05  1.692e-05   3.493  0.00082 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02916 on 72 degrees of freedom
## Multiple R-squared:  0.1449, Adjusted R-squared:  0.133
## F-statistic:  12.2 on 1 and 72 DF,  p-value: 0.0008201
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.045346 -0.014244 -0.002835  0.010818  0.062528
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.845e-01  9.894e-03  28.759   <2e-16 ***
## nutts       -2.322e-06  1.358e-05  -0.171    0.865
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02341 on 72 degrees of freedom
## Multiple R-squared:  0.0004058,  Adjusted R-squared:  -0.01348
## F-statistic: 0.02923 on 1 and 72 DF,  p-value: 0.8647
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.085026 -0.014191  0.003447  0.018734  0.061488
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.990e-01  1.204e-02  49.761   <2e-16 ***
## nutts       1.705e-05  1.653e-05   1.031    0.306
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02847 on 72 degrees of freedom
## Multiple R-squared:  0.01456,    Adjusted R-squared:  0.0008726
## F-statistic: 1.064 on 1 and 72 DF,  p-value: 0.3058
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit ==
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.080338 -0.017236  0.002418  0.019056  0.050435
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.370e-01  1.153e-02  37.897   <2e-16 ***
## nutts       -3.017e-05  1.583e-05  -1.906   0.0607 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02728 on 72 degrees of freedom
## Multiple R-squared:  0.04801,    Adjusted R-squared:  0.03479
## F-statistic: 3.631 on 1 and 72 DF,  p-value: 0.0607
##
##
## Call:
## lm(formula = token_fscore ~ nutts + (1/clean_name), data = res,
##     subset = c(res$algo == levels(res$algo)[thisalgo] & res$unit !=
##         "phone"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.056256 -0.016871 -0.006254  0.014135  0.071996
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.841e-01  1.186e-02  15.522  < 2e-16 ***
## nutts       7.044e-05  1.628e-05   4.326 4.81e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02806 on 72 degrees of freedom
## Multiple R-squared:  0.2063, Adjusted R-squared:  0.1953
## F-statistic: 18.71 on 1 and 72 DF,  p-value: 4.808e-05
```

```
colnames(tab)<-c("Algo","Unit","R2","B","p")
tab[,3]=round(as.numeric(as.character(tab[,3]))*100,1)
tab[,4]=round(as.numeric(as.character(tab[,4]))*10^4,3)
tab[,5]=round(as.numeric(as.character(tab[,5])),3)
tab[,1]=gsub("_b","",tab[,1])
tab[,5]<-ifelse(tab[,5]<0.05,paste0(tab[,5],"*"),tab[,5])
```

```
tab[tab[,5]=="0*",5]<-"<.001*"
write.table(tab,"derived/lng-ind-tab.txt",eol="\\\\\n",sep=" & ",quote=F,row.names = F)
```

Next we repeat the analyses, but this time on the concatenated transcripts.

```
read.table("derived/length_all.txt",header=T)->lng
lng$iteration=as.numeric(as.character(gsub(".*_","",lng$clean_name)))


ind=NULL
for(thisc in levels(lng$child)){
  subset(lng, child==thisc)-> xx
  ind=rbind(ind,subset(xx, iteration==max(xx$iteration)))
}

