Data analysis of the database : NDC, CCDB and IFADV

In [ ]:
```
import copy
import matplotlib.pyplot as plt
import warnings
import pandas as pd
from IPython.display import display, Markdown
pd.set_option('display.max_rows', None)

from snl_stats_extraction_data import *
from snl_stats_visualization_database import *
DIR, databases_pair_paths, databases_paths, tier_lists, databases, databases_pai
```

Parameters

In [ ]:
```
databases_name = [key.replace('_paths','').upper() for key in databases.keys()]
databases_pairs = [key for key in databases_pairs.keys()]
expressions = ["Smiles_0", "Laughs_0"]
# entities = {expression : tier_lists[expression] for expression in expressions}
laughs_intensities = tier_lists['Laughs_0']
smiles_intensities = tier_lists['Smiles_0']
```

1-Database informations

In this section, we will perform an analysis of the distribution of smiles and laughs in the dataset to understand their frequency and patterns of occurrence. We will also examine the intensity levels of smiles and laughs in the different datasets and analyze whether there are differences in these levels depending on the role of the participants.

To visualize the distribution of smiles and laughs, we'll generate graphs and charts that highlight the relative frequencies of these facial expressions in each dataset. This will allow us to determine which expressions are most common and how they are distributed between the different roles of the participants, such as speaker and listener.

Looking at the intensity levels of smiles and laughs, we will analyze the data to see if there are significant variations in expression levels depending on the role of the participant. For example, we could compare the intensity levels of smiles and laughs produced by speakers and listeners to determine if there are systematic differences between the two roles.

General informations:

In [ ]:
```
dfs = []
dataset_names = []

for i in databases.keys():
    databases_list = databases_paths[i]
```

```
    data = display_general_informations_files(databases_list)
    columns_names = ["Filename", "Duration"] + list(tier_lists.keys())
    df = pd.DataFrame(data, columns=columns_names)
    zero_columns = df.columns[(df == 0).all()]
    df = df.drop(zero_columns, axis=1)
    dfs.append(df)
    dataset_names.append(i)

for i in range(len(dfs)):
    display(Markdown(f"#### {dataset_names[i].replace('_paths','').upper()} infc
    display(dfs[i])
```

CCDB informations:

	Filename	Duration	S&L_0	Laughs_0	Smiles_0	Role
0	P12_P2_1402_dizzy.eaf	120.170	44	1	43	36
1	P12_P2_1402_monk.eaf	124.840	55	4	51	34
2	P12_P3_2202_dizzy.eaf	60.120	28	0	28	5
3	P12_P3_2202_monk.eaf	122.028	40	7	33	22
4	P14_P7_2502_dizzy.eaf	121.227	36	10	26	10
5	P14_P7_2502_monk.eaf	66.140	20	1	19	8
6	P15_P13_2402_dizzy.eaf	120.025	73	10	63	23
7	P15_P13_2402_monk.eaf	60.425	21	2	19	19
8	P16_P6_0903_dizzy.eaf	60.990	19	0	19	9
9	P16_P6_0903_monk.eaf	61.015	21	0	21	4
10	P18_P10_2102_dizzy.eaf	60.035	37	0	37	11
11	P18_P10_2102_monk.eaf	61.230	24	1	23	14
12	P18_P1_2102_dizzy.eaf	60.900	42	0	42	10
13	P18_P1_2102_monk.eaf	60.030	31	1	30	10

IFADV informations:

	Filename	Duration	S&L_0	Laughs_0	Smiles_0	Role
0	DVA1A.eaf	120.020	25	0	25	24
1	DVA2C.eaf	120.990	46	1	45	31
2	DVA3E.eaf	60.010	11	4	7	8
3	DVA4C.eaf	60.190	27	2	25	15
4	DVA5G.eaf	63.780	9	0	9	10
5	DVA6H.eaf	61.415	11	0	11	15
6	DVA7B.eaf	60.010	32	4	28	19
7	DVA8K.eaf	60.020	15	5	10	7
8	DVB1B.eaf	120.020	36	2	34	17
9	DVB2D.eaf	120.025	45	3	43	33
10	DVB3F.eaf	119.980	51	4	47	14
11	DVB4G.eaf	122.915	53	3	50	48
12	DVB5H.eaf	123.010	21	1	20	38
13	DVB6I.eaf	119.990	27	2	25	38
14	DVB7J.eaf	119.995	50	0	50	36
15	DVB8L.eaf	120.040	55	5	50	28

NDC informations:

	Filename	Duration	S&L_0	Laughs_0	Smiles_0	Role
0	13_1_A_M.eaf	97.932	19	3	16	15
1	13_1_B_F.eaf	98.141	39	11	30	10
2	13_2_A_M.eaf	102.156	24	1	23	16
3	13_2_B_F.eaf	101.708	71	2	69	12
4	13_3_B_F.eaf	165.459	63	11	52	34
5	13_4_A_M.eaf	247.236	50	7	43	34
6	13_4_B_F.eaf	247.317	135	14	121	27
7	14_1_A_M.eaf	333.970	29	4	25	14
8	14_1_B_F.eaf	334.752	50	0	50	13
9	14_2_A_M.eaf	261.912	40	2	38	36
10	14_2_B_F.eaf	262.700	106	6	100	33
11	17_1_A_F.eaf	87.617	19	4	15	6
12	17_1_B_F.eaf	87.876	16	2	14	4
13	17_2_A_F.eaf	133.591	24	2	22	7
14	17_2_B_F.eaf	133.638	18	4	14	5
15	17_3_A_F.eaf	124.608	22	5	17	10
16	17_3_B_F.eaf	125.096	27	4	23	12
17	17_4_A_F.eaf	204.074	47	3	44	6
18	17_4_B_F.eaf	204.050	58	4	54	9
19	18_1_A_M.eaf	175.543	72	7	66	34
20	18_1_B_M.eaf	175.119	75	4	71	36
21	20_2_A_M.eaf	108.450	30	1	29	4
22	20_2_B_M.eaf	107.717	9	3	6	8
23	21_1_A_M.eaf	127.720	25	4	21	8
24	21_1_B_M.eaf	127.730	30	2	29	8
25	21_2_A_M.eaf	143.116	59	10	49	11
26	8_1_A_M.eaf	189.150	64	7	57	19
27	8_1_B_M.eaf	188.731	37	8	30	23
28	8_2_A_M.eaf	219.602	35	3	32	13
29	8_2_B_M.eaf	218.955	61	10	51	10
30	8_3_A_M.eaf	269.405	72	6	66	10
31	8_3_B_M.eaf	269.481	52	14	38	9
32	8_4_A_M.eaf	178.770	54	4	50	8
33	8_4_B_M.eaf	178.275	47	14	33	11

Visualization of smile and laugh intensity levels in the dataset:

In [ ]:
```
for i in databases.keys():
    databases_list = databases_paths[i]
    for tier in expressions:
        data=display_specific_informations(databases_list, tier, tier_lists[tier
        columns_names=["Filename", "Min duration", "Max duration"]+tier_lists[ti
        df=pd.DataFrame(data, columns=columns_names)
        zero_columns = df.columns[(df == 0).all()]
        df = df.drop(zero_columns, axis=1)
        display(Markdown(f"## {i.replace('_paths','').upper()} informations for
        display(df)
```

CCDB informations for Smiles_0:

	Filename	Min duration	Max duration	low	high	medium	subtle
0	P12_P2_1402_dizzy.eaf	0.260	9.050	16	7	13	7
1	P12_P2_1402_monk.eaf	0.310	8.965	12	19	20	0
2	P12_P3_2202_dizzy.eaf	0.265	13.100	10	4	5	9
3	P12_P3_2202_monk.eaf	0.340	14.511	15	3	9	6
4	P14_P7_2502_dizzy.eaf	0.554	11.370	9	6	5	6
5	P14_P7_2502_monk.eaf	0.520	12.815	3	6	9	1
6	P15_P13_2402_dizzy.eaf	0.105	4.395	26	19	11	7
7	P15_P13_2402_monk.eaf	0.270	10.490	4	8	6	1
8	P16_P6_0903_dizzy.eaf	0.400	4.260	7	1	5	6
9	P16_P6_0903_monk.eaf	0.675	7.090	11	0	3	7
10	P18_P10_2102_dizzy.eaf	0.170	8.330	12	13	11	1
11	P18_P10_2102_monk.eaf	0.280	7.230	7	2	9	5
12	P18_P1_2102_dizzy.eaf	0.175	8.050	8	14	18	2
13	P18_P1_2102_monk.eaf	0.270	5.950	14	7	6	3

CCDB informations for Laughs_0:

	Filename	Min duration	Max duration	medium	low	high
0	P12_P2_1402_dizzy.eaf	1.315	1.315	1	0	0
1	P12_P2_1402_monk.eaf	0.440	0.635	0	4	0
2	P12_P3_2202_dizzy.eaf	inf	0.000	0	0	0
3	P12_P3_2202_monk.eaf	0.290	1.690	3	4	0
4	P14_P7_2502_dizzy.eaf	1.010	4.240	2	3	5
5	P14_P7_2502_monk.eaf	1.610	1.610	0	1	0
6	P15_P13_2402_dizzy.eaf	0.310	1.270	2	6	2
7	P15_P13_2402_monk.eaf	0.450	1.200	0	2	0
8	P16_P6_0903_dizzy.eaf	inf	0.000	0	0	0
9	P16_P6_0903_monk.eaf	inf	0.000	0	0	0
10	P18_P10_2102_dizzy.eaf	inf	0.000	0	0	0
11	P18_P10_2102_monk.eaf	0.365	0.365	0	1	0
12	P18_P1_2102_dizzy.eaf	inf	0.000	0	0	0
13	P18_P1_2102_monk.eaf	0.620	0.620	0	1	0

IFADV informations for Smiles_0:

	Filename	Min duration	Max duration	low	high	medium	subtle
0	DVA1A.eaf	0.220	8.770	14	1	7	3
1	DVA2C.eaf	0.210	10.950	17	8	17	3
2	DVA3E.eaf	0.410	11.480	4	0	0	3
3	DVA4C.eaf	0.310	5.730	10	6	8	1
4	DVA5G.eaf	0.710	7.220	7	0	2	0
5	DVA6H.eaf	0.250	8.140	7	1	3	0
6	DVA7B.eaf	0.160	7.780	7	10	9	2
7	DVA8K.eaf	0.270	2.590	3	0	4	3
8	DVB1B.eaf	0.180	9.380	12	13	8	1
9	DVB2D.eaf	0.300	8.010	20	5	14	4
10	DVB3F.eaf	0.110	7.230	20	10	16	1
11	DVB4G.eaf	0.240	3.310	24	4	12	10
12	DVB5H.eaf	0.470	8.340	8	2	6	4
13	DVB6I.eaf	0.120	14.510	16	1	3	5
14	DVB7J.eaf	0.395	9.130	20	8	22	0
15	DVB8L.eaf	0.160	7.965	17	14	16	3

IFADV informations for Laughs_0:

	Filename	Min duration	Max duration	medium	low
0	DVA1A.eaf	inf	0.000	0	0
1	DVA2C.eaf	1.240	1.240	1	0
2	DVA3E.eaf	0.550	0.990	1	3
3	DVA4C.eaf	0.490	1.245	0	2
4	DVA5G.eaf	inf	0.000	0	0
5	DVA6H.eaf	inf	0.000	0	0
6	DVA7B.eaf	0.570	1.060	0	4
7	DVA8K.eaf	0.230	9.590	1	4
8	DVB1B.eaf	0.890	1.375	0	2
9	DVB2D.eaf	0.502	6.280	0	2
10	DVB3F.eaf	0.605	1.815	0	4
11	DVB4G.eaf	0.750	1.190	0	3
12	DVB5H.eaf	0.455	0.455	0	1
13	DVB6I.eaf	0.650	1.675	0	2
14	DVB7J.eaf	inf	0.000	0	0
15	DVB8L.eaf	1.050	2.110	1	4

NDC informations for Smiles_0:

	Filename	Min duration	Max duration	low	high	medium	subtle
0	13_1_A_M.eaf	0.250	4.860	7	2	3	4
1	13_1_B_F.eaf	0.265	12.380	8	12	8	2
2	13_2_A_M.eaf	0.400	6.460	13	2	3	5
3	13_2_B_F.eaf	0.155	7.020	30	21	14	4
4	13_3_B_F.eaf	0.325	12.320	12	19	18	3
5	13_4_A_M.eaf	0.430	5.105	24	7	7	5
6	13_4_B_F.eaf	0.170	8.530	39	29	42	11
7	14_1_A_M.eaf	0.170	3.280	12	1	2	10
8	14_1_B_F.eaf	0.230	8.860	20	5	10	15
9	14_2_A_M.eaf	0.320	3.050	18	3	6	11
10	14_2_B_F.eaf	0.170	9.900	29	31	26	14
11	17_1_A_F.eaf	0.445	7.000	6	2	3	4
12	17_1_B_F.eaf	0.220	12.750	5	1	3	5
13	17_2_A_F.eaf	0.560	12.380	4	4	4	10
14	17_2_B_F.eaf	0.900	18.070	3	1	5	5
15	17_3_A_F.eaf	0.510	2.980	5	4	3	5
16	17_3_B_F.eaf	0.240	10.320	9	4	6	4
17	17_4_A_F.eaf	0.382	23.010	16	5	9	14
18	17_4_B_F.eaf	0.160	16.860	22	3	14	15
19	18_1_A_M.eaf	0.150	5.790	18	3	21	24
20	18_1_B_M.eaf	0.220	6.950	33	8	11	19
21	20_2_A_M.eaf	0.330	8.449	9	1	7	12
22	20_2_B_M.eaf	1.060	2.610	2	2	2	0
23	21_1_A_M.eaf	0.270	15.800	4	4	5	8
24	21_1_B_M.eaf	0.240	9.700	13	3	5	7
25	21_2_A_M.eaf	0.360	7.145	9	15	16	9
26	8_1_A_M.eaf	0.490	9.650	22	0	13	22
27	8_1_B_M.eaf	0.200	25.510	7	7	3	13
28	8_2_A_M.eaf	0.300	10.910	6	3	6	17
29	8_2_B_M.eaf	0.385	12.270	22	3	9	17
30	8_3_A_M.eaf	0.230	6.740	22	8	16	20
31	8_3_B_M.eaf	0.230	18.320	14	7	8	9
32	8_4_A_M.eaf	0.310	8.410	17	5	10	18
33	8_4_B_M.eaf	0.390	15.150	11	3	8	11

NDC informations for Laughs_0:

	Filename	Min duration	Max duration	medium	low	high
0	13_1_A_M.eaf	0.720	1.100	0	3	0
1	13_1_B_F.eaf	0.290	12.380	3	5	3
2	13_2_A_M.eaf	0.860	0.860	0	1	0
3	13_2_B_F.eaf	0.400	0.595	1	1	0
4	13_3_B_F.eaf	0.460	2.150	3	3	5
5	13_4_A_M.eaf	0.250	1.271	1	6	0
6	13_4_B_F.eaf	0.405	2.050	6	4	4
7	14_1_A_M.eaf	0.400	1.020	0	4	0
8	14_1_B_F.eaf	inf	0.000	0	0	0
9	14_2_A_M.eaf	0.450	0.560	1	1	0
10	14_2_B_F.eaf	0.510	1.020	5	1	0
11	17_1_A_F.eaf	0.870	1.765	1	1	2
12	17_1_B_F.eaf	1.060	1.435	0	2	0
13	17_2_A_F.eaf	1.350	3.330	1	1	0
14	17_2_B_F.eaf	0.450	4.920	1	2	1
15	17_3_A_F.eaf	0.700	3.370	3	2	0
16	17_3_B_F.eaf	0.605	1.960	0	4	0
17	17_4_A_F.eaf	1.280	1.440	1	1	1
18	17_4_B_F.eaf	0.580	1.900	2	2	0
19	18_1_A_M.eaf	0.280	1.320	2	5	0
20	18_1_B_M.eaf	0.440	1.860	2	2	0
21	20_2_A_M.eaf	1.050	1.050	0	1	0
22	20_2_B_M.eaf	0.485	0.910	2	1	0
23	21_1_A_M.eaf	0.575	3.340	0	3	1
24	21_1_B_M.eaf	2.240	3.660	1	1	0
25	21_2_A_M.eaf	0.440	3.130	5	5	0
26	8_1_A_M.eaf	1.055	2.080	2	4	1
27	8_1_B_M.eaf	1.130	3.490	3	3	2
28	8_2_A_M.eaf	0.710	1.540	0	3	0
29	8_2_B_M.eaf	0.260	1.630	5	5	0
30	8_3_A_M.eaf	0.545	3.055	4	2	0
31	8_3_B_M.eaf	0.805	4.950	6	8	0
32	8_4_A_M.eaf	0.940	2.020	2	2	0
33	8_4_B_M.eaf	0.900	1.970	8	5	1

For CCDB:

- Smiles:
  - Min duration: The values vary between 0.105 and 0.675 s.
  - Max duration: The values vary between 4.260 and 14.511 s.
  - Count: subtle -> 61, low -> 154, medium -> 130, high -> 109
- Laughs:
  - Min duration: The values vary between 0 and 1.610 s.
  - Max duration: The values vary between 0 and 4.240 s.
  - Count: low -> 22, medium -> 8, high -> 7

For IFADV:

- Smiles:
  - Min duration: The values vary between 0.110 and 0.710 s.
  - Max duration: The values vary between 2.590 and 14.510 s.
  - Count: subtle -> 43, low -> 206, medium -> 147, high -> 83
- Laughs:
  - Min duration: The values vary between 0.455 and 1.240 s.
  - Max duration: The values vary between 0.455 and 9.590 s.
  - Count: low -> 31, medium -> 4, high -> 0

For NDC:

- Smiles:
  - Min duration: The values vary between 0.150 and 1.060 s.
  - Max duration: The values vary between 2.610 and 25.510 s.
  - Count: subtle -> 352, low -> 491, medium -> 326, high -> 228
- Laughs:
  - Min duration: The values vary between 0 and 2.240 s.
  - Max duration: The values vary between 0 and 12.380 s.
  - Count: low -> 94, medium -> 71, high -> 21

- Duration of "Smiles":
  - CCDB and IFADV have a higher maximum duration than NDC.
  - "Smiles" in NDC have a higher minimum duration compared to other datasets.
- Duration of "Laughs":
  - IFADV has the highest maximum duration among the datasets, while CCDB has a lower maximum duration.
  - CCDB has a greater variety of laughter intensities, while IFADV does not contain any recordings classified with high laughter intensity.
- Distribution of intensities:
  - CCDB has a balanced distribution of intensities, with relatively equal frequency between categories.
  - IFADV: the majority of "Smiles" and "Laughs" are classified with a low or subtle intensity.
  - NDC presents a predominance of low intensities for the "Laughs", while the "Smiles" have a more balanced distribution.

Impact of the role on the distribution of S&L:

In [ ]:
```
for i in databases.keys():
    databases_list = databases_paths[i]
    for role in tier_lists["Role"]:
        df = display_filtered_informations(databases_list, i.replace('_paths','')
        display(Markdown(f"## {i.replace('_paths','').upper()} informations for
        display(df)
```

CCDB informations for lsn:

	Filename	Count_Smiles_0	Count_Laughs_0	Duration_Smiles_0 (ms)	Duration_Laugh (
0	P12_P2_1402_dizzy.eaf	40	1	123035	1
1	P12_P2_1402_monk.eaf	32	2	86385	1
2	P12_P3_2202_dizzy.eaf	6	0	8808	
3	P12_P3_2202_monk.eaf	37	5	150827	4
4	P14_P7_2502_dizzy.eaf	10	8	15590	17
5	P14_P7_2502_monk.eaf	14	1	69440	1
6	P15_P13_2402_dizzy.eaf	51	9	58245	7
7	P15_P13_2402_monk.eaf	18	1	67573	
8	P16_P6_0903_dizzy.eaf	16	0	32000	
9	P16_P6_0903_monk.eaf	3	0	11590	
10	P18_P10_2102_dizzy.eaf	9	0	37645	
11	P18_P10_2102_monk.eaf	16	1	28915	
12	P18_P1_2102_dizzy.eaf	13	0	37475	
13	P18_P1_2102_monk.eaf	12	1	33205	

CCDB informations for spk:

	Filename	Count_Smiles_0	Count_Laughs_0	Duration_Smiles_0 (ms)	Duration_Laugh (
0	P12_P2_1402_dizzy.eaf	35	1	111210	1
1	P12_P2_1402_monk.eaf	46	4	123445	2
2	P12_P3_2202_dizzy.eaf	26	0	56733	
3	P12_P3_2202_monk.eaf	14	2	72629	1
4	P14_P7_2502_dizzy.eaf	20	5	52903	10
5	P14_P7_2502_monk.eaf	10	0	29440	
6	P15_P13_2402_dizzy.eaf	24	3	20200	2
7	P15_P13_2402_monk.eaf	18	1	69688	1
8	P16_P6_0903_dizzy.eaf	9	0	15940	
9	P16_P6_0903_monk.eaf	20	0	52825	
10	P18_P10_2102_dizzy.eaf	38	0	72180	
11	P18_P10_2102_monk.eaf	16	0	23910	
12	P18_P1_2102_dizzy.eaf	37	0	60700	
13	P18_P1_2102_monk.eaf	25	0	36785	

IFADV informations for lsn:

	Filename	Count_Smiles_0	Count_Laughs_0	Duration_Smiles_0 (ms)	Duration_Laughs_0 (ms)
0	DVA1A.eaf	7	0	31770	0
1	DVA2C.eaf	26	1	90240	1240
2	DVA3E.eaf	5	2	23760	1540
3	DVA4C.eaf	22	2	54785	1735
4	DVA5G.eaf	6	0	26700	0
5	DVA6H.eaf	12	0	27690	0
6	DVA7B.eaf	18	3	69345	2350
7	DVA8K.eaf	1	2	270	10885
8	DVB1B.eaf	30	1	100460	890
9	DVB2D.eaf	36	2	89330	7930
10	DVB3F.eaf	12	2	20470	2915
11	DVB4G.eaf	35	1	56710	1190
12	DVB5H.eaf	22	1	67570	455
13	DVB6I.eaf	22	2	73350	2325
14	DVB7J.eaf	33	0	121370	0
15	DVB8L.eaf	30	1	58545	1935

IFADV informations for spk:

	Filename	Count_Smiles_0	Count_Laughs_0	Duration_Smiles_0 (ms)	Duration_Laughs_0 (ms)
0	DVA1A.eaf	27	0	63870	0
1	DVA2C.eaf	48	0	118320	0
2	DVA3E.eaf	4	2	15270	1340
3	DVA4C.eaf	16	0	33095	0
4	DVA5G.eaf	9	0	28870	0
5	DVA6H.eaf	9	0	21970	0
6	DVA7B.eaf	26	3	68110	2140
7	DVA8K.eaf	9	6	7070	24510
8	DVB1B.eaf	18	2	48385	2265
9	DVB2D.eaf	34	2	89183	6782
10	DVB3F.eaf	33	3	52115	3520
11	DVB4G.eaf	46	3	69930	3100
12	DVB5H.eaf	20	0	60165	0
13	DVB6I.eaf	20	3	76500	4000
14	DVB7J.eaf	52	0	148225	0
15	DVB8L.eaf	36	4	51290	6595

NDC informations for lsn:

	Filename	Count_Smiles_0	Count_Laughs_0	Duration_Smiles_0 (ms)	Duration_Laughs_0 (ms)
0	13_1_A_M.eaf	7	2	11402	1440
1	13_1_B_F.eaf	20	8	74881	21515
2	13_2_A_M.eaf	23	1	49976	860
3	13_2_B_F.eaf	14	0	36148	0
4	13_3_B_F.eaf	45	9	170769	11260
5	13_4_A_M.eaf	42	7	79668	5514
6	13_4_B_F.eaf	19	3	53022	4750
7	14_1_A_M.eaf	2	0	1630	0
8	14_1_B_F.eaf	38	0	73100	0
9	14_2_A_M.eaf	19	1	29150	560
10	14_2_B_F.eaf	36	3	80635	1975
11	17_1_A_F.eaf	5	1	19755	1565
12	17_1_B_F.eaf	9	1	33700	1060
13	17_2_A_F.eaf	19	2	83070	4680
14	17_2_B_F.eaf	4	2	19360	3580
15	17_3_A_F.eaf	14	3	19770	5560
16	17_3_B_F.eaf	20	2	74906	2565
17	17_4_A_F.eaf	35	2	112685	2750
18	17_4_B_F.eaf	9	2	43280	2235
19	18_1_A_M.eaf	27	3	52743	2485
20	18_1_B_M.eaf	61	3	100399	3950
21	20_2_A_M.eaf	1	0	7650	0
22	20_2_B_M.eaf	6	3	9660	1905
23	21_1_A_M.eaf	9	2	26775	4050
24	21_1_B_M.eaf	24	2	86945	5900
25	21_2_A_M.eaf	46	10	109605	15220
26	8_1_A_M.eaf	22	4	64874	6465
27	8_1_B_M.eaf	26	6	175750	13960
28	8_2_A_M.eaf	33	3	121375	3330
29	8_2_B_M.eaf	9	0	27575	0
30	8_3_A_M.eaf	11	0	26207	0
31	8_3_B_M.eaf	36	13	206371	33680
32	8_4_A_M.eaf	46	4	111165	6155
33	8_4_B_M.eaf	4	1	14680	1220

NDC informations for spk:

	Filename	Count_Smiles_0	Count_Laughs_0	Duration_Smiles_0 (ms)	Duration_Laughs_0 (ms)
0	13_1_A_M.eaf	14	2	23232	1820
1	13_1_B_F.eaf	17	4	38060	5590
2	13_2_A_M.eaf	11	0	29531	0
3	13_2_B_F.eaf	66	2	93411	995
4	13_3_B_F.eaf	37	5	126809	6410
5	13_4_A_M.eaf	16	2	32114	1670
6	13_4_B_F.eaf	125	15	231534	13523
7	14_1_A_M.eaf	24	4	30885	3145
8	14_1_B_F.eaf	17	0	15095	0
9	14_2_A_M.eaf	29	1	40675	450
10	14_2_B_F.eaf	87	4	127785	3345
11	17_1_A_F.eaf	12	4	22590	5815
12	17_1_B_F.eaf	3	1	5915	1435
13	17_2_A_F.eaf	7	0	33231	0
14	17_2_B_F.eaf	12	3	88760	9980
15	17_3_A_F.eaf	8	2	8640	2528
16	17_3_B_F.eaf	13	3	51360	4270
17	17_4_A_F.eaf	11	2	23992	2720
18	17_4_B_F.eaf	53	2	187430	2480
19	18_1_A_M.eaf	64	5	104693	4130
20	18_1_B_M.eaf	41	1	61275	440
21	20_2_A_M.eaf	29	1	72509	1050
22	21_1_A_M.eaf	16	3	72467	4555
23	21_1_B_M.eaf	11	0	38425	0
24	21_2_A_M.eaf	11	1	29900	1010
25	8_1_A_M.eaf	50	3	145390	4520
26	8_1_B_M.eaf	19	4	133100	8020
27	8_2_A_M.eaf	6	0	27370	0
28	8_2_B_M.eaf	50	10	122955	7945
29	8_3_A_M.eaf	59	6	130937	8430
30	8_3_B_M.eaf	7	4	33146	8380
31	8_4_A_M.eaf	10	0	20140	0
32	8_4_B_M.eaf	14	9	45645	10675

Database information summary for role impact on S&L:

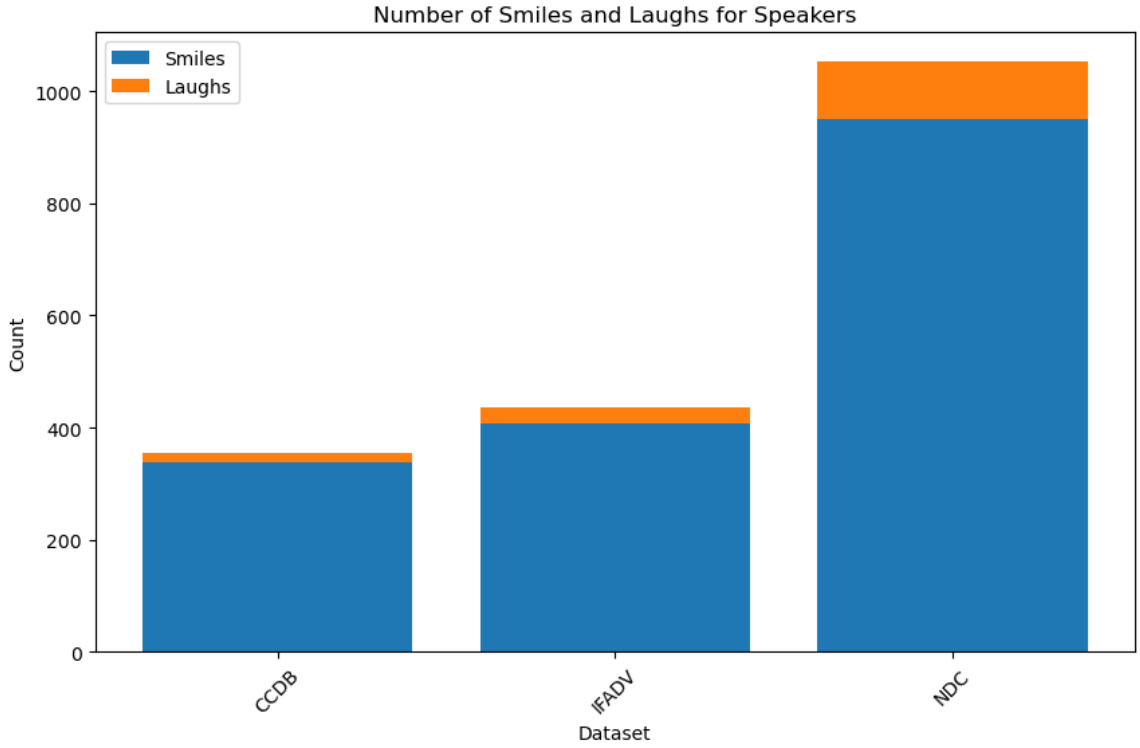
In [ ]:
```
result_data = []
for i in databases.keys():
    databases_list = databases_paths[i]
    for role in tier_lists["Role"]:
        df = display_filtered_informations(databases_list, i.replace('_paths','')
        tier_totals = []
        for tier in expressions:
            count_total = df[f'Count_{tier}'].sum()
            duration_total = df[f'Duration_{tier} (ms)'].sum()
            tier_totals.extend([count_total, duration_total])

        role_data = [i.replace('_paths','').upper(), role] + tier_totals
        result_data.append(role_data)
columns = ['Dataset','Role']
for tier in expressions:
    columns.extend([f'count_total_{tier}', f'duration_total_{tier} (ms)'])

result_df = pd.DataFrame(result_data, columns=columns)
display(Markdown(f"## Database informations:"))
display(result_df)
```

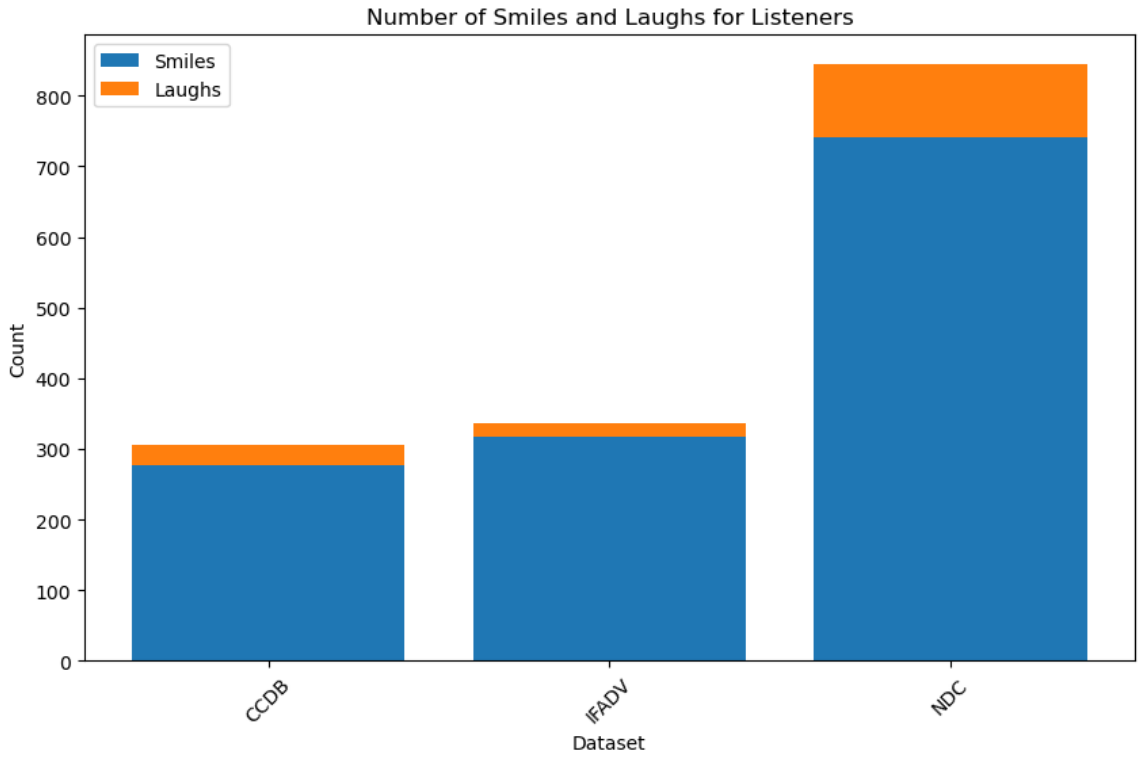
Database informations:

	Dataset	Role	count_total_Smiles_0	duration_total_Smiles_0 (ms)	count_total_Laughs_0	duration_
0	CCDB	lsn	277	760733	29	
1	CCDB	spk	338	798588	16	
2	IFADV	lsn	317	912365	20	
3	IFADV	spk	407	952368	28	
4	NDC	lsn	741	2208681	103	
5	NDC	spk	949	2249001	103	

In [ ]:
```
result_data = []
for i in databases.keys():
    databases_list = databases_paths[i]
    for role in tier_lists["Role"]:
        df = display_filtered_informations(databases_list, i.replace('_paths','')
        tier_totals = []
        for tier in expressions:
            count_total = df[f'Count_{tier}'].sum()
            duration_total = df[f'Duration_{tier} (ms)'].sum()
            tier_totals.extend([count_total, duration_total])

        role_data = [i.replace('_paths','').upper(), role] + tier_totals
        result_data.append(role_data)
columns = ['Dataset','Role']
for tier in expressions:
    columns.extend([f'count_total_{tier}', f'duration_total_{tier} (ms)'])

result_df = pd.DataFrame(result_data, columns=columns)
plot_df = result_df[['Dataset', 'Role', 'count_total_Smiles_0', 'count_total_Lau
```

```
# split dataframe into speakers and listeners
spk_df = plot_df[plot_df['Role'] == 'spk']
lsn_df = plot_df[plot_df['Role'] == 'lsn']

# prepare data for graph
spk_labels = spk_df['Dataset']
spk_smiles = spk_df['count_total_Smiles_0']
spk_laughs = spk_df['count_total_Laughs_0']

lsn_labels = lsn_df['Dataset']
lsn_smiles = lsn_df['count_total_Smiles_0']
lsn_laughs = lsn_df['count_total_Laughs_0']

# create speaker histogram
plt.figure(figsize=(10, 6))
plt.bar(spk_labels, spk_smiles, label='Smiles')
plt.bar(spk_labels, spk_laughs, bottom=spk_smiles, label='Laughs')
plt.xlabel('Dataset')
plt.ylabel('Count')
plt.title('Number of Smiles and Laughs for Speakers')
plt.legend()
plt.xticks(rotation=45)
plt.show()

# create listener histogram
plt.figure(figsize=(10, 6))
plt.bar(lsn_labels, lsn_smiles, label='Smiles')
plt.bar(lsn_labels, lsn_laughs, bottom=lsn_smiles, label='Laughs')
plt.xlabel('Dataset')
plt.ylabel('Count')
plt.title('Number of Smiles and Laughs for Listeners')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```





We can conclude that the datasets are not balanced between "spk" and "lsn" roles. There is a difference in the number of samples available for each role in each dataset. This must be taken into account when training the ML model to account for this disparity and avoid any potential bias in the predictions.

Impact of the role on the intensities of S&L:

In [ ]:
```
for i in databases.keys():
    databases_list = databases_paths[i]
    for tier in expressions :
        combined_df = pd.DataFrame()
        added_columns = set()
        filename_columns = []
        for role in tier_lists["Role"]:
            df = display_filtered_informations_by_entity(databases_list, i.repla
            new_columns = {col: f"{col}_for_{role}" for col in df.columns}
            df = df.rename(columns=new_columns)
            columns_to_add = [col for col in df.columns if col not in added_colu
            if columns_to_add:
                df = df[columns_to_add]
                combined_df = pd.concat([combined_df, df], axis=1)
                added_columns.update(columns_to_add)
            filename_columns.extend([col for col in df.columns if col.startswith
        filename_col_1 = combined_df[filename_columns[0]]
        filename_col_2 = combined_df[filename_columns[1]]

        if filename_col_1.hasnans:
            combined_df["Filename"] = filename_col_2
        else:
            combined_df["Filename"] = filename_col_1

        combined_df = combined_df.drop(columns=[filename_columns[0], filename_cc
        combined_df = combined_df.fillna(0)
        filename_column = combined_df.pop("Filename")
```

```
        combined_df.insert(0, "Filename", filename_column)