minfit=lm(token_fscore~algo*unit ,data=ind) #
summary(minfit)
```

```
##
## Call:
## lm(formula = token_fscore ~ algo * unit, data = ind)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.066920 -0.010670 -0.000487  0.010020  0.061960
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.70632    0.01019  69.339  < 2e-16 ***
## algobaseline-00               -0.65368    0.01441 -45.376  < 2e-16 ***
## algobaseline-05               -0.61812    0.01441 -42.908  < 2e-16 ***
## algobaseline-10               -0.68077    0.01441 -47.256  < 2e-16 ***
## algodibs                      -0.37126    0.01441 -25.771  < 2e-16 ***
## algopuddle                    -0.07848    0.01441  -5.448 8.69e-07 ***
## algotpabs                     -0.44610    0.01441 -30.967  < 2e-16 ***
## algotprel                     -0.27940    0.01441 -19.395  < 2e-16 ***
## unitsyllable                  -0.10860    0.01441  -7.539 2.10e-10 ***
## algobaseline-00:unitsyllable   0.10860    0.02037   5.331 1.36e-06 ***
## algobaseline-05:unitsyllable   0.38896    0.02037  19.092  < 2e-16 ***
## algobaseline-10:unitsyllable   0.80685    0.02037  39.604  < 2e-16 ***
## algodibs:unitsyllable          0.49822    0.02037  24.455  < 2e-16 ***
## algopuddle:unitsyllable       -0.35250    0.02037 -17.302  < 2e-16 ***
## algotpabs:unitsyllable         0.48506    0.02037  23.809  < 2e-16 ***
## algotprel:unitsyllable        -0.05104    0.02037  -2.505   0.0148 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02278 on 64 degrees of freedom
## Multiple R-squared:  0.9935, Adjusted R-squared:  0.992
## F-statistic: 655.4 on 15 and 64 DF,  p-value: < 2.2e-16
```

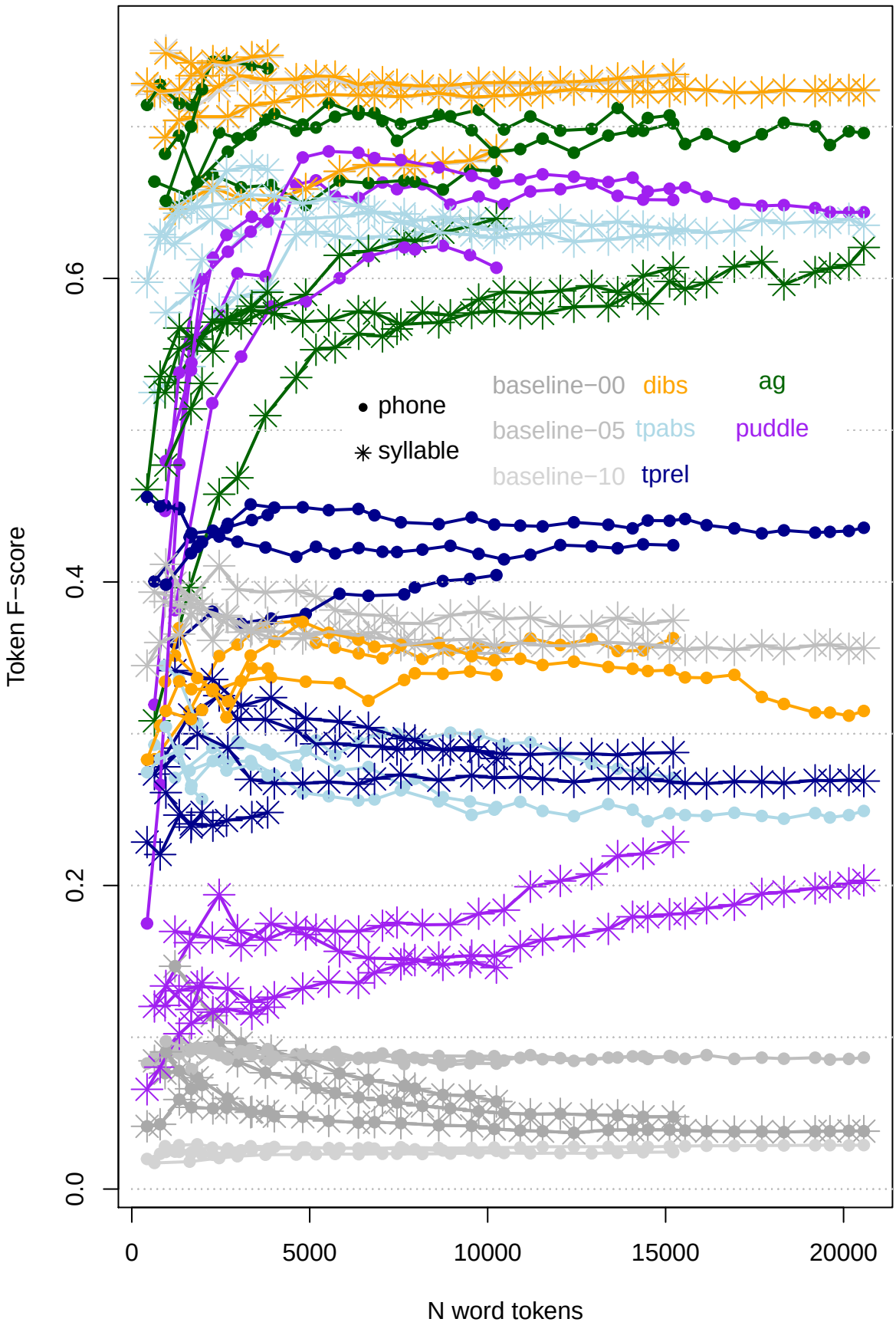
```
for(thispred in c("nwords_tokens","nphone_tokens","nutts")){
  myfit=lm(token_fscore~algo*unit*ind[,thispred] ,data=ind)
  print(thispred)
  print(summary(myfit)$adj.r.squared)
```