        display(Markdown(f"## {i.replace('_paths','').upper()} informations for
        display(combined_df)
```

CCDB informations for Smiles_0:

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0
0	P12_P2_1402_dizzy.eaf	14.0	medium	
1	P12_P2_1402_dizzy.eaf	13.0	low	
2	P12_P2_1402_dizzy.eaf	8.0	high	
3	P12_P2_1402_dizzy.eaf	5.0	subtle	
4	P12_P2_1402_monk.eaf	14.0	medium	
5	P12_P2_1402_monk.eaf	10.0	high	
6	P12_P2_1402_monk.eaf	8.0	low	
7	P12_P3_2202_dizzy.eaf	2.0	high	
8	P12_P3_2202_dizzy.eaf	2.0	medium	
9	P12_P3_2202_dizzy.eaf	2.0	low	
10	P12_P3_2202_dizzy.eaf	17.0	low	
11	P12_P3_2202_monk.eaf	11.0	medium	
12	P12_P3_2202_monk.eaf	6.0	subtle	
13	P12_P3_2202_monk.eaf	3.0	high	
14	P14_P7_2502_dizzy.eaf	4.0	low	
15	P14_P7_2502_dizzy.eaf	2.0	subtle	
16	P14_P7_2502_dizzy.eaf	2.0	medium	
17	P14_P7_2502_dizzy.eaf	2.0	high	
18	P14_P7_2502_monk.eaf	7.0	high	
19	P14_P7_2502_monk.eaf	7.0	medium	
20	P14_P7_2502_monk.eaf	22.0	low	
21	P15_P13_2402_dizzy.eaf	17.0	high	
22	P15_P13_2402_dizzy.eaf	8.0	medium	
23	P15_P13_2402_dizzy.eaf	4.0	subtle	
24	P15_P13_2402_dizzy.eaf	10.0	high	
25	P15_P13_2402_monk.eaf	5.0	medium	
26	P15_P13_2402_monk.eaf	3.0	low	
27	P15_P13_2402_monk.eaf	6.0	low	
28	P15_P13_2402_monk.eaf	6.0	subtle	
29	P16_P6_0903_dizzy.eaf	3.0	medium	
30	P16_P6_0903_dizzy.eaf	1.0	high	
31	P16_P6_0903_dizzy.eaf	1.0	subtle	
32	P16_P6_0903_monk.eaf	1.0	medium	
33	P16_P6_0903_monk.eaf	1.0	low	
34	P16_P6_0903_monk.eaf	4.0	high	

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0
35	P18_P10_2102_dizzy.eaf	4.0	medium	
36	P18_P10_2102_dizzy.eaf	1.0	low	
37	P18_P10_2102_dizzy.eaf	7.0	low	
38	P18_P10_2102_dizzy.eaf	4.0	medium	
39	P18_P10_2102_monk.eaf	3.0	subtle	
40	P18_P10_2102_monk.eaf	2.0	high	
41	P18_P10_2102_monk.eaf	8.0	high	
42	P18_P10_2102_monk.eaf	5.0	medium	
43	P18_P1_2102_dizzy.eaf	4.0	high	
44	P18_P1_2102_dizzy.eaf	4.0	low	
45	P18_P1_2102_dizzy.eaf	3.0	medium	
46	P18_P1_2102_dizzy.eaf	1.0	subtle	
47	P18_P1_2102_monk.eaf	0.0	0	
48	P18_P1_2102_monk.eaf	0.0	0	
49	P18_P1_2102_monk.eaf	0.0	0	
50	P18_P1_2102_monk.eaf	0.0	0	

CCDB informations for Laughs_0:

	Filename	Count_Laughs_0_for_lsn	Intensity_Laughs_0_for_lsn	Diff_time_Laughs
0	P12_P2_1402_dizzy.eaf	1	medium	
1	P12_P2_1402_monk.eaf	2	low	
2	P12_P3_2202_monk.eaf	3	low	
3	P12_P3_2202_monk.eaf	2	medium	
4	P14_P7_2502_dizzy.eaf	4	high	
5	P14_P7_2502_dizzy.eaf	2	medium	
6	P14_P7_2502_dizzy.eaf	2	low	
7	P14_P7_2502_monk.eaf	1	low	
8	P15_P13_2402_dizzy.eaf	5	low	
9	P15_P13_2402_dizzy.eaf	2	medium	
10	P15_P13_2402_dizzy.eaf	2	high	
11	P15_P13_2402_monk.eaf	1	low	
12	P18_P10_2102_monk.eaf	1	low	
13	P18_P1_2102_monk.eaf	1	low	

IFADV informations for Smiles_0:

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0_for_lsn	Co
0	DVA1A.eaf	5.0	low	20710.0	
1	DVA1A.eaf	2.0	medium	11060.0	
2	DVA1A.eaf	16.0	medium	46700.0	
3	DVA1A.eaf	7.0	low	34000.0	
4	DVA2C.eaf	2.0	high	8490.0	
5	DVA2C.eaf	1.0	subtle	1050.0	
6	DVA2C.eaf	3.0	subtle	21320.0	
7	DVA2C.eaf	2.0	low	2440.0	
8	DVA3E.eaf	10.0	low	23325.0	
9	DVA3E.eaf	6.0	medium	16440.0	
10	DVA4C.eaf	5.0	high	11150.0	
11	DVA4C.eaf	1.0	subtle	3870.0	
12	DVA4C.eaf	4.0	low	12260.0	
13	DVA5G.eaf	2.0	medium	14440.0	
14	DVA5G.eaf	7.0	low	11520.0	
15	DVA6H.eaf	3.0	medium	12150.0	
16	DVA6H.eaf	2.0	high	4020.0	
17	DVA6H.eaf	10.0	high	42765.0	
18	DVA7B.eaf	4.0	medium	17000.0	
19	DVA7B.eaf	2.0	low	3330.0	
20	DVA7B.eaf	2.0	subtle	6250.0	
21	DVA8K.eaf	1.0	subtle	270.0	
22	DVA8K.eaf	11.0	low	23885.0	
23	DVA8K.eaf	10.0	high	41365.0	
24	DVB1B.eaf	8.0	medium	34940.0	
25	DVB1B.eaf	1.0	subtle	270.0	
26	DVB1B.eaf	14.0	low	22650.0	
27	DVB2D.eaf	9.0	medium	17750.0	
28	DVB2D.eaf	9.0	high	32390.0	
29	DVB2D.eaf	4.0	subtle	16540.0	
30	DVB2D.eaf	6.0	medium	11400.0	
31	DVB3F.eaf	4.0	low	6150.0	
32	DVB3F.eaf	2.0	high	2920.0	
33	DVB3F.eaf	20.0	low	34430.0	
34	DVB3F.eaf	8.0	medium	11585.0	

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0_for_lsn	Co
35	DVB4G.eaf	4.0	high	7995.0	
36	DVB4G.eaf	3.0	subtle	2700.0	
37	DVB4G.eaf	8.0	low	32400.0	
38	DVB4G.eaf	7.0	medium	20830.0	
39	DVB5H.eaf	4.0	high	7660.0	
40	DVB5H.eaf	3.0	subtle	6680.0	
41	DVB5H.eaf	15.0	low	62850.0	
42	DVB5H.eaf	4.0	medium	7510.0	
43	DVB6I.eaf	2.0	subtle	1670.0	
44	DVB6I.eaf	1.0	high	1320.0	
45	DVB6I.eaf	18.0	medium	90265.0	
46	DVB6I.eaf	9.0	low	23235.0	
47	DVB7J.eaf	6.0	high	7870.0	
48	DVB7J.eaf	12.0	low	24145.0	
49	DVB7J.eaf	7.0	high	22470.0	
50	DVB8L.eaf	7.0	medium	8000.0	
51	DVB8L.eaf	4.0	subtle	3930.0	
52	DVB8L.eaf	0.0	0	0.0	
53	DVB8L.eaf	0.0	0	0.0	

IFADV informations for Laughs_0:

	Filename	Count_Laughs_0_for_lsn	Intensity_Laughs_0_for_lsn	Diff_time_Laughs_0_for_lsn	C
0	DVA2C.eaf	1	medium	1240	
1	DVA3E.eaf	2	low	1540	
2	DVA4C.eaf	2	low	1735	
3	DVA7B.eaf	3	low	2350	
4	DVA8K.eaf	2	low	10885	
5	DVB1B.eaf	1	low	890	
6	DVB2D.eaf	1		6280	
7	DVB2D.eaf	1	low	1650	
8	DVB3F.eaf	2	low	2915	
9	DVB4G.eaf	1	low	1190	
10	DVB5H.eaf	1	low	455	
11	DVB6I.eaf	2	low	2325	
12	DVB8L.eaf	1	low	1935	

NDC informations for Smiles_0:

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0_for_lsn
0	13_1_A_M.eaf	3.0	subtle	8002.0
1	13_1_A_M.eaf	2.0	low	2380.0
2	13_1_A_M.eaf	2.0	medium	1020.0
3	13_1_A_M.eaf	10.0	high	38875.0
4	13_1_B_F.eaf	5.0	low	21606.0
5	13_1_B_F.eaf	5.0	medium	14400.0
6	13_1_B_F.eaf	12.0	low	22740.0
7	13_1_B_F.eaf	6.0	subtle	23091.0
8	13_2_A_M.eaf	3.0	medium	2700.0
9	13_2_A_M.eaf	2.0	high	1445.0
10	13_2_A_M.eaf	6.0	high	21448.0
11	13_2_B_F.eaf	5.0	low	4230.0
12	13_2_B_F.eaf	2.0	medium	10080.0
13	13_2_B_F.eaf	1.0	subtle	390.0
14	13_2_B_F.eaf	17.0	medium	75830.0
15	13_3_B_F.eaf	16.0	high	67190.0
16	13_3_B_F.eaf	11.0	low	22679.0
17	13_3_B_F.eaf	1.0	subtle	5070.0
18	13_3_B_F.eaf	23.0	low	46148.0
19	13_4_A_M.eaf	8.0	high	14185.0
20	13_4_A_M.eaf	6.0	medium	10925.0
21	13_4_A_M.eaf	5.0	subtle	8410.0
22	13_4_B_F.eaf	8.0	medium	21875.0
23	13_4_B_F.eaf	6.0	low	17040.0
24	13_4_B_F.eaf	5.0	high	14107.0
25	13_4_B_F.eaf	1.0	low	1170.0
26	14_1_A_M.eaf	1.0	medium	460.0
27	14_1_A_M.eaf	20.0	low	32960.0
28	14_1_A_M.eaf	13.0	subtle	34805.0
29	14_1_A_M.eaf	4.0	medium	4420.0
30	14_1_B_F.eaf	1.0	high	915.0
31	14_1_B_F.eaf	10.0	low	16945.0
32	14_1_B_F.eaf	5.0	subtle	7640.0
33	14_1_B_F.eaf	3.0	medium	3500.0
34	14_2_A_M.eaf	1.0	high	1065.0

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0_for_lsn
35	14_2_A_M.eaf	14.0	high	41745.0
36	14_2_A_M.eaf	8.0	medium	21255.0
37	14_2_A_M.eaf	8.0	subtle	8335.0
38	14_2_B_F.eaf	6.0	low	9300.0
39	14_2_B_F.eaf	3.0	subtle	11815.0
40	14_2_B_F.eaf	2.0	low	7940.0
41	14_2_B_F.eaf	3.0	low	10885.0
42	17_1_A_F.eaf	3.0	subtle	15470.0
43	17_1_A_F.eaf	2.0	medium	5362.0
44	17_1_A_F.eaf	1.0	high	1983.0
45	17_1_A_F.eaf	9.0	subtle	44520.0
46	17_1_B_F.eaf	4.0	high	11810.0
47	17_1_B_F.eaf	3.0	medium	10000.0
48	17_1_B_F.eaf	3.0	low	16740.0
49	17_2_A_F.eaf	2.0	medium	9110.0
50	17_2_A_F.eaf	1.0	subtle	6410.0
51	17_2_A_F.eaf	1.0	high	3840.0
52	17_2_A_F.eaf	4.0	high	5560.0
53	17_2_B_F.eaf	4.0	subtle	7080.0
54	17_2_B_F.eaf	3.0	low	3665.0
55	17_2_B_F.eaf	3.0	medium	3465.0
56	17_2_B_F.eaf	8.0	low	39725.0
57	17_3_A_F.eaf	5.0	medium	20305.0
58	17_3_A_F.eaf	4.0	subtle	10640.0
59	17_3_A_F.eaf	3.0	high	4236.0
60	17_3_A_F.eaf	14.0	low	36225.0
61	17_3_B_F.eaf	11.0	subtle	60940.0
62	17_3_B_F.eaf	7.0	medium	9540.0
63	17_3_B_F.eaf	3.0	high	5980.0
64	17_4_A_F.eaf	5.0	low	29150.0
65	17_4_A_F.eaf	2.0	medium	7670.0
66	17_4_A_F.eaf	2.0	subtle	6460.0
67	17_4_A_F.eaf	11.0	subtle	33685.0
68	17_4_B_F.eaf	8.0	low	9320.0
69	17_4_B_F.eaf	6.0	medium	7795.0

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0_for_lsn
70	17_4_B_F.eaf	2.0	high	1943.0
71	17_4_B_F.eaf	30.0	low	53194.0
72	18_1_A_M.eaf	15.0	subtle	16505.0
73	18_1_A_M.eaf	8.0	medium	10500.0
74	18_1_A_M.eaf	8.0	high	20200.0
75	18_1_A_M.eaf	1.0	subtle	7650.0
76	18_1_B_M.eaf	2.0	high	2885.0
77	18_1_B_M.eaf	2.0	medium	3105.0
78	18_1_B_M.eaf	2.0	low	3670.0
79	18_1_B_M.eaf	3.0	low	12140.0
80	20_2_A_M.eaf	3.0	medium	9505.0
81	20_2_A_M.eaf	2.0	subtle	2830.0
82	20_2_A_M.eaf	1.0	high	2300.0
83	20_2_A_M.eaf	12.0	low	36355.0
84	21_1_A_M.eaf	5.0	subtle	18090.0
85	21_1_A_M.eaf	4.0	medium	18840.0
86	21_1_A_M.eaf	2.0	high	13080.0
87	21_1_A_M.eaf	1.0		580.0
88	21_1_B_M.eaf	16.0	medium	52170.0
89	21_1_B_M.eaf	15.0	high	28945.0
90	21_1_B_M.eaf	8.0	low	14025.0
91	21_1_B_M.eaf	7.0	subtle	14465.0
92	21_1_B_M.eaf	10.0	low	32089.0
93	21_2_A_M.eaf	7.0	subtle	15420.0
94	21_2_A_M.eaf	5.0	medium	17365.0
95	21_2_A_M.eaf	13.0	subtle	146980.0
96	21_2_A_M.eaf	7.0	high	16430.0
97	8_1_A_M.eaf	5.0	low	9320.0
98	8_1_A_M.eaf	1.0	medium	3020.0
99	8_1_A_M.eaf	17.0	subtle	68250.0
100	8_1_B_M.eaf	7.0	medium	32320.0
101	8_1_B_M.eaf	6.0	low	15915.0
102	8_1_B_M.eaf	3.0	high	4890.0
103	8_1_B_M.eaf	5.0	low	14840.0
104	8_2_A_M.eaf	2.0	subtle	9010.0

	Filename	Count_Smiles_0_for_lsn	Intensity_Smiles_0_for_lsn	Diff_time_Smiles_0_for_lsn
105	8_2_A_M.eaf	2.0	medium	3725.0
106	8_2_B_M.eaf	4.0	medium	12217.0
107	8_2_B_M.eaf	3.0	subtle	6510.0
108	8_2_B_M.eaf	3.0	low	6230.0
109	8_2_B_M.eaf	1.0	high	1250.0
110	8_3_A_M.eaf	15.0	low	119696.0
111	8_3_A_M.eaf	8.0	subtle	50470.0
112	8_3_A_M.eaf	7.0	high	14830.0
113	8_3_A_M.eaf	6.0	medium	21375.0
114	8_3_B_M.eaf	16.0	subtle	43370.0
115	8_3_B_M.eaf	16.0	low	37030.0
116	8_3_B_M.eaf	9.0	medium	18115.0
117	8_3_B_M.eaf	5.0	high	12650.0
118	8_4_A_M.eaf	2.0	low	6150.0
119	8_4_A_M.eaf	1.0	subtle	6710.0
120	8_4_A_M.eaf	1.0	medium	1820.0
121	8_4_B_M.eaf	0.0	0	0.0
122	8_4_B_M.eaf	0.0	0	0.0
123	8_4_B_M.eaf	0.0	0	0.0
124	8_4_B_M.eaf	0.0	0	0.0

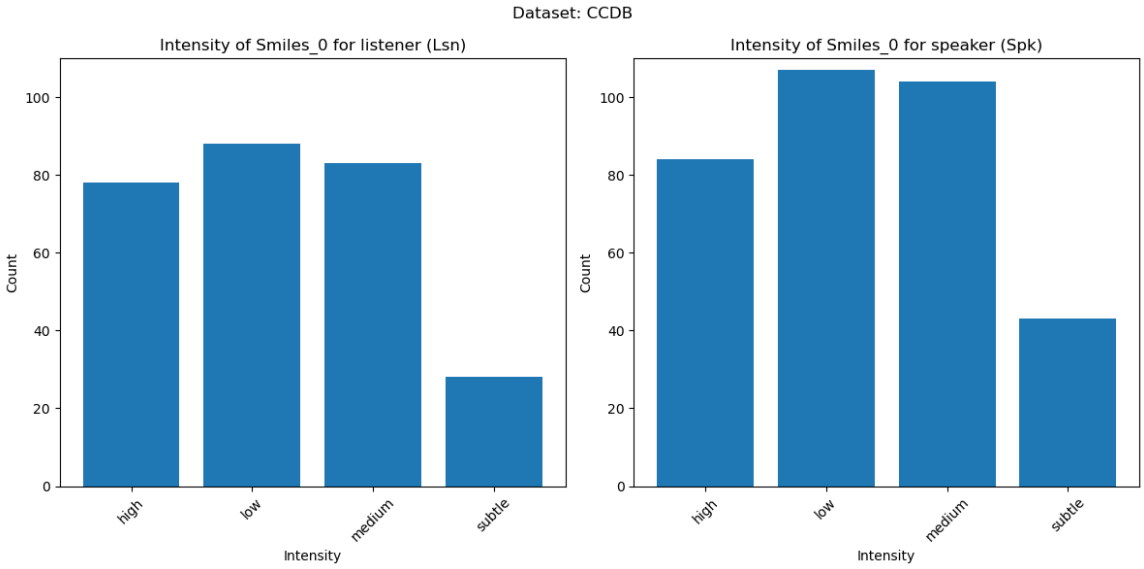
NDC informations for Laughs_0:

	Filename	Count_Laughs_0_for_lsn	Intensity_Laughs_0_for_lsn	Diff_time_Laughs_0_for_lsn
0	13_1_A_M.eaf	2	low	1440
1	13_1_B_F.eaf	4	low	5500
2	13_1_B_F.eaf	2	high	3115
3	13_1_B_F.eaf	2	medium	12900
4	13_2_A_M.eaf	1	low	860
5	13_3_B_F.eaf	3	medium	4810
6	13_3_B_F.eaf	3	high	4470
7	13_3_B_F.eaf	3	low	1980
8	13_4_A_M.eaf	6	low	4741
9	13_4_A_M.eaf	1	medium	773
10	13_4_B_F.eaf	3	high	4750
11	14_2_A_M.eaf	1	low	560
12	14_2_B_F.eaf	2	medium	1465
13	14_2_B_F.eaf	1	low	510
14	17_1_A_F.eaf	1	high	1565
15	17_1_B_F.eaf	1	low	1060
16	17_2_A_F.eaf	1	medium	1350
17	17_2_A_F.eaf	1	low	3330
18	17_2_B_F.eaf	1	medium	3130
19	17_2_B_F.eaf	1	low	450
20	17_3_A_F.eaf	2	low	2190
21	17_3_A_F.eaf	1	medium	3370
22	17_3_B_F.eaf	2	low	2565
23	17_4_A_F.eaf	1	medium	1440
24	17_4_A_F.eaf	1	high	1310
25	17_4_B_F.eaf	1	medium	1480
26	17_4_B_F.eaf	1	low	755
27	18_1_A_M.eaf	2	low	1980
28	18_1_A_M.eaf	1	medium	505
29	18_1_B_M.eaf	2	medium	3000
30	18_1_B_M.eaf	1	low	950
31	20_2_B_M.eaf	2	medium	1420
32	20_2_B_M.eaf	1	low	485
33	21_1_A_M.eaf	1	low	710
34	21_1_A_M.eaf	1	high	3340

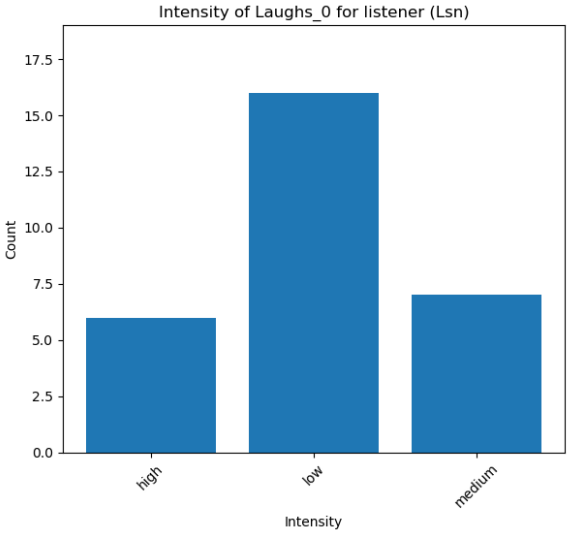
	Filename	Count_Laughs_0_for_lsn	Intensity_Laughs_0_for_lsn	Diff_time_Laughs_0_for_lsn
35	21_1_B_M.eaf	1	medium	3660
36	21_1_B_M.eaf	1	low	2240
37	21_2_A_M.eaf	5	low	8965
38	21_2_A_M.eaf	5	medium	6255
39	8_1_A_M.eaf	2	low	3280
40	8_1_A_M.eaf	1	high	2080
41	8_1_A_M.eaf	1	medium	1105
42	8_1_B_M.eaf	3	low	6170
43	8_1_B_M.eaf	2	medium	4300
44	8_1_B_M.eaf	1	high	3490
45	8_2_A_M.eaf	3	low	3330
46	8_3_B_M.eaf	8	low	17995
47	8_3_B_M.eaf	5	medium	15685
48	8_4_A_M.eaf	2	low	3290
49	8_4_A_M.eaf	2	medium	2865

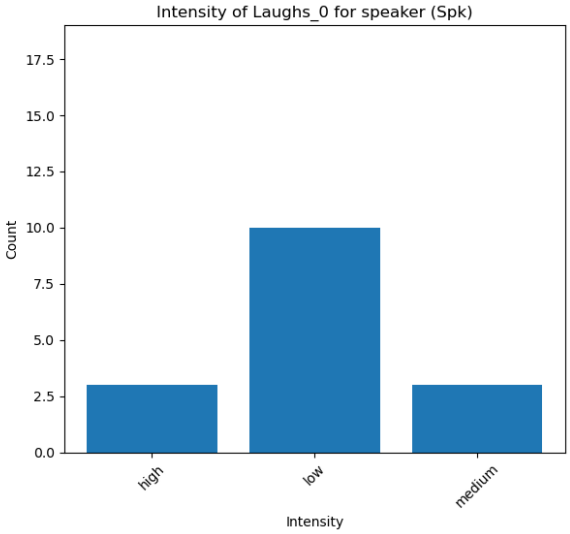
In [ ]:
```
for i in databases.keys():
    databases_list = databases_paths[i]
    for tier in expressions :
        combined_df = pd.DataFrame()
        added_columns = set()
        filename_columns = []
        for role in tier_lists["Role"]:
            df = display_filtered_informations_by_entity(databases_list, i.repla
            new_columns = {col: f"{col}_for_{role}" for col in df.columns}
            df = df.rename(columns=new_columns)
            columns_to_add = [col for col in df.columns if col not in added_colu
            if columns_to_add:
                df = df[columns_to_add]
                combined_df = pd.concat([combined_df, df], axis=1)
                added_columns.update(columns_to_add)
            filename_columns.extend([col for col in df.columns if col.startswith
        filename_col_1 = combined_df[filename_columns[0]]
        filename_col_2 = combined_df[filename_columns[1]]

        if filename_col_1.hasnans:
            combined_df["Filename"] = filename_col_2
        else:
            combined_df["Filename"] = filename_col_1

        globals()[f"{tier}_df"] = combined_df[['Filename', f'Count_{tier}_for_ls

        lsn_data = globals()[f"{tier}_df"][[f'Intensity_{tier}_for_lsn', f'Count
        lsn_data = lsn_data.groupby(f'Intensity_{tier}_for_lsn').sum().reset_inc
        lsn_intensity = lsn_data[f'Intensity_{tier}_for_lsn']
        lsn_count = lsn_data[f'Count_{tier}_for_lsn']
```

```
        spk_data = globals()[f"{tier}_df"][[f'Intensity_{tier}_for_spk', f'Count
        spk_data = spk_data.groupby(f'Intensity_{tier}_for_spk').sum().reset_inc
        spk_intensity = spk_data[f'Intensity_{tier}_for_spk']
        spk_count = spk_data[f'Count_{tier}_for_spk']

        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

        ax1.bar(lsn_intensity.astype(str), lsn_count)
        ax1.set_xlabel('Intensity')
        ax1.set_ylabel('Count')
        ax1.set_title(f'Intensity of {tier} for listener (Lsn)')
        ax1.set_xticklabels(lsn_intensity.astype(str), rotation=45)

        ax2.bar(spk_intensity.astype(str), spk_count)
        ax2.set_xlabel('Intensity')
        ax2.set_ylabel('Count')
        ax2.set_title(f'Intensity of {tier} for speaker (Spk)')
        ax2.set_xticklabels(spk_intensity.astype(str), rotation=45)

        fig.suptitle(f'Dataset: {i.replace("_paths","").upper()}')

        max_count = max(lsn_count.max(), spk_count.max()) + 3
        ax1.set_ylim(0, max_count)
        ax2.set_ylim(0, max_count)

        plt.tight_layout()
        plt.show()
```

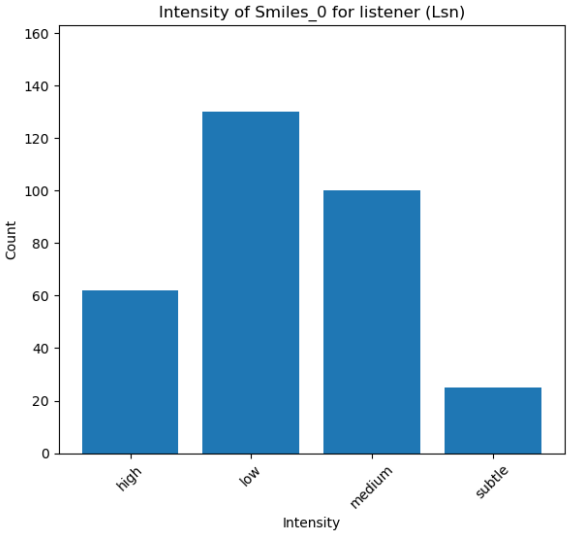


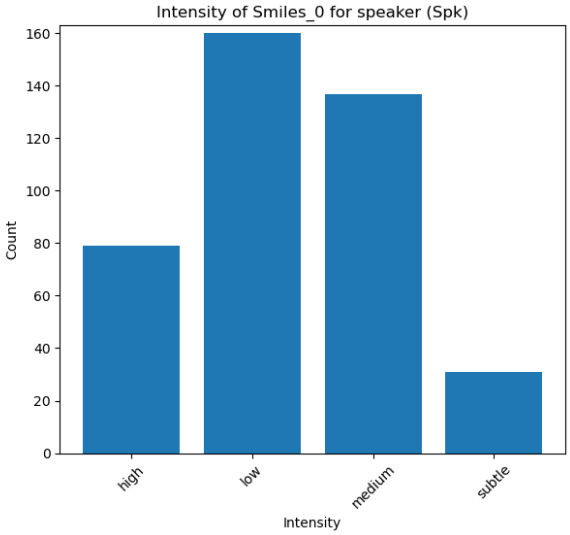
Dataset: CCDB



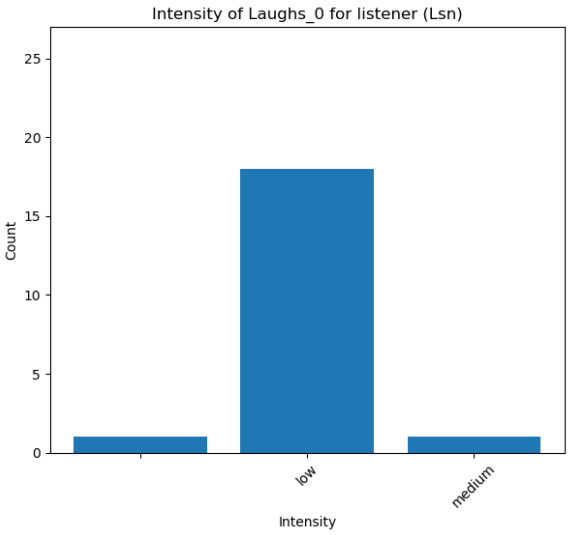


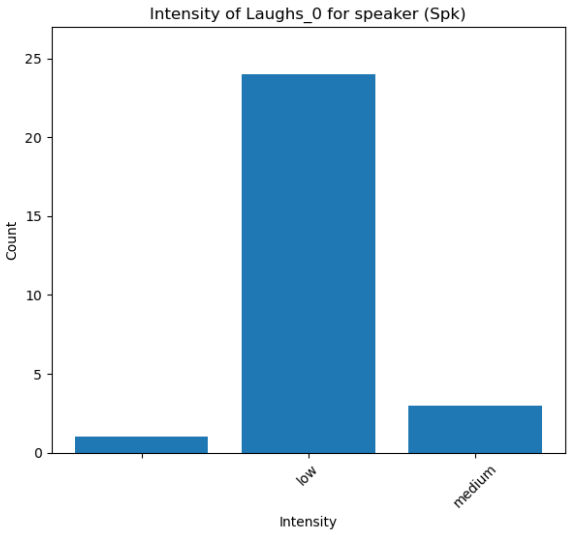
Dataset: IFADV

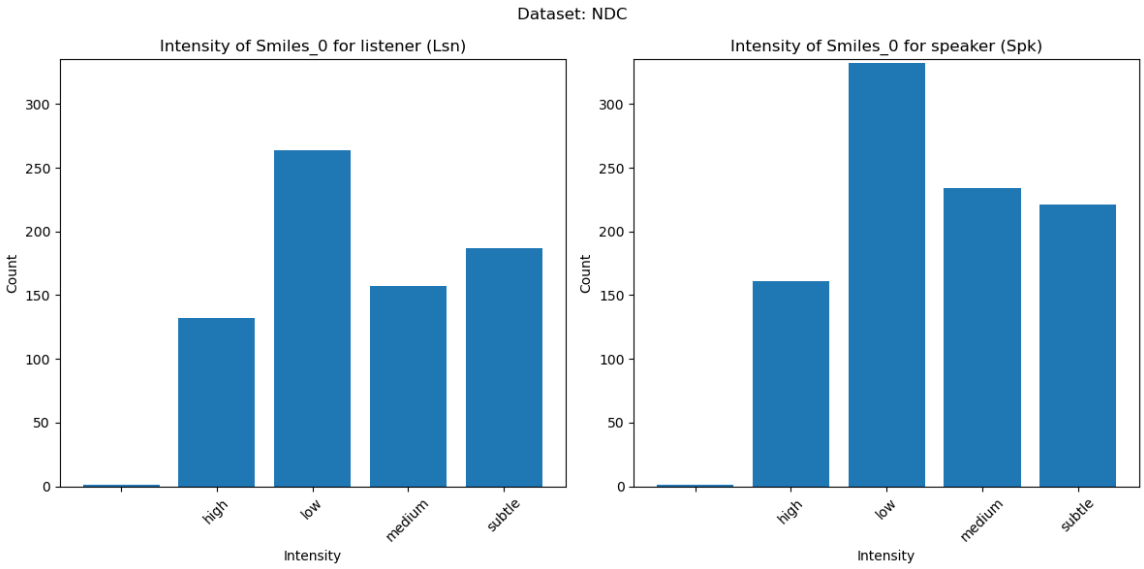


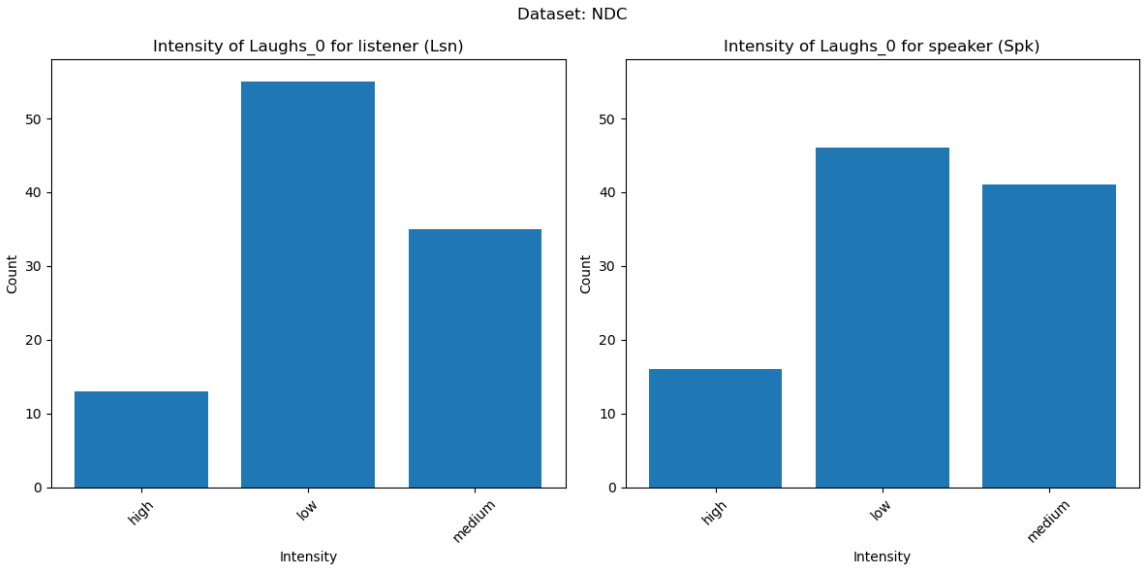


Dataset: IFADV









In summary, this analysis provides a better understanding of the distribution, occurrence patterns, and intensity levels of smiles and laughs in each dataset, taking into account the roles of the participants. This information will be valuable to inform the development of machine learning models aimed at realistically replicating smiles and laughs in conversations, taking context and social interactions into account.

2-Mimicry (inter)

We look at the capacity of someone to mimic someone else expression in an interaction.

By definition, an event is mimicry when the same expression is replicated. For the purpose of this study, and to analyse the influence of two expressions on each other (smiles and laughs), the definition is extended to copying different expressions as well, i.e., smiles mimicking laughs and vice versa.

We thus calculate the mimicry of each expression at a given intensity on another, for spk on lsn and vice versa for a delta = 1000 ms (automatic mimicry). We look at person B mimicking A (A/B) and vice versa between pairs files in order to see if it has an impact on the result.

Explanation of delta :

Automatic (also called unconscious or reflexive) mimicry occurs unconsciously and quickly, usually within 1000 ms of seeing a facial expression, while effortful mimicry (also called intentional or voluntary) involves a slower conscious effort to reproduce the facial expression of another.

- Automatic mimicry occurs without awareness and is considered an innate aspect of human nature that plays a key role in social interactions.
- Effortful facial mimicry, or the deliberate act of copying another person's facial expression (for example, being told explicitly to mimic a photograph), is primarily distinguished from automatic mimicry as a conscious process (versus unconscious).

In [ ]:
```
delta = 1000
```

a) B mimicking A

Let's start with person B mimicking A in an interaction (so all second files in the pairs files mimicking first files in database order):

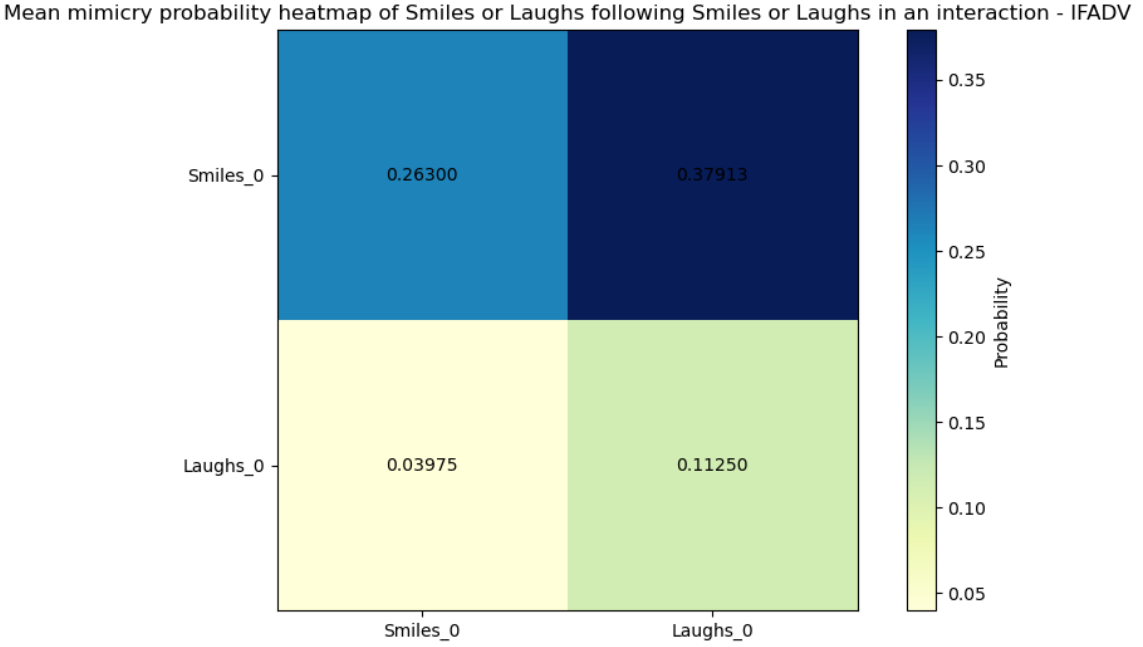
In [ ]:
```
# A/B -> person B mimics person A
mimic_choice = 'A/B'
```

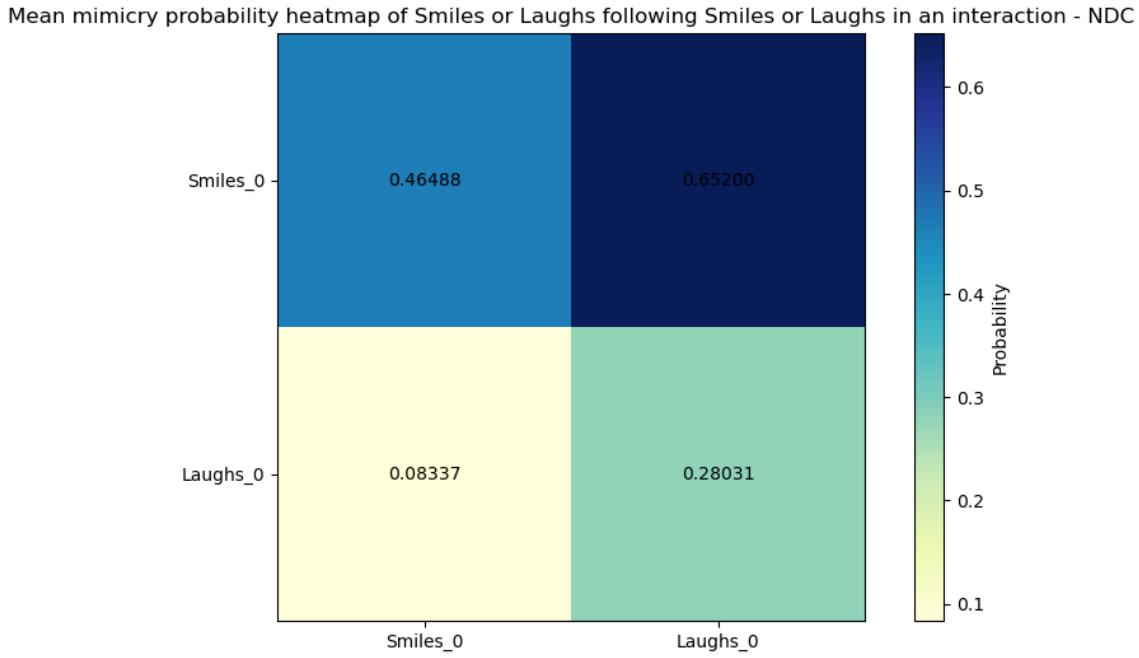
Mimicry Smiles vs Laughs (all entities) for each database

Extraction of the stats :

In [ ]:
```
probabilities_matrix = []
moyenne_prob_interaction = 0
for i, database in enumerate(databases_name):
    if database==databases_pairs[i].replace('_pairs','').upper():
                databases_list=databases_pair_paths[databases_pairs[i]]
    # Create a 2x2 matrix with zeros
    probabilities_matrix = np.zeros((len(expressions), len(expressions)))
    for j in range(len(expressions)):
        expression_choiceA = expressions[j]
        for k in range(len(expressions)):
            expression_choiceB = expressions[k]
            list_mimicry_SL = give_mimicry_folder2(databases_list, database.lowe
            for item in list_mimicry_SL:
                 moyenne_prob_interaction += item[1]
            moyenne_prob_interaction = moyenne_prob_interaction/len(list_mimicry
            probabilities_matrix[j, k] = moyenne_prob_interaction
            moyenne_prob_interaction = 0

    # Create a new figure for each database
    fig, ax = plt.subplots(figsize=(10, 6))
```

```
    # Create the heatmap plot
    im = ax.imshow(probabilities_matrix, cmap='YlGnBu', interpolation='nearest')

    # Add the probability values within each square
    for x in range(len(expressions)):
        for y in range(len(expressions)):
            text_color = 'black'  # Default text color
            ax.text(y, x, f'{probabilities_matrix[x, y]:.5f}', ha='center', va='

    # Customize the plot
    fig.colorbar(im, ax=ax, label='Probability')
    ax.set_xticks(range(len(expressions)))
    ax.set_yticks(range(len(expressions)))
    ax.set_xticklabels(expressions)
    ax.set_yticklabels(expressions)
    ax.set_title(f'Mean mimicry probability heatmap of Smiles or Laughs followin

    # Show the plot
    plt.show()
```





Analysis of the stats:

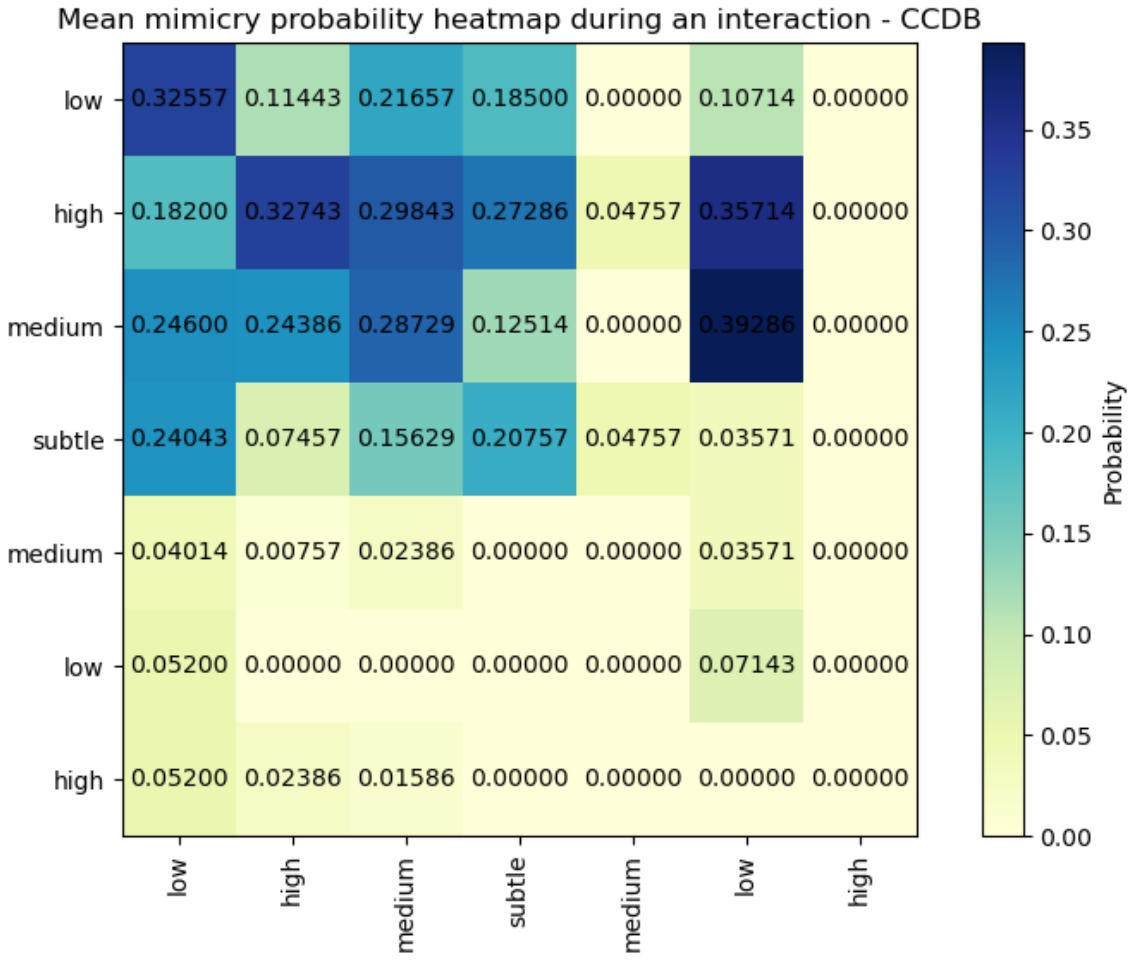
For A\B: We first observe that, in general, laughs are mostly mimicked by laughs while smiles are mimicked by smiles and laughs.

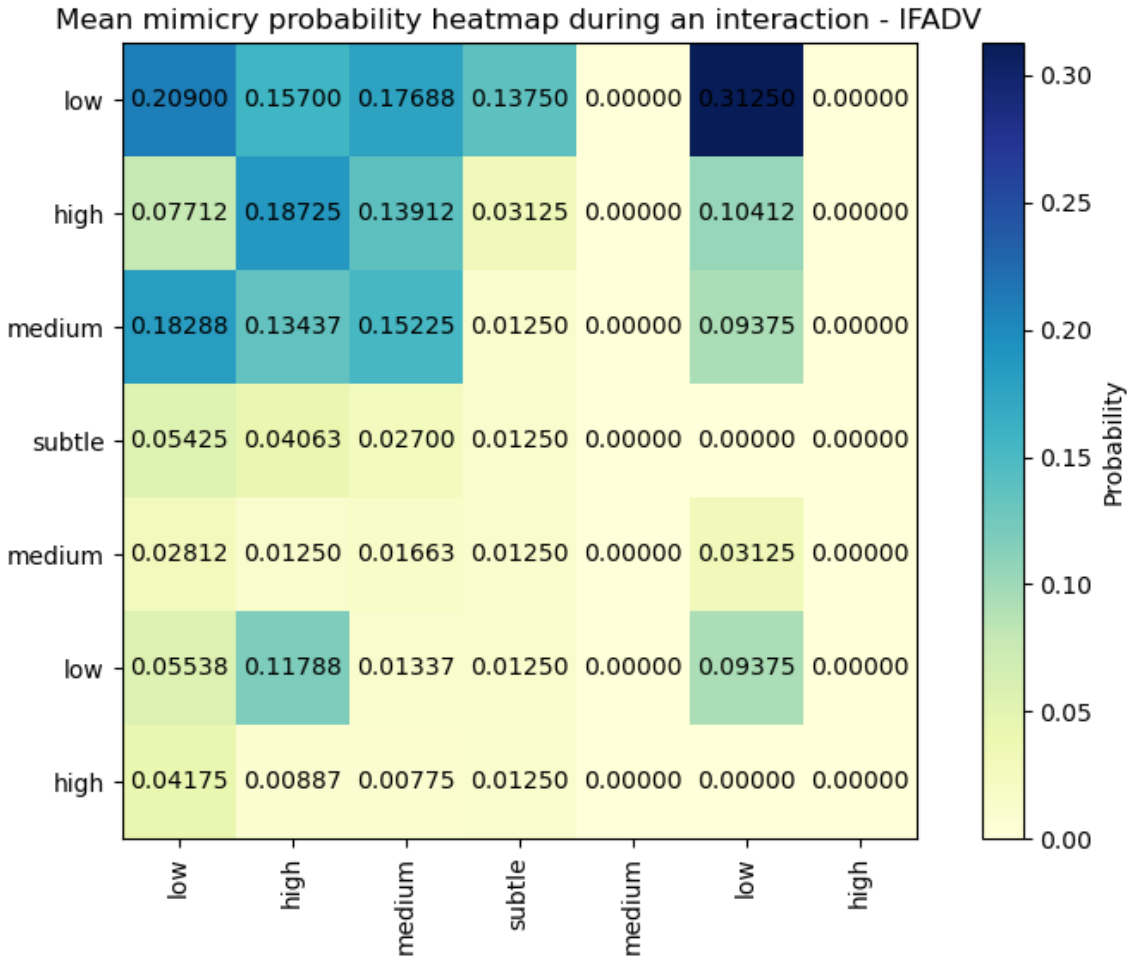
Mimicry Smiles vs Laughs (per entity) for each database

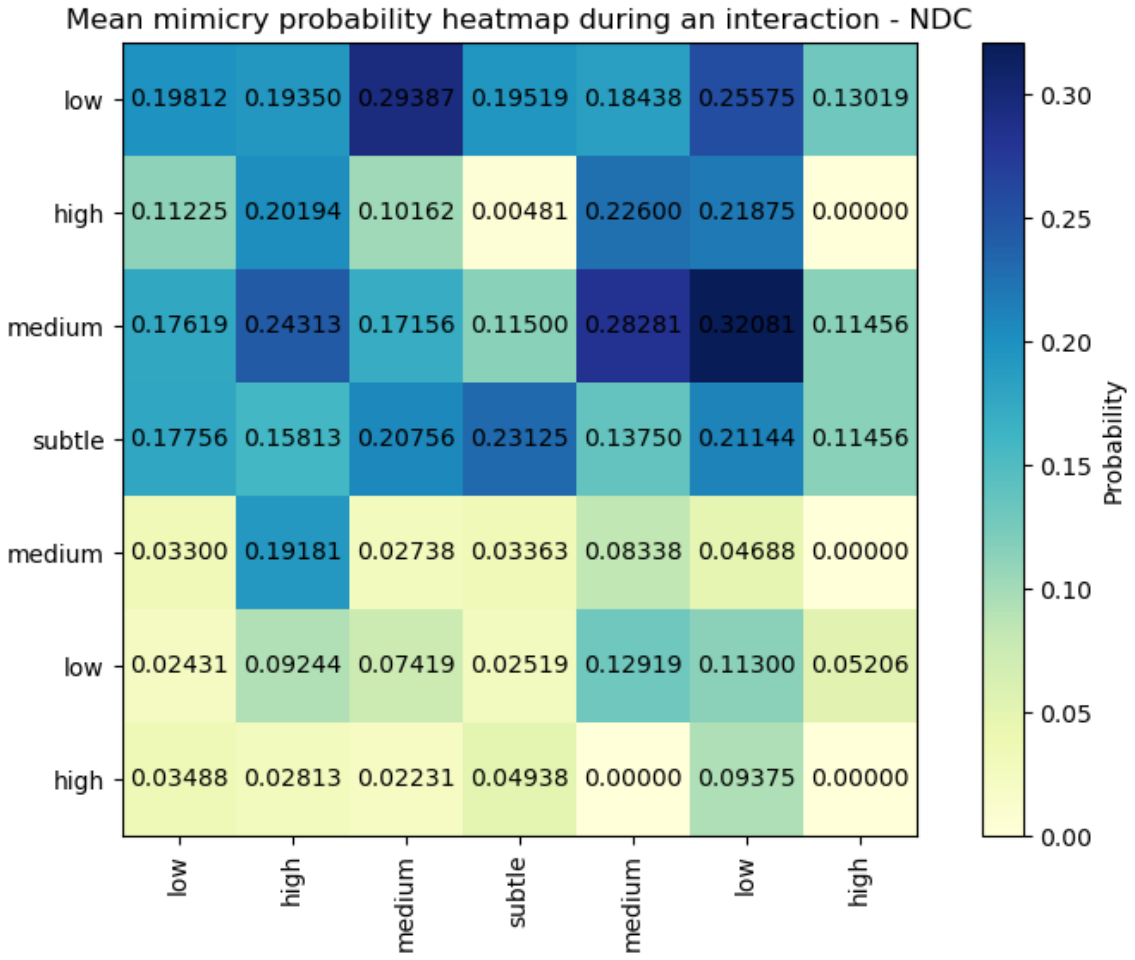
Extraction of the stats:

In [ ]:
```
probabilities_matrix_2 = []
moyenne_prob_interaction_2 = 0

for i, database in enumerate(databases_name):
    if database == databases_pairs[i].replace('_pairs', '').upper():
        databases_list = databases_pair_paths[databases_pairs[i]]