```
}
```

```
## [1] "nwords_tokens"
## [1] 0.9933158
## [1] "nphone_tokens"
## [1] 0.9933648
## [1] "nutts"
## [1] 0.9936874
```

Corpus size effects with all children

```
plot(token_fscore~nutts,data=lng,type="n",xlab="N word tokens",ylab="Token F-score")

for (thischild in levels(lng$child)){
  subset(lng,child==thischild)->nai


  for(thisscore in 0:10) lines(c(0,10^6),c(thisscore/10,thisscore/10),lty=3,col="gray")
  for(thisalgo in 1:length(levels(nai$algo))) {
    points(token_fscore~nutts,data=nai,subset=c(algo==levels(nai$algo)[thisalgo]),
           pch=ifelse(nai$unit[nai$algo==levels(nai$algo)[thisalgo]]=="phone",20,8),
           col=mycols[thisalgo],cex=ifelse(nai$unit[nai$algo==levels(nai$algo)[thisalgo]]=="phone",1.5,1
  }

  nai$kind=paste(nai$thisconc,nai$unit,nai$algo)
  for(i in 2:max(nai$iteration) ){
    for(thiskind in levels(as.factor(nai$kind))){
      origin=nai[nai$kind==thiskind & nai$iteration==(i-1),c("nutts","token_fscore")]
      end=nai[nai$kind==thiskind & nai$iteration==(i),c("nutts","token_fscore")]
      lines(c(origin$nutts,end$nutts),c(origin$token_fscore,end$token_fscore),lwd=2,
            lty=1, #ifelse(strsplit(thiskind," ")[[1]][2]=="phone",1,3), #this line is not working
            col=mycols[names(mycols)==strsplit(thiskind," ")[[1]][3]])
    }
  }

  #add legend
  top=.53
  mid=.50
  bottom=.47

  polygon(c(20000,6000,6000,20000),c(top+.01,top+.01,bottom-.01,bottom-.01),border="white",bg="white",c

  points(6500,mean(c(top,mid)),pch=20)
  text(6510,mean(c(top,mid)),"phone",pos=4)
  points(6500,mean(c(bottom,mid)),pch=8)
  text(6510,mean(c(bottom,mid)),"syllable",pos=4)

  text(c(18000,12000,12000,12000,15000,18000,   15000,15000),
       c(top     ,top,mid,bottom,     top,  mid,    mid,bottom),names(mycols),col=mycols)
}
```


Token F−score
0.0
0.2
0.4
0.6
0
5000
10000
15000
20000
N word tokens
phone
syllable
baseline−00
baseline−05
baseline−10
dibs
tpabs
tprel
ag
puddle