    # Create a matrix with zeros
    num_entities_A = sum(len(tier_lists[expression_choiceA]) for expression_choi
    num_entities_B = sum(len(tier_lists[expression_choiceB]) for expression_choi
    probabilities_matrix_2 = np.zeros((num_entities_A, num_entities_B))

    # Track the current row and column index
    current_row = 0
    current_col = 0

    for j in range(len(expressions)):
        expression_choiceA = expressions[j]
        entitiesA = tier_lists[expression_choiceA]

        for entityA in entitiesA:
            for k in range(len(expressions)):
                expression_choiceB = expressions[k]
                entitiesB = tier_lists[expression_choiceB]

                for entityB in entitiesB:
                    list_mimicry_SL_entity = give_mimicry_folder2(databases_list
```

```
                    for item in list_mimicry_SL_entity:
                        moyenne_prob_interaction_2 += item[1]

                    moyenne_prob_interaction_2 = moyenne_prob_interaction_2 / le
                    probabilities_matrix_2[current_row, current_col] = moyenne_p
                    moyenne_prob_interaction_2 = 0

                    current_col += 1

            current_row += 1
            current_col = 0

    # Create a new figure for each database
    fig, ax = plt.subplots(figsize=(10, 6))

    # Create the heatmap plot
    im = ax.imshow(probabilities_matrix_2, cmap='YlGnBu', interpolation='nearest

    # Add the probability values within each square
    for x in range(num_entities_A):
        for y in range(num_entities_B):
            text_color = 'black'  # Default text color
            ax.text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='center', va

    # Customize the plot
    fig.colorbar(im, ax=ax, label='Probability')

    # Set the x and y ticks and labels based on the number of entities
    xticks = np.arange(num_entities_B)
    yticks = np.arange(num_entities_A)

    ax.set_xticks(xticks)
    ax.set_yticks(yticks)

    # Set the x and y tick labels based on the expressions and entities
    xtick_labels = []
    ytick_labels = []

    for expression_choiceB in expressions:
        entitiesB = tier_lists[expression_choiceB]
        xtick_labels.extend(entitiesB)

    for expression_choiceA in expressions:
        entitiesA = tier_lists[expression_choiceA]
        ytick_labels.extend(entitiesA)

    ax.set_xticklabels(xtick_labels, rotation=90)
    ax.set_yticklabels(ytick_labels)

    ax.set_title(f'Mean mimicry probability heatmap during an interaction - {dat

    # Show the plot
    plt.show()
```

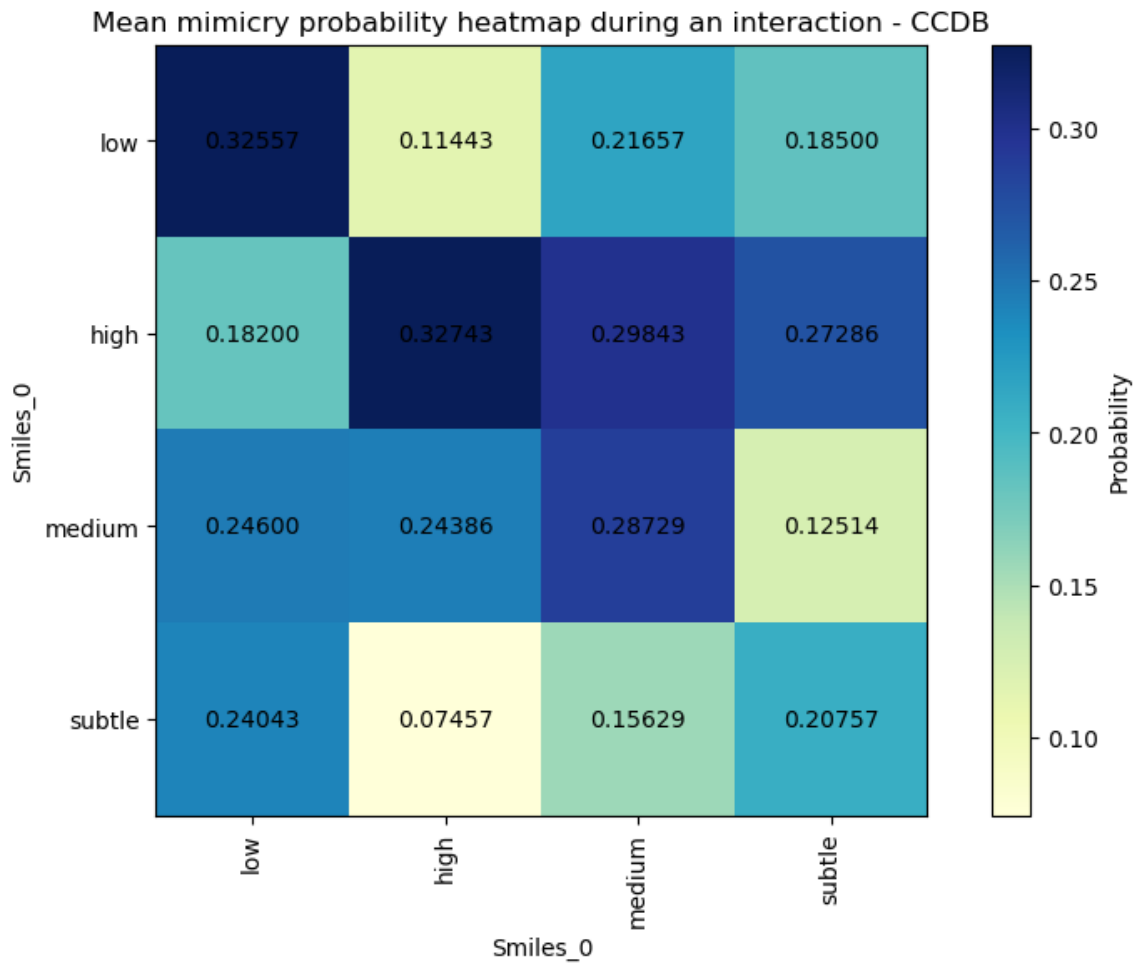

Mean mimicry probability heatmap during an interaction - CCDB
low 0.32557 0.11443 0.21657 0.18500 0.00000 0.10714 0.00000
high 0.18200 0.32743 0.29843 0.27286 0.04757 0.35714 0.00000
medium 0.24600 0.24386 0.28729 0.12514 0.00000 0.39286 0.00000
subtle 0.24043 0.07457 0.15629 0.20757 0.04757 0.03571 0.00000
medium 0.04014 0.00757 0.02386 0.00000 0.00000 0.03571 0.00000
low 0.05200 0.00000 0.00000 0.00000 0.00000 0.07143 0.00000
high 0.05200 0.02386 0.01586 0.00000 0.00000 0.00000 0.00000
low high medium subtle medium low high
Probability
0.35 0.30 0.25 0.20 0.15 0.10 0.05 0.00

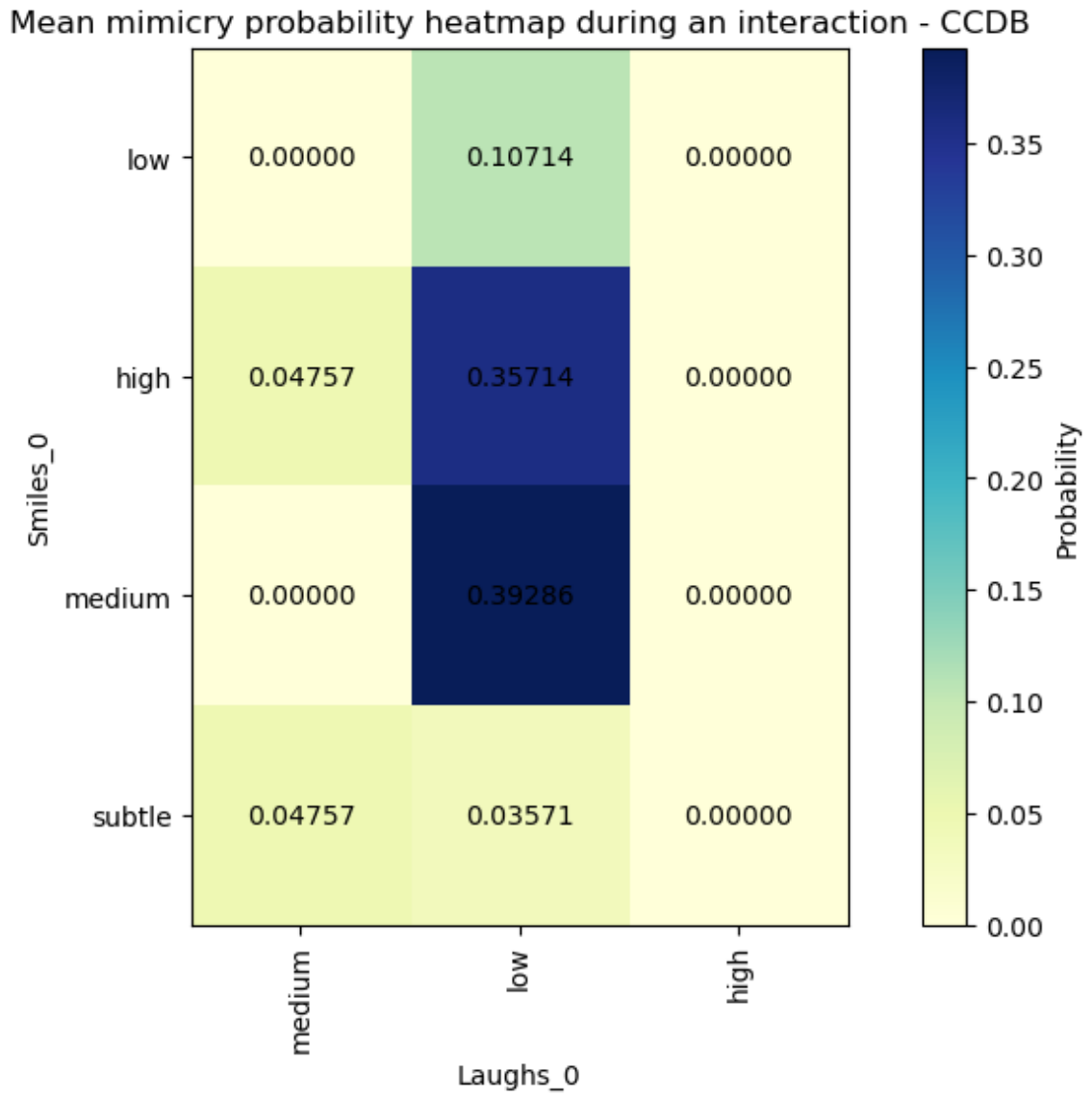

Mean mimicry probability heatmap during an interaction - IFADV
low 0.20900 0.15700 0.17688 0.13750 0.00000 0.31250 0.00000
high 0.07712 0.18725 0.13912 0.03125 0.00000 0.10412 0.00000
medium 0.18288 0.13437 0.15225 0.01250 0.00000 0.09375 0.00000
subtle 0.05425 0.04063 0.02700 0.01250 0.00000 0.00000 0.00000
medium 0.02812 0.01250 0.01663 0.01250 0.00000 0.03125 0.00000
low 0.05538 0.11788 0.01337 0.01250 0.00000 0.09375 0.00000
high 0.04175 0.00887 0.00775 0.01250 0.00000 0.00000 0.00000
low high medium subtle medium low high
Probability
0.30 0.25 0.20 0.15 0.10 0.05 0.00

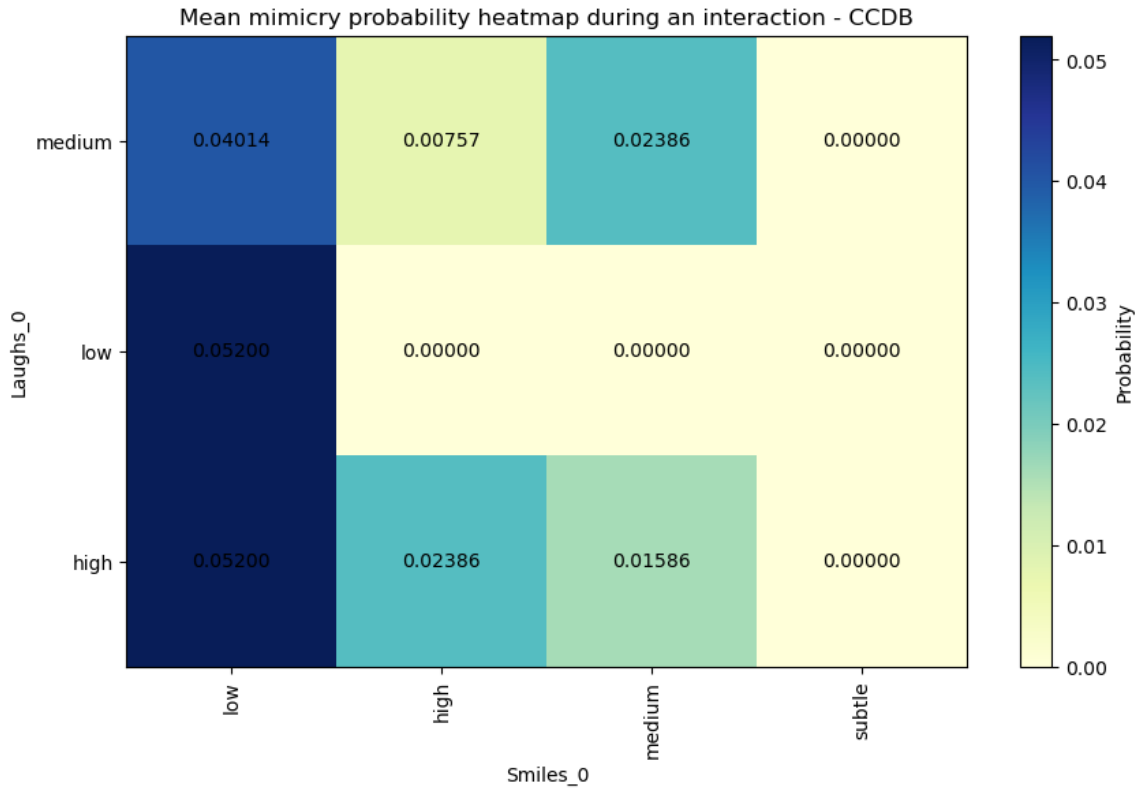


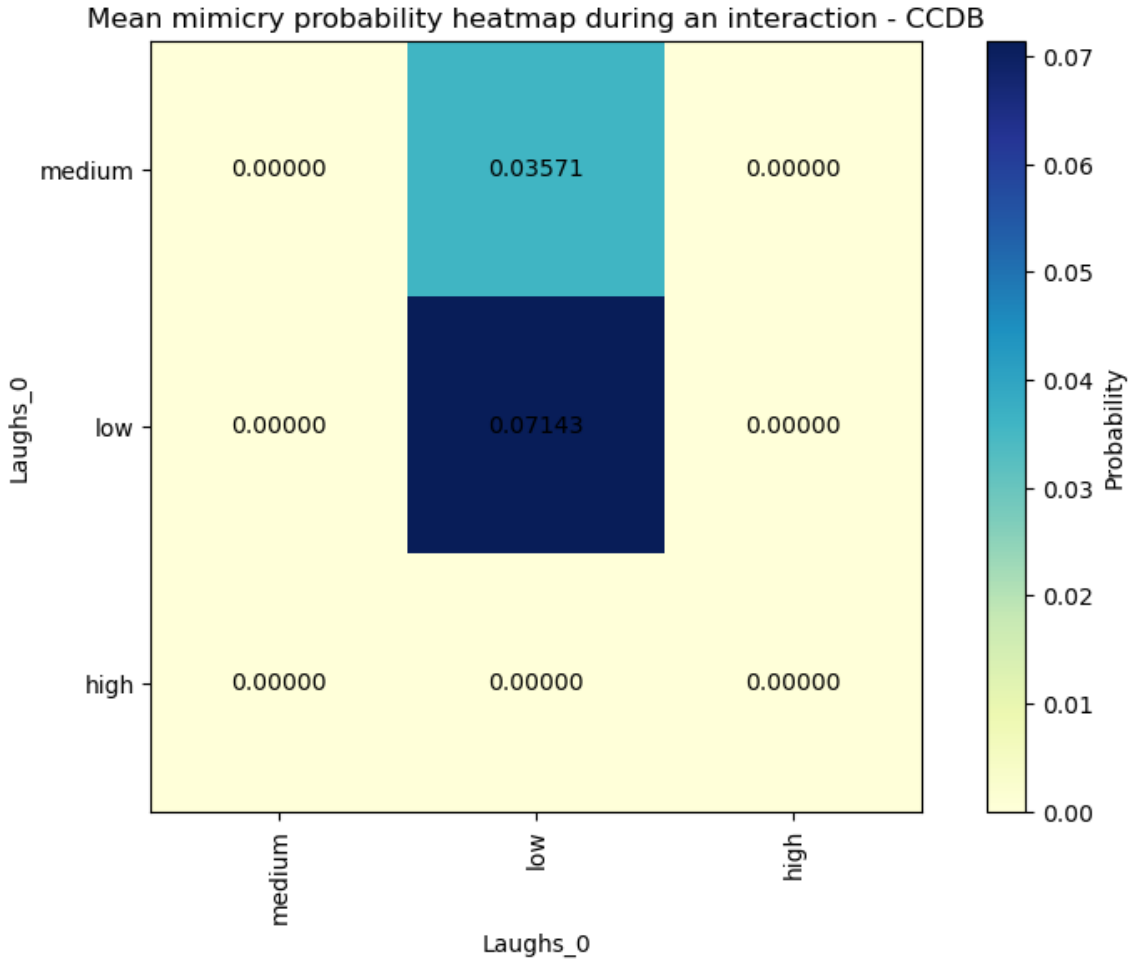
In [ ]:
```
probabilities_matrix_2 = []
moyenne_prob_interaction_2 = 0


for i, database in enumerate(databases_name):
    if database == databases_pairs[i].replace('_pairs', '').upper():
        databases_list = databases_pair_paths[databases_pairs[i]]

    # Create a dictionary to store entities grouped by expressions
    entities_by_expression = {}

    # Group entities by expressions
    for expression_choiceA in expressions:
        entitiesA = tier_lists[expression_choiceA]
        for entityA in entitiesA:
            if expression_choiceA not in entities_by_expression:
                entities_by_expression[expression_choiceA] = []
            entities_by_expression[expression_choiceA].append(entityA)

    # Iterate over expression pairs and entities for each database
    for expression_choiceA, entitiesA in entities_by_expression.items():
        for expression_choiceB, entitiesB in entities_by_expression.items():
            # Create a matrix with zeros
            num_entities_A = len(entitiesA)
            num_entities_B = len(entitiesB)
            probabilities_matrix_2 = np.zeros((num_entities_A, num_entities_B))

            # Track the current row and column index
            current_row = 0
            current_col = 0
```

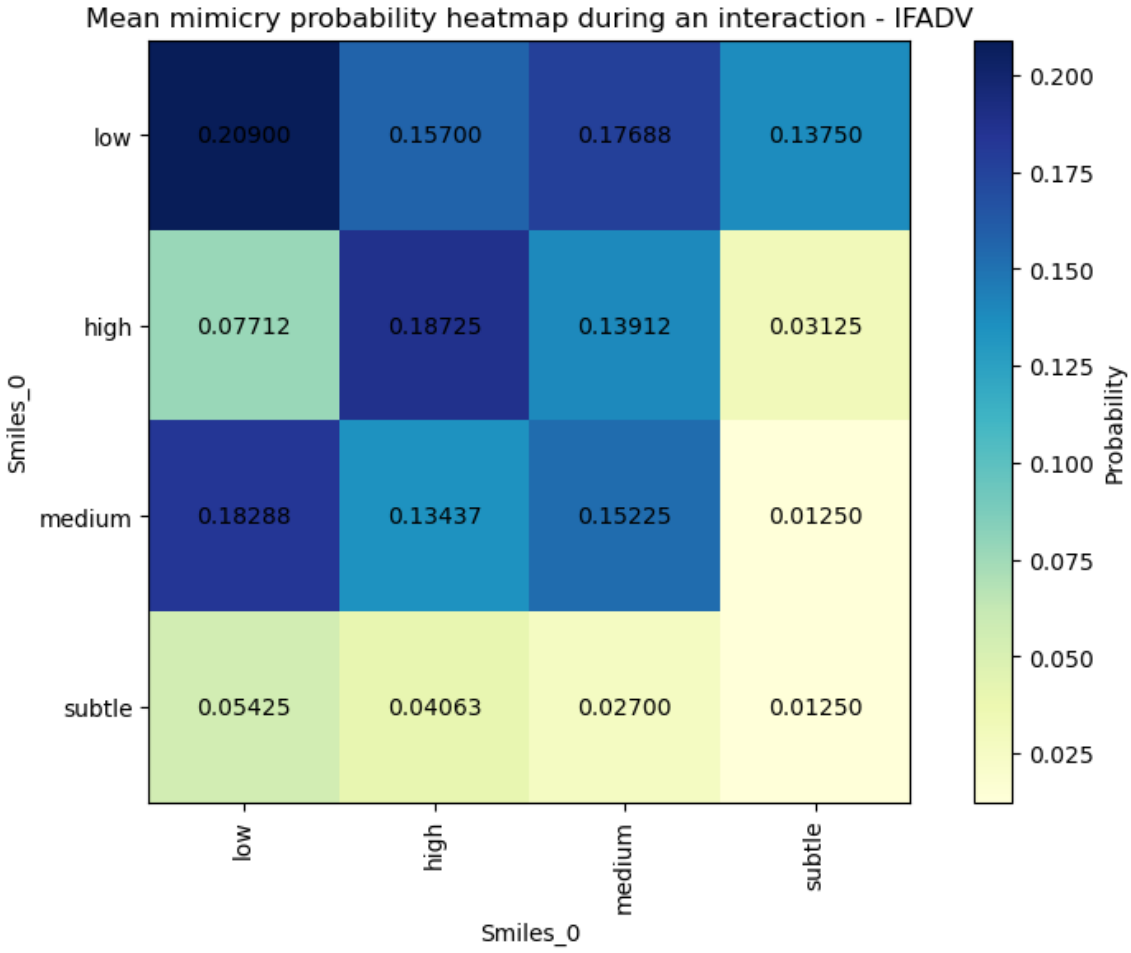
```
            for entityA in entitiesA:
                for entityB in entitiesB:
                    list_mimicry_SL_entity = give_mimicry_folder2(databases_list

                    for item in list_mimicry_SL_entity:
                        moyenne_prob_interaction_2 += item[1]

                    moyenne_prob_interaction_2 = moyenne_prob_interaction_2 / le
                    probabilities_matrix_2[current_row, current_col] = moyenne_p
                    moyenne_prob_interaction_2 = 0

                    current_col += 1

                current_row += 1
                current_col = 0

            # Create a new figure for each expression pair and database
            fig, ax = plt.subplots(figsize=(10, 6))

            # Create the heatmap plot
            im = ax.imshow(probabilities_matrix_2, cmap='YlGnBu', interpolation=

            # Add the probability values within each square
            for x in range(num_entities_A):
                for y in range(num_entities_B):
                    text_color = 'black'
                    ax.text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='cer

            # Customize the plot
            fig.colorbar(im, ax=ax, label='Probability')

            # Set the x and y ticks and labels based on the number of entities
            xticks = np.arange(num_entities_B)
            yticks = np.arange(num_entities_A)

            ax.set_xticks(xticks)
            ax.set_yticks(yticks)

            # Set the x and y tick labels based on the entities
            ax.set_xticklabels(entitiesB, rotation=90)
            ax.set_yticklabels(entitiesA)

            ax.set_xlabel(expression_choiceB)
            ax.set_ylabel(expression_choiceA)
            ax.set_title(f'Mean mimicry probability heatmap during an interactio

            # Show the plot
            plt.show()
```

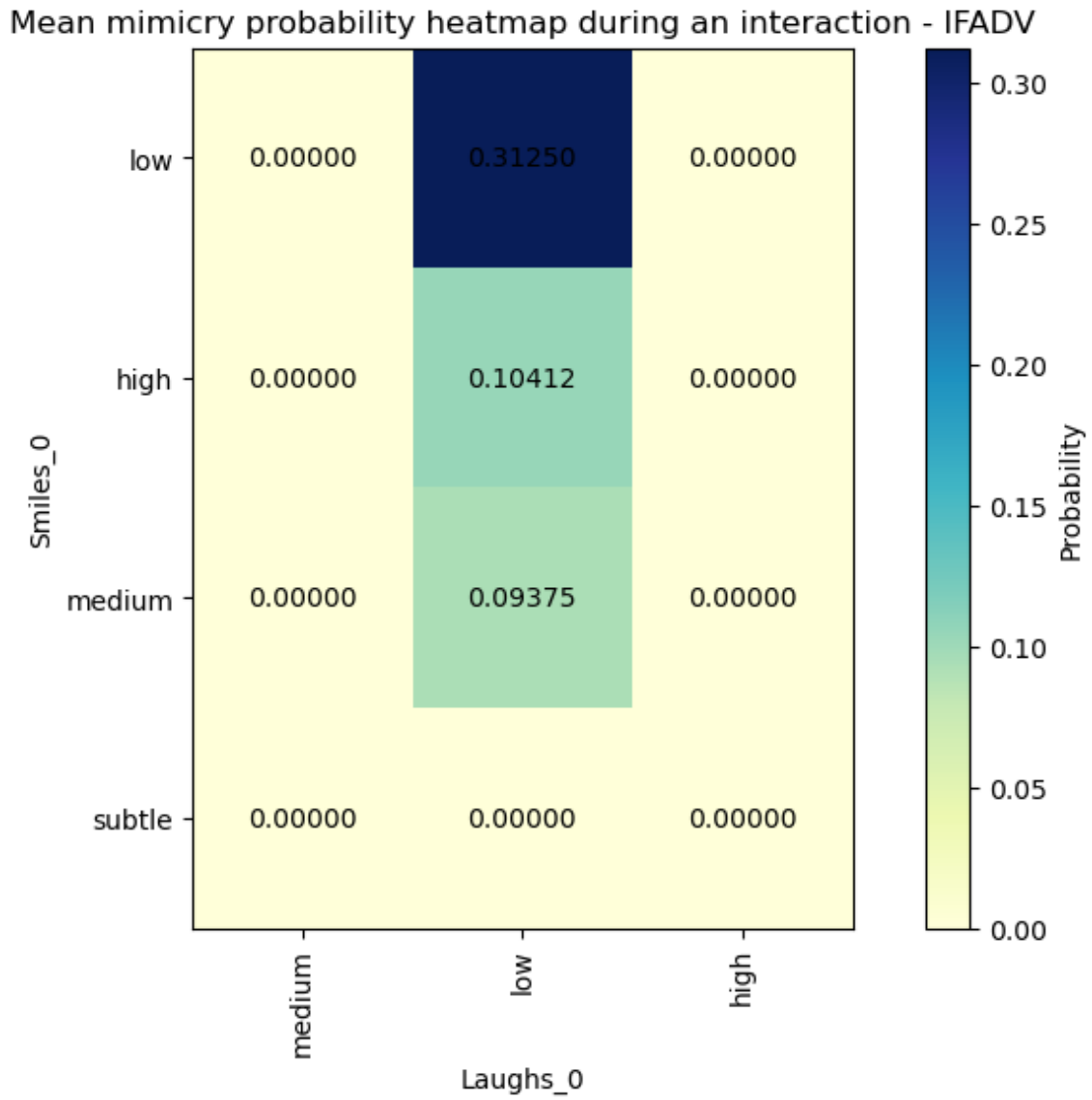

Mean mimicry probability heatmap during an interaction - CCDB
low
high
medium
subtle
Smiles_0
0.32557
0.11443
0.21657
0.18500
0.18200
0.32743
0.29843
0.27286
0.24600
0.24386
0.28729
0.12514
0.24043
0.07457
0.15629
0.20757
low
high
medium
subtle
Smiles_0
0.30
0.25
0.20
0.15
0.10
Probability

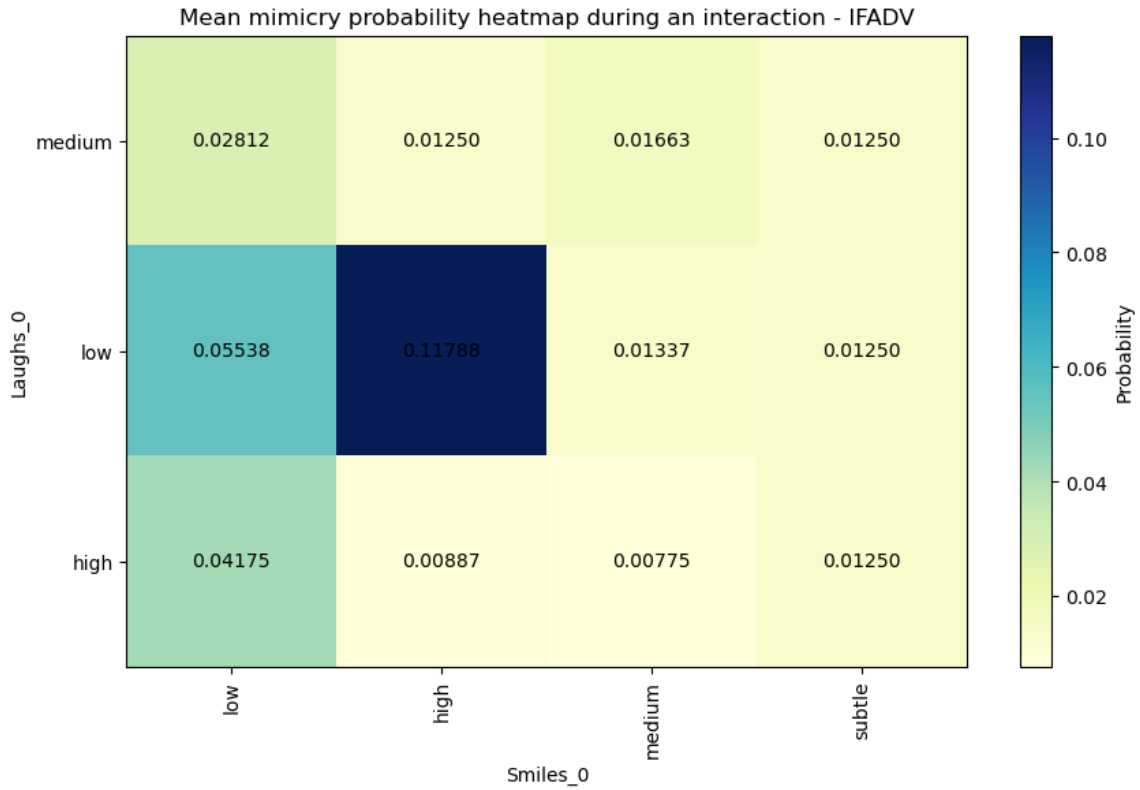

Mean mimicry probability heatmap during an interaction - CCDB
low
high
medium
subtle
Smiles_0
0.00000
0.10714
0.00000
0.04757
0.35714
0.00000
0.00000
0.39286
0.00000
0.04757
0.03571
0.00000
medium
low
high
Laughs_0
Probability
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

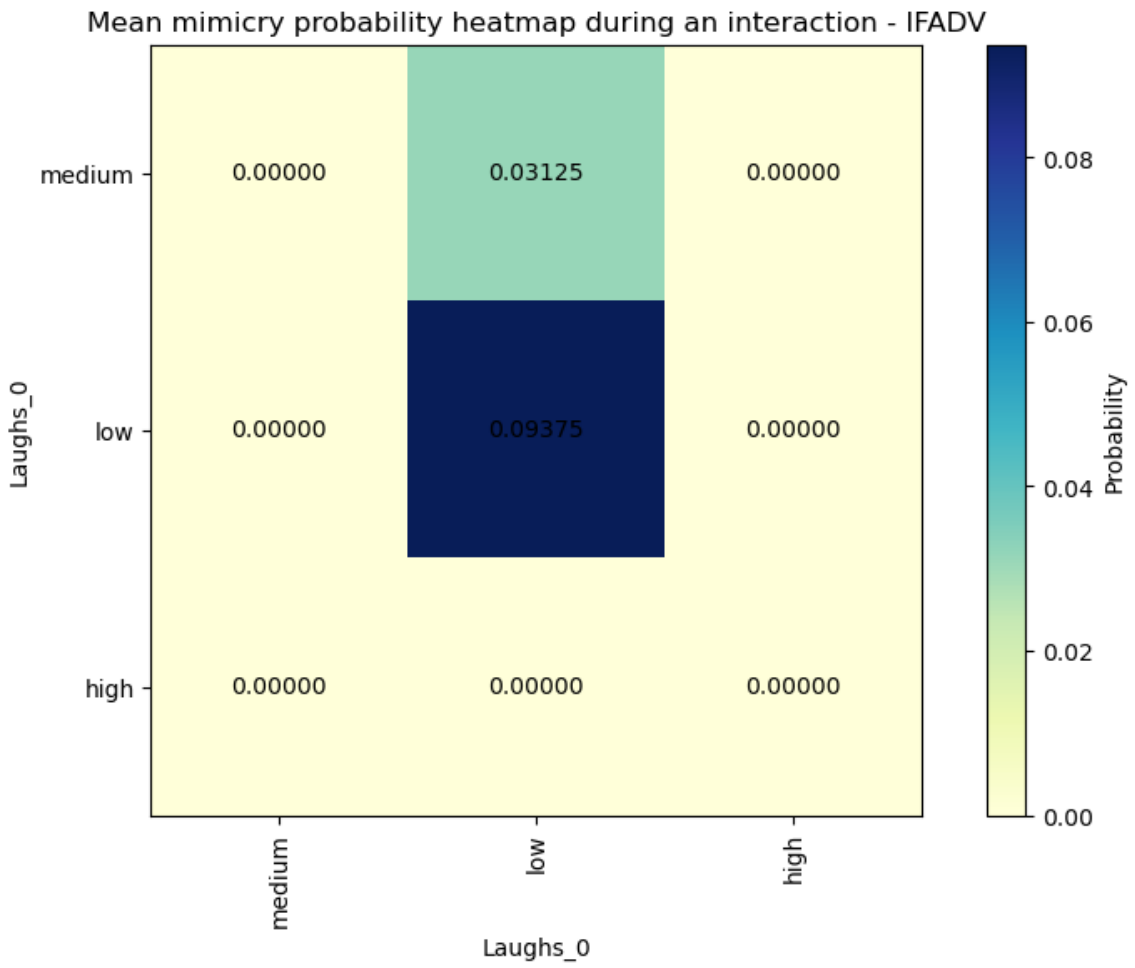

Mean mimicry probability heatmap during an interaction - CCDB
medium
low
high
Laughs_0
0.04014
0.00757
0.02386
0.00000
0.05200
0.00000
0.00000
0.00000
0.05200
0.02386
0.01586
0.00000
low
high
medium
subtle
Smiles_0
Probability
0.05
0.04
0.03
0.02
0.01
0.00

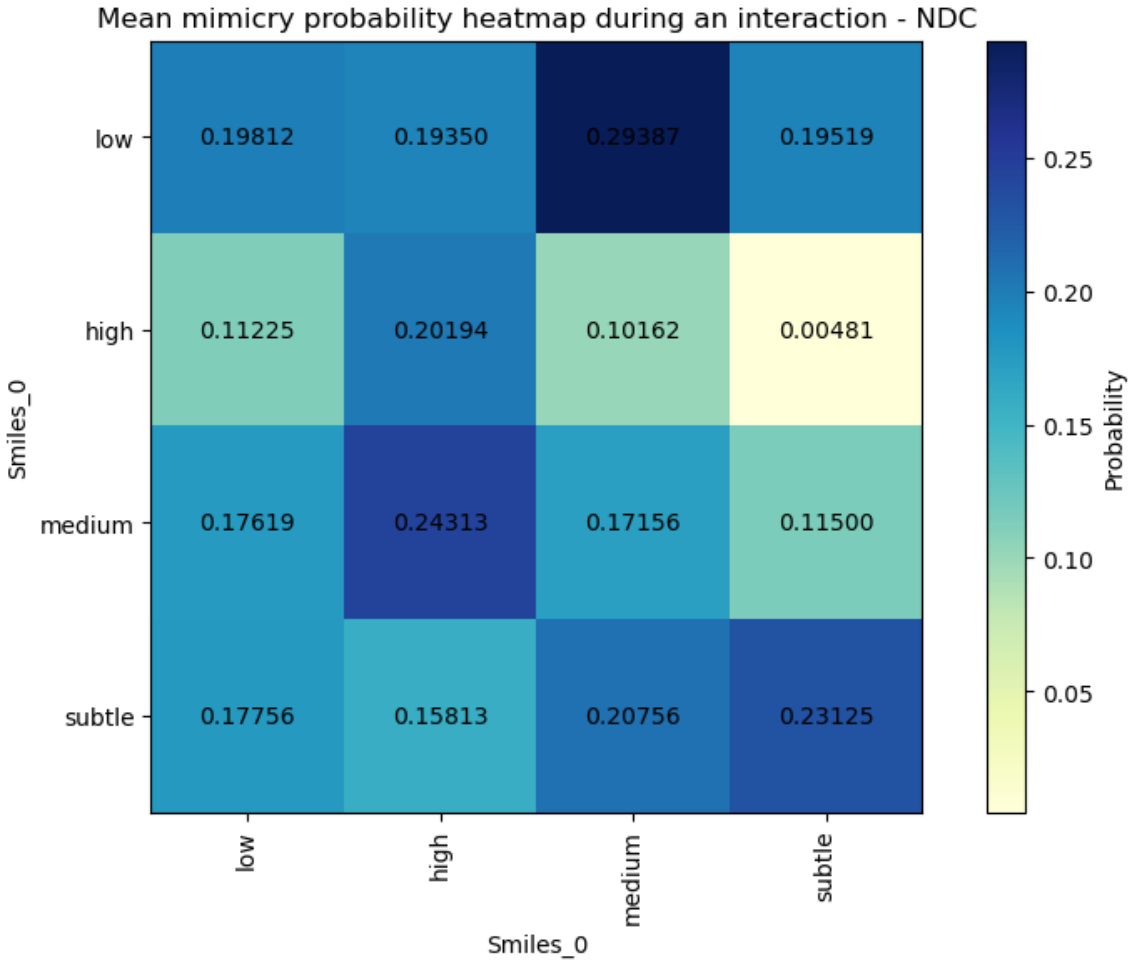

Mean mimicry probability heatmap during an interaction - CCDB
medium
low
high
Laughs_0
0.00000
0.03571
0.00000
0.00000
0.07143
0.00000
0.00000
0.00000
0.00000
medium
low
high
Laughs_0
Probability
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

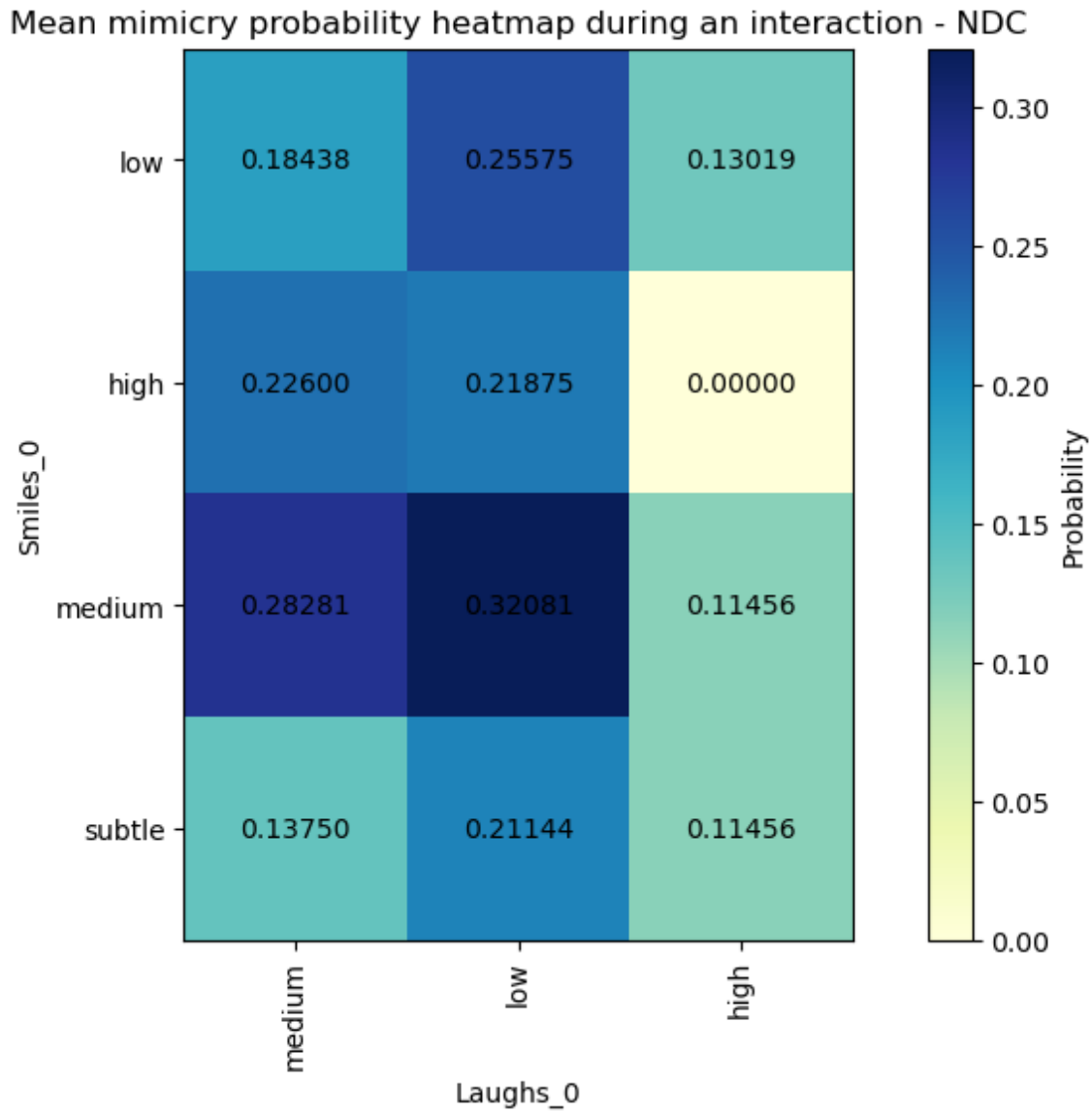

Mean mimicry probability heatmap during an interaction - IFADV
low
high
medium
subtle
Smiles_0
0.20900
0.15700
0.17688
0.13750
0.07712
0.18725
0.13912
0.03125
0.18288
0.13437
0.15225
0.01250
0.05425
0.04063
0.02700
0.01250
low
high
medium
subtle
Smiles_0
Probability
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025

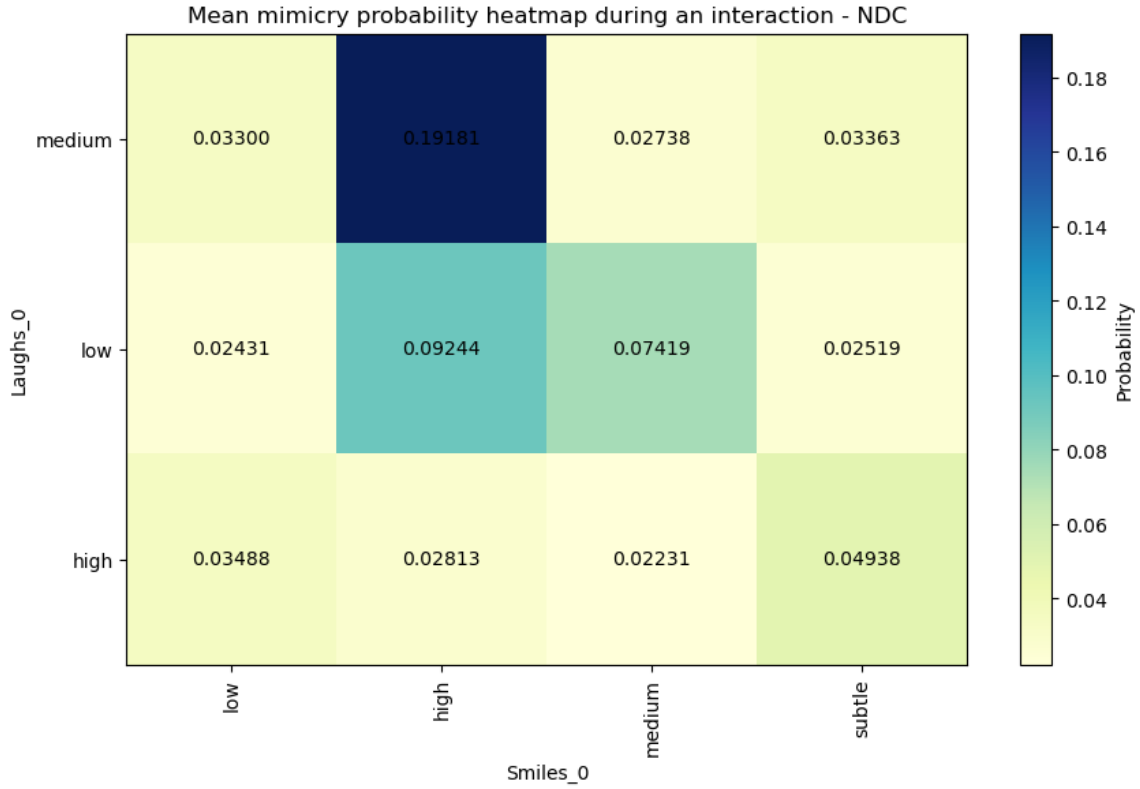

Mean mimicry probability heatmap during an interaction - IFADV
low
high
medium
subtle
Smiles_0
0.00000
0.31250
0.00000
0.00000
0.10412
0.00000
0.00000
0.09375
0.00000
0.00000
0.00000
0.00000
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Probability
medium
low
high
Laughs_0

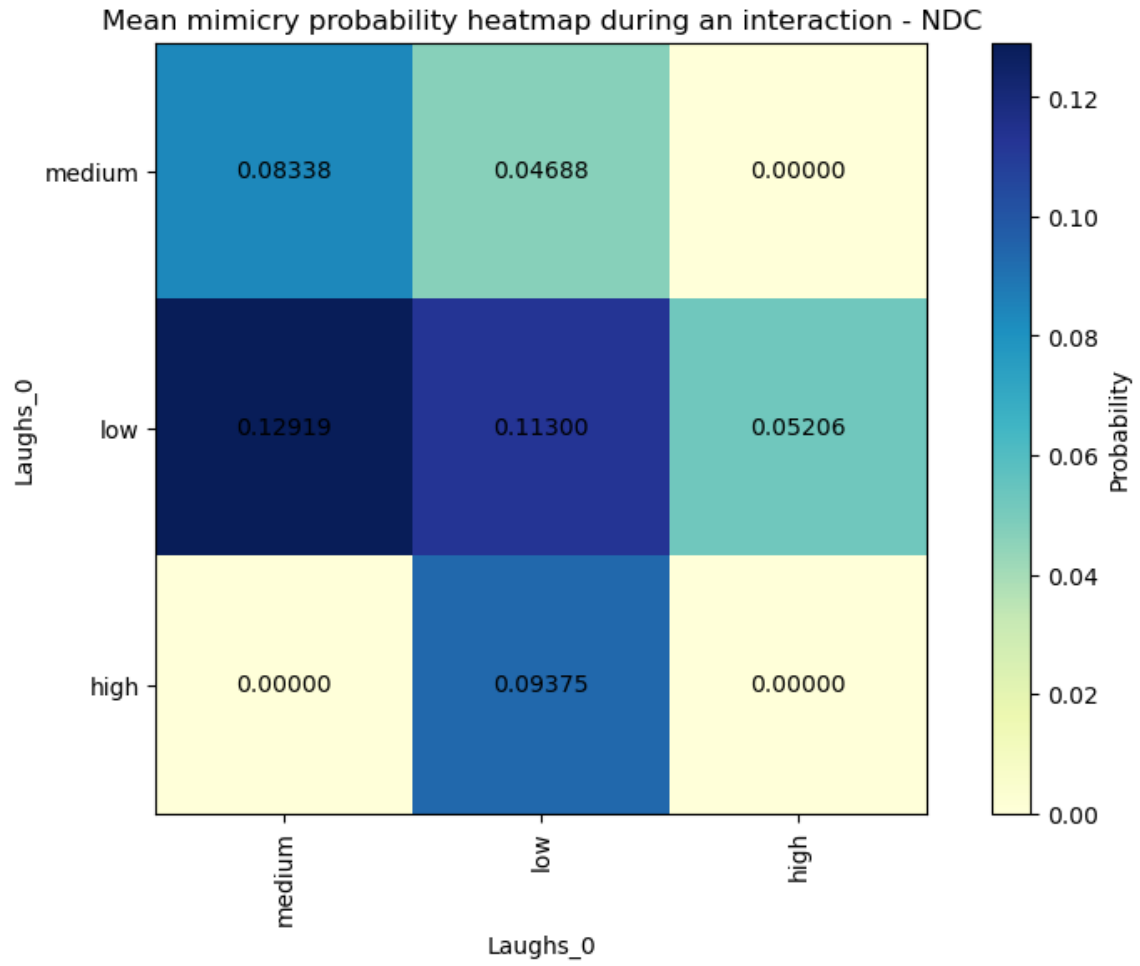

Mean mimicry probability heatmap during an interaction - IFADV
medium
low
high
Laughs_0
0.02812
0.01250
0.01663
0.01250
0.05538
0.11788
0.01337
0.01250
0.04175
0.00887
0.00775
0.01250
0.10
0.08
0.06
0.04
0.02
Probability
low
high
medium
subtle
Smiles_0


Mean mimicry probability heatmap during an interaction - IFADV
medium
low
high
Laughs_0
0.00000
0.03125
0.00000
0.00000
0.09375
0.00000
0.00000
0.00000
0.00000
medium
low
high
Laughs_0
0.08
0.06
0.04
0.02
0.00
Probability


Mean mimicry probability heatmap during an interaction - NDC
low
high
medium
subtle
Smiles_0
0.19812
0.19350
0.29387
0.19519
0.11225
0.20194
0.10162
0.00481
0.17619
0.24313
0.17156
0.11500
0.17756
0.15813
0.20756
0.23125
low
high
medium
subtle
Smiles_0
0.25
0.20
0.15
0.10
0.05
Probability


Mean mimicry probability heatmap during an interaction - NDC
low
high
medium
subtle
Smiles_0
0.18438
0.25575
0.13019
0.22600
0.21875
0.00000
0.28281
0.32081
0.11456
0.13750
0.21144
0.11456
medium
low
high
Laughs_0
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Probability

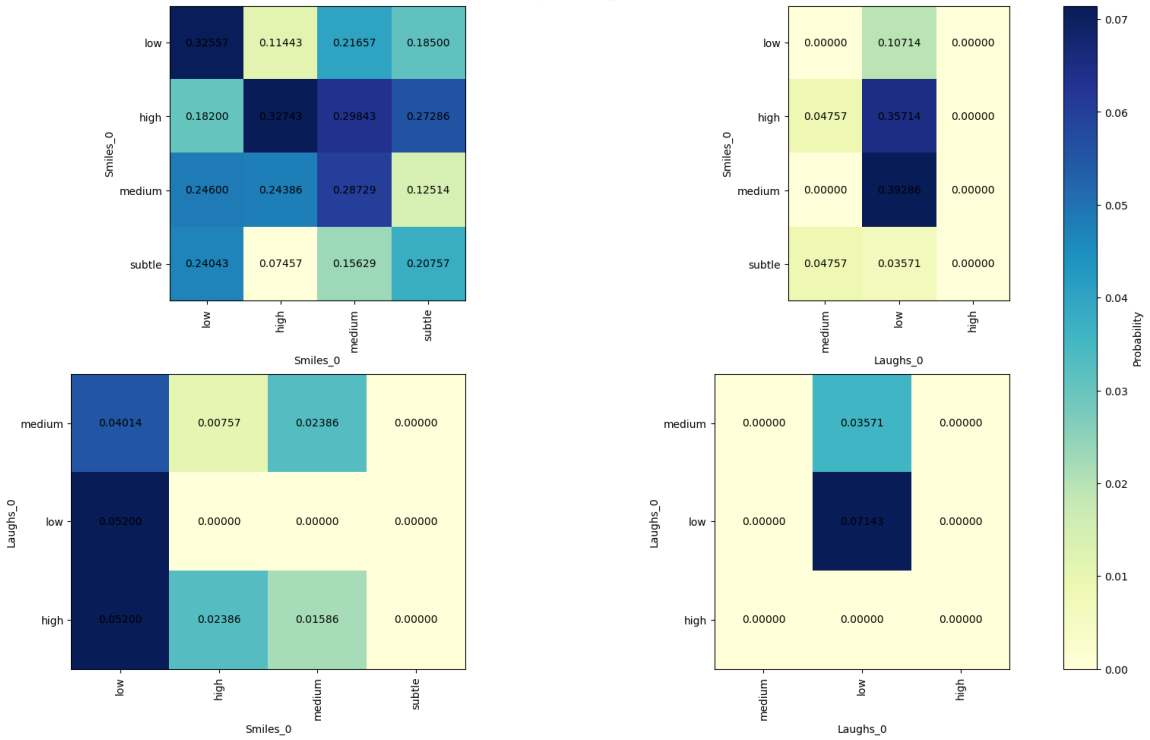

Mean mimicry probability heatmap during an interaction - NDC
medium
low
high
Laughs_0
0.03300
0.19181
0.02738
0.03363
0.02431
0.09244
0.07419
0.02519
0.03488
0.02813
0.02231
0.04938
low
high
medium
subtle
Smiles_0
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
Probability



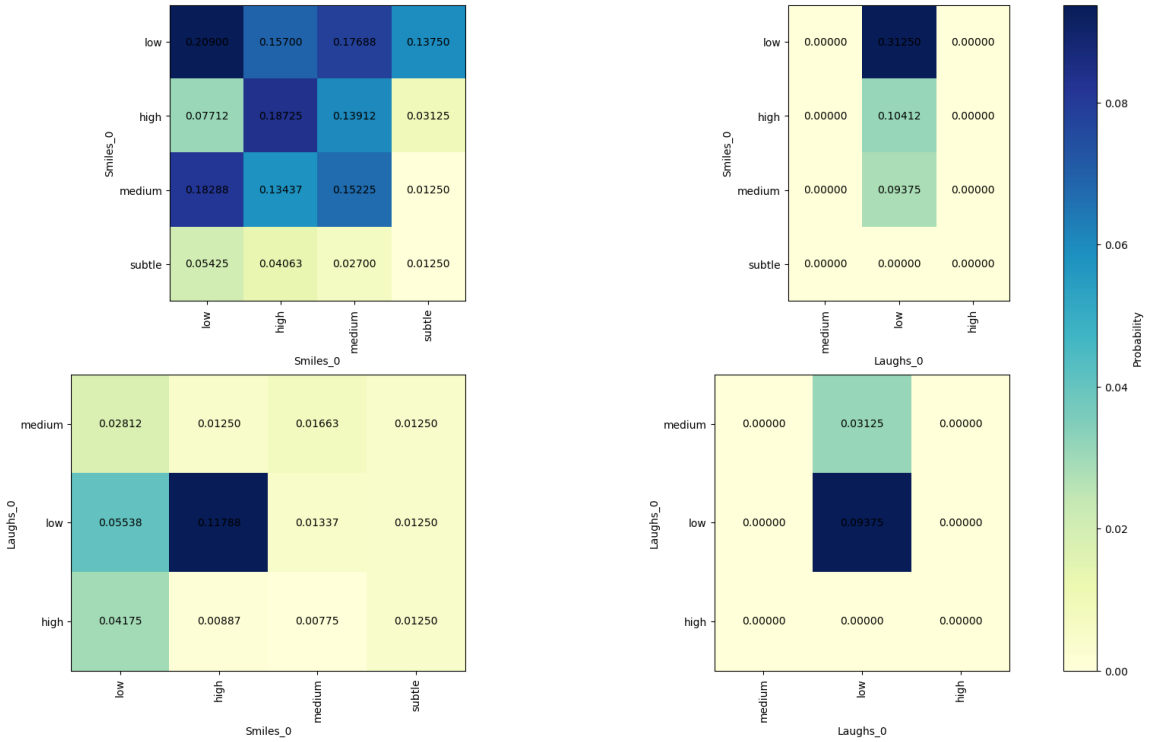
In [ ]:
```
probabilities_matrix_2 = []
moyenne_prob_interaction_2 = 0

# Iterate over each database
for i, database in enumerate(databases_name):
    if database == databases_pairs[i].replace('_pairs', '').upper():
        databases_list = databases_pair_paths[databases_pairs[i]]

    # Create a new figure and axes
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 10), constrained_layc

    # Create a new dictionary to store entities grouped by expressions for each
    entities_by_expression = {}

    # Group entities by expressions
    for expression_choiceA in expressions:
        entitiesA = tier_lists[expression_choiceA]
        for entityA in entitiesA:
            if expression_choiceA not in entities_by_expression:
                entities_by_expression[expression_choiceA] = []
            entities_by_expression[expression_choiceA].append(entityA)

    # Iterate over expression pairs and entities for each database
    for j, (expression_choiceA, entitiesA) in enumerate(entities_by_expression.i
        for k, (expression_choiceB, entitiesB) in enumerate(entities_by_expressi
            # Create a matrix with zeros
            num_entities_A = len(entitiesA)
            num_entities_B = len(entitiesB)
            probabilities_matrix_2 = np.zeros((num_entities_A, num_entities_B))
```

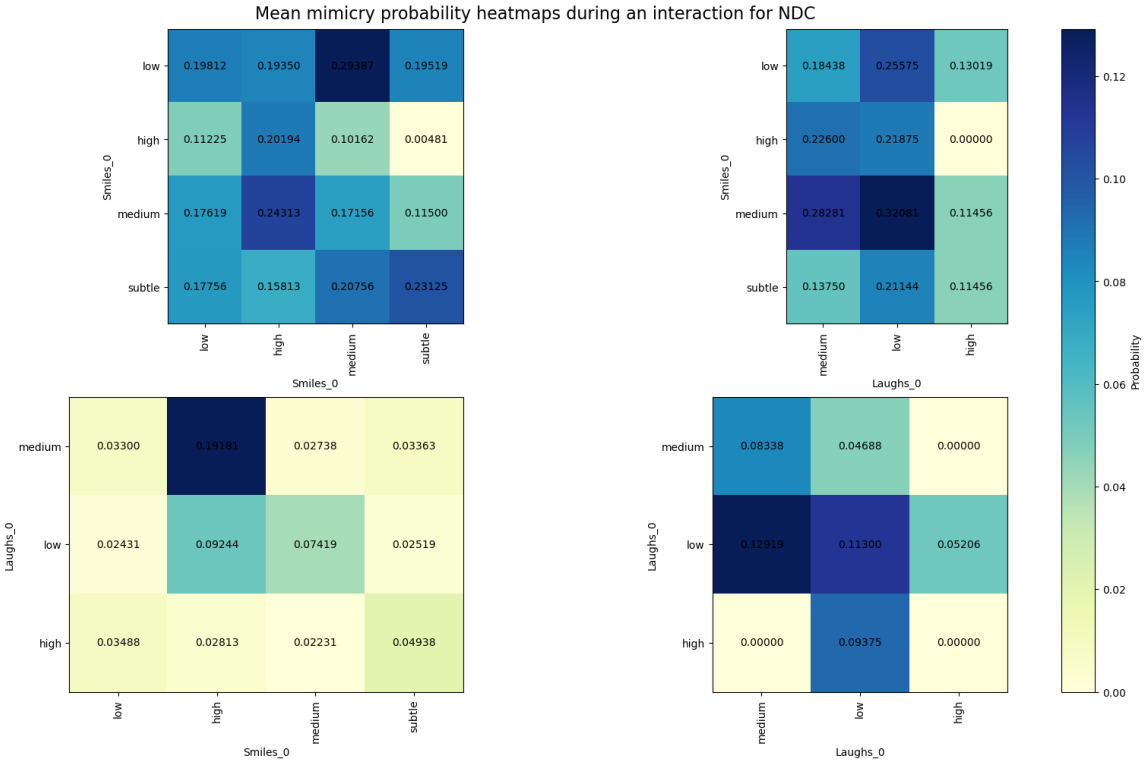
```
            # Track the current row and column index
            current_row = 0
            current_col = 0

            for entityA in entitiesA:
                for entityB in entitiesB:
                    list_mimicry_SL_entity = give_mimicry_folder2(databases_list

                    for item in list_mimicry_SL_entity:
                        moyenne_prob_interaction_2 += item[1]

                    moyenne_prob_interaction_2 = moyenne_prob_interaction_2 / le
                    probabilities_matrix_2[current_row, current_col] = moyenne_p
                    moyenne_prob_interaction_2 = 0

                    current_col += 1

                current_row += 1
                current_col = 0

            # Select the appropriate subplot for each heatmap
            ax = axs[j, k]

            # Create the heatmap plot
            im = ax.imshow(probabilities_matrix_2, cmap='YlGnBu', interpolation=

            # Add the probability values within each square
            for x in range(num_entities_A):
                for y in range(num_entities_B):
                    text_color = 'black'
                    ax.text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='cer

            # Customize the plot
            ax.set_xticks(np.arange(num_entities_B))
            ax.set_xticklabels(entitiesB, rotation=90)
            ax.set_yticks(np.arange(num_entities_A))
            ax.set_yticklabels(entitiesA)
            ax.set_xlabel(expression_choiceB)
            ax.set_ylabel(expression_choiceA)

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im, ax=axs, label='Probability')

    # Add a common title for the heatmaps of each database
    fig.suptitle(f'Mean mimicry probability heatmaps during an interaction for {

    # Show the plot
    plt.show()
```

Mean mimicry probability heatmaps during an interaction for CCDB



Mean mimicry probability heatmaps during an interaction for IFADV





Analysis of the stats:

For A/B:

- For CCDB:
  - Smiles mimicking smiles are of the same intensity or higher
  - Laughs mimicking smiles have a lower intensity
  - Smiles mimicking laughs are mostly of a lower intensity but we have really small probabilities (lack of data?)
  - Laughs mimicking laughs have the same intensity or lower
- For IFADV:
  - Smiles mimicking smiles are of the same intensity
  - Laughs mimicking smiles are mostly mimicked by low laughs
  - Smiles mimicking laughs are mostly of the same intensity or higher
  - Laughs mimicking laughs have the same intensity or lower
- For NDC:
  - Smiles mimicked by smiles are of the same intensity or higher
  - Laughs mimicking smiles have the same or lower intensity
  - Smiles mimicking laughs are mostly higher in intensity
  - Laughs mimicking laughs have the same intensity

Lower levels of laughs mimicking smiles and higher levels of smiles mimicking laughs are in favor of the smile-laugh continuum theory mentioned earlier with smiles being on the low arousal side and laughs on the high side of a common arousal level scale for both S&L.

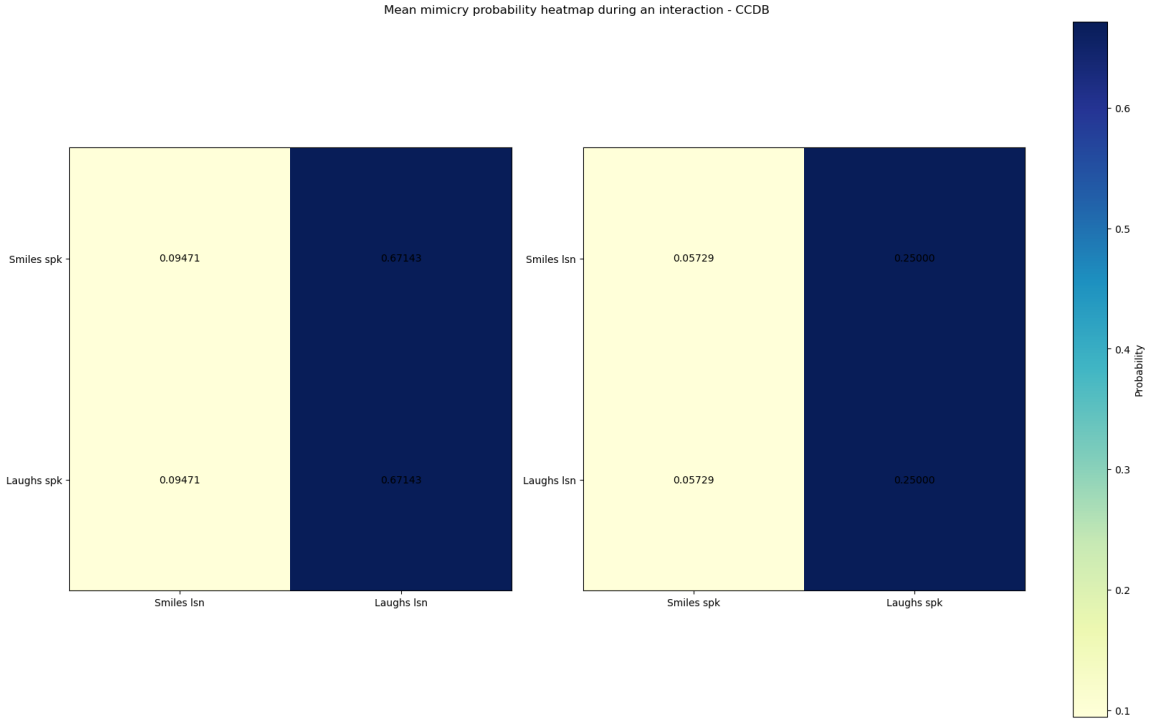
Filtered by role

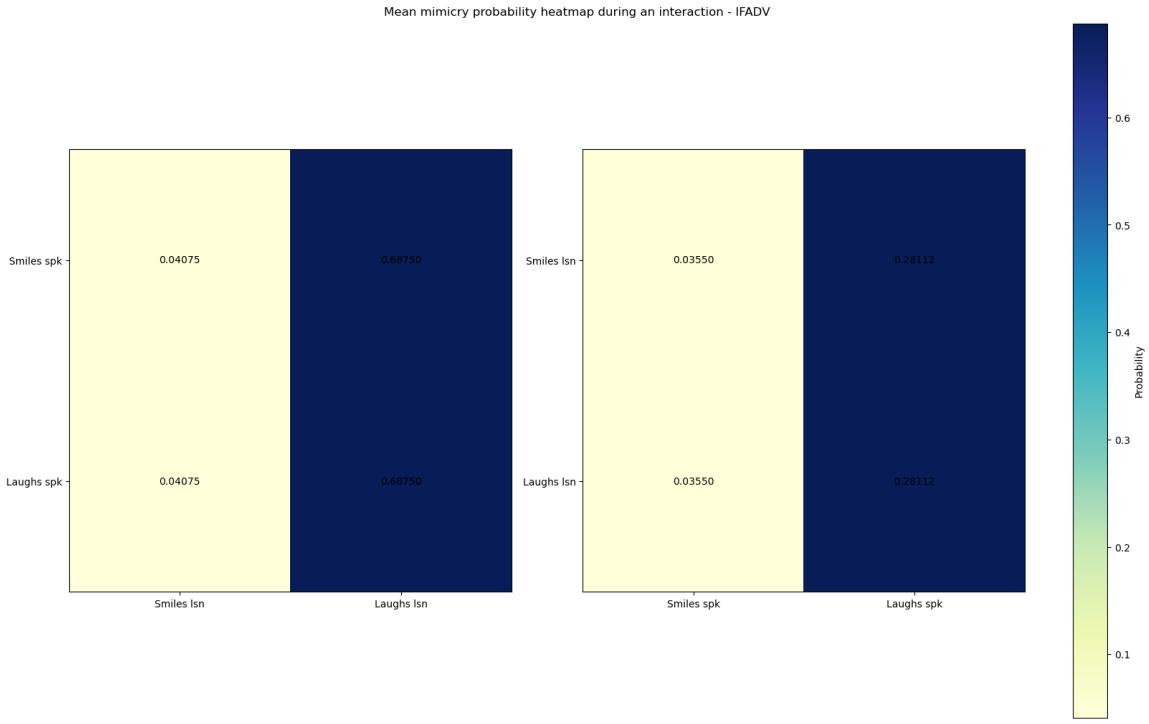
Here, we look at S&L mimicry in relation to the role of the mimicker (listener or speaker).

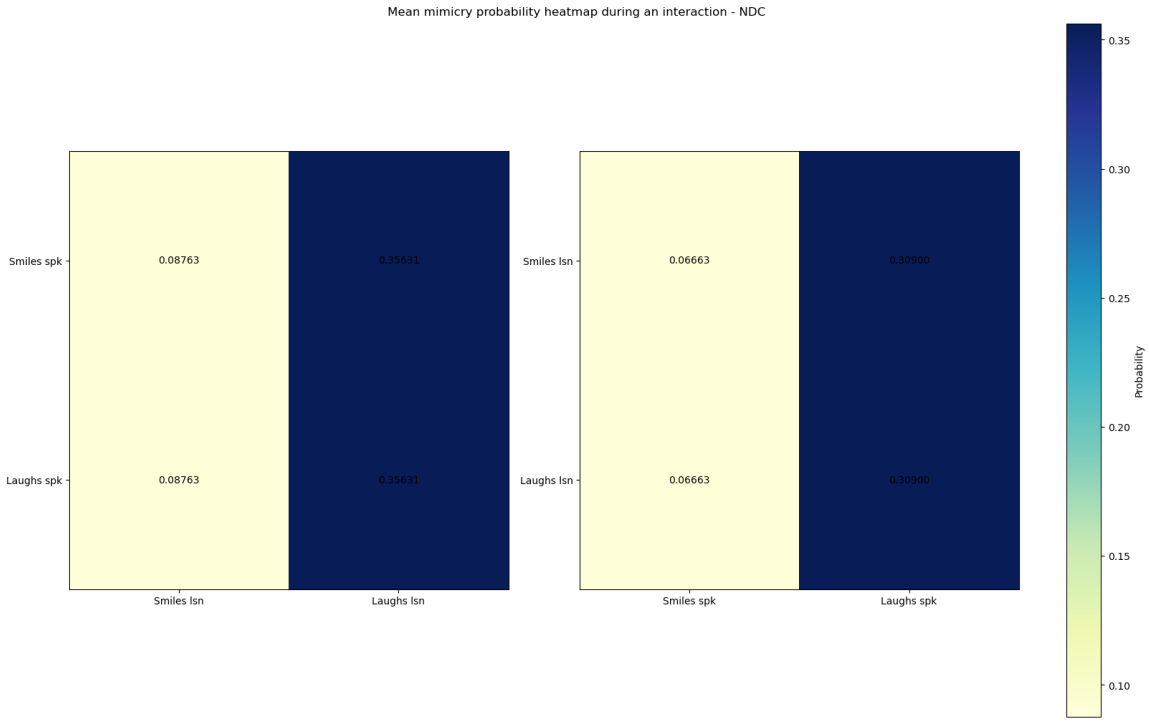
For all entities:

In [ ]:
```
probabilities_matrix = []
moyenne_prob_interaction = 0
# Define the pairs of expressions for the heatmaps
entity_pairs = [("spk", "lsn"), ("lsn", "spk")]
for i, database in enumerate(databases_name):
    if database==databases_pairs[i].replace('_pairs','').upper():
                databases_list=databases_pair_paths[databases_pairs[i]]
    # Create a new figure and axes
    fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16, 10), constrained_layc

    # Track the current row and column index
    current_row = 0
    current_col = 0

    # Create a 2x2 matrix with zeros
    probabilities_matrix_1 = np.zeros((len(expressions), len(expressions)))
    probabilities_matrix_2 = np.zeros((len(expressions), len(expressions)))
    for j in range(len(expressions)):
        expression_choiceA = expressions[j]
        for k in range(len(expressions)):
            expression_choiceB = expressions[k]
            for pair_index, (entityA, entityB) in enumerate(entity_pairs):
                # Get the statistics for each entity of Role (spk or lsn)
                list_mimicry_SL_by_role = give_mimicry_folder4(databases_list, c
                moyenne_prob_interaction = 0
                for item in list_mimicry_SL_by_role:
                    moyenne_prob_interaction += item[1]
                moyenne_prob_interaction = moyenne_prob_interaction/len(list_mim
                if entityA == "spk":
                    probabilities_matrix_1[current_row, current_col] = moyenne_p
                elif entityA == "lsn":
                    probabilities_matrix_2[current_row, current_col] = moyenne_p

            current_col += 1

        current_row += 1
        current_col = 0
    # Create the heatmaps for spk and lsn
    im_spk = axs[0].imshow(probabilities_matrix_1, cmap='YlGnBu', interpolation=
    im_lsn = axs[1].imshow(probabilities_matrix_2, cmap='YlGnBu', interpolation=

    # Add the probability values within each square for spk
    for x in range(len(expressions)):
        for y in range(len(expressions)):
            text_color_spk = 'black'
            axs[0].text(y, x, f'{probabilities_matrix_1[x, y]:.5f}', ha='center'

    # Add the probability values within each square for lsn
    for x in range(len(expressions)):
        for y in range(len(expressions)):
            text_color_lsn = 'black'
            axs[1].text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='center'

    # Customize the plots for spk and lsn
    axs[0].set_xticks(range(len(expressions)))
    axs[0].set_yticks(range(len(expressions)))
```

```
    axs[0].set_xticklabels(['Smiles lsn', 'Laughs lsn'])
    axs[0].set_yticklabels(['Smiles spk', 'Laughs spk'])


    axs[1].set_xticks(range(len(expressions)))
    axs[1].set_yticks(range(len(expressions)))
    axs[1].set_xticklabels(['Smiles spk', 'Laughs spk'])
    axs[1].set_yticklabels(['Smiles lsn', 'Laughs lsn'])

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im_spk, ax=axs, label='Probability')

    # Add a common title for the figure
    fig.suptitle(f'Mean mimicry probability heatmap during an interaction - {dat

    # Show the plot
    plt.show()
```







Results:

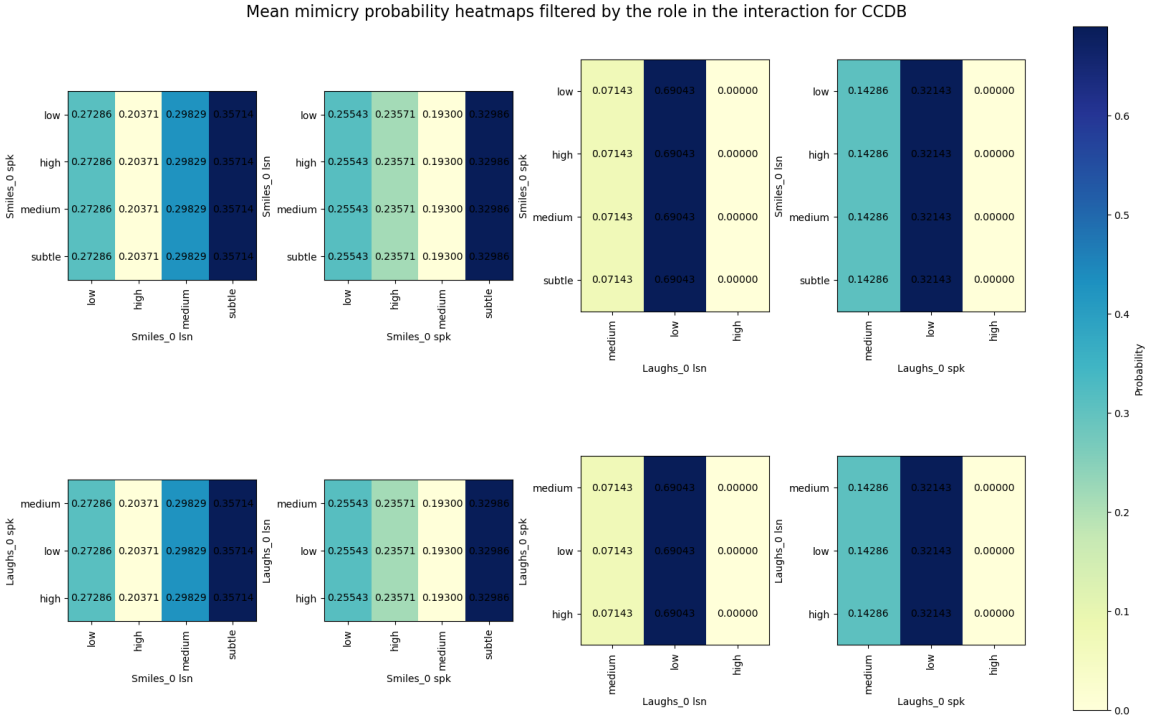
For A/B: Smiles and laughs are mostly mimicked by laughs and more often in the case of the listener mimicking the speaker.

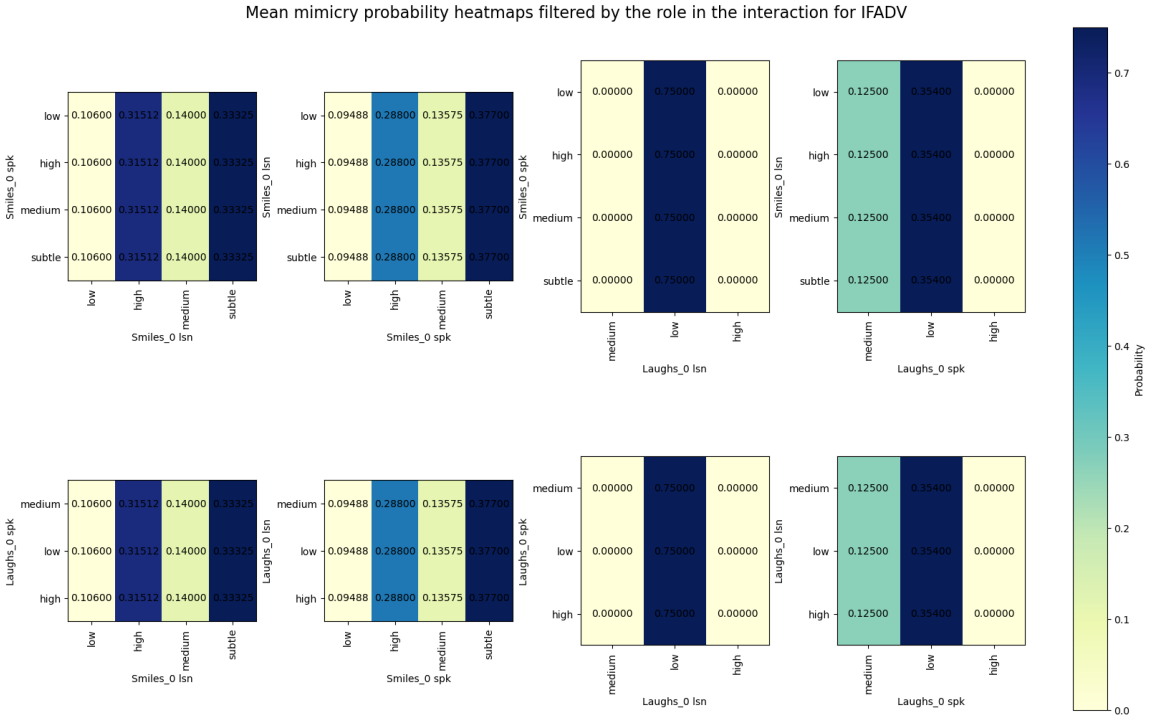
For each entity:

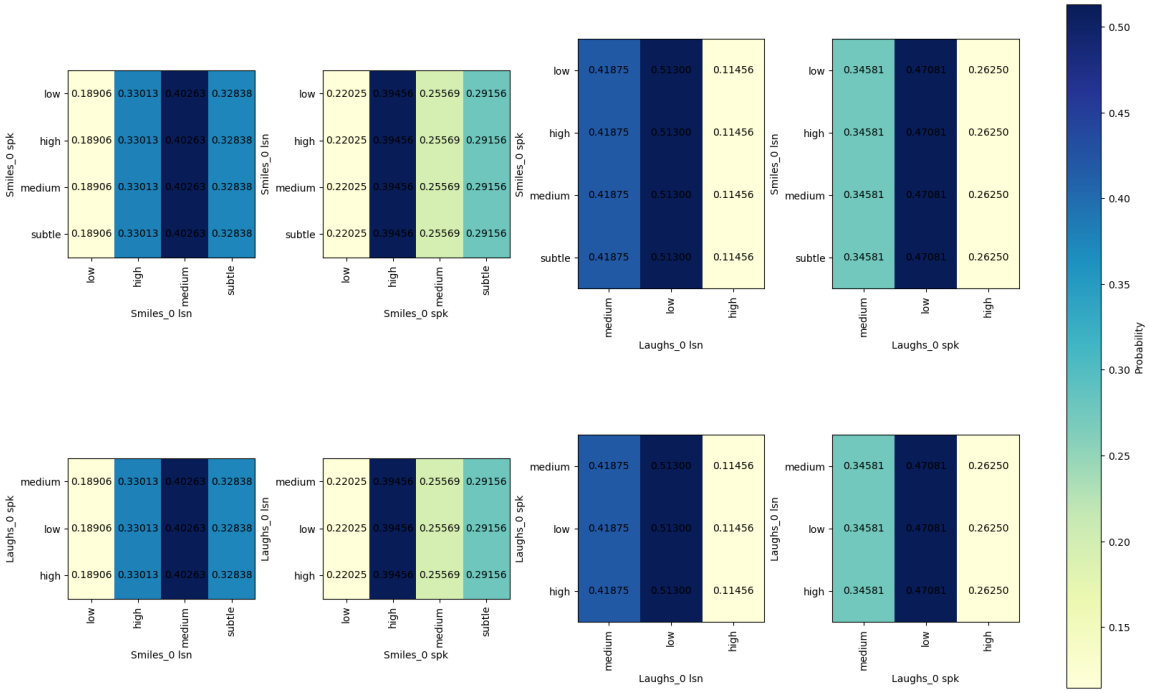
In [ ]:
```
probabilities_matrix_3 = []
moyenne_prob_interaction_3 = 0
# Define the pairs of expressions for the heatmaps
entity_pairs = [("spk", "lsn"), ("lsn", "spk")]

# Iterate over each database
for i, database in enumerate(databases_name):
    if database == databases_pairs[i].replace('_pairs', '').upper():
```

```
        databases_list = databases_pair_paths[databases_pairs[i]]

    # Create a new figure and axes
    fig, axs = plt.subplots(nrows=2, ncols=4, figsize=(16, 10), constrained_layc

    # Create a new dictionary to store entities grouped by expressions for each
    entities_by_expression = {}

    # Group entities by expressions
    for expression_choiceA in expressions:
        entitiesA = tier_lists[expression_choiceA]
        for entityA in entitiesA:
            if expression_choiceA not in entities_by_expression:
                entities_by_expression[expression_choiceA] = []
            entities_by_expression[expression_choiceA].append(entityA)

    # Iterate over expression pairs and entities for each database
    for j, (expression_choiceA, entitiesA) in enumerate(entities_by_expression.i
        for k, (expression_choiceB, entitiesB) in enumerate(entities_by_expressi
            # Create a matrix with zeros for spk and lsn categories
            num_entities_A = len(entitiesA)
            num_entities_B = len(entitiesB)
            probabilities_matrix_1 = np.zeros((num_entities_A, num_entities_B))
            probabilities_matrix_2 = np.zeros((num_entities_A, num_entities_B))

            # Track the current row and column index
            current_row = 0
            current_col = 0

            for entityA in entitiesA:
                for entityB in entitiesB:
                    for pair_index, (entity1, entity2) in enumerate(entity_pairs
                        # Get the statistics for each entity of Role (spk or lsn
                        list_mimicry_SL_by_role = give_mimicry_folder4(databases
                        moyenne_prob_interaction_3 = 0
                        for item in list_mimicry_SL_by_role:
                            moyenne_prob_interaction_3 += item[1]

                        moyenne_prob_interaction_3 /= len(list_mimicry_SL_by_rol

                        if entity1 == "spk":
                            probabilities_matrix_1[current_row, current_col] = m
                        elif entity1 == "lsn":
                            probabilities_matrix_2[current_row, current_col] = m

                    current_col += 1

                current_row += 1
                current_col = 0

            # Select the appropriate subplots for each heatmap
            ax_1 = axs[j, k * 2]
            ax_2 = axs[j, k * 2 + 1]

            # Create the heatmaps for spk and lsn
            im_spk = ax_1.imshow(probabilities_matrix_1, cmap='YlGnBu', interpol
            im_lsn = ax_2.imshow(probabilities_matrix_2, cmap='YlGnBu', interpol

            # Add the probability values within each square for spk and lsn
            for x in range(num_entities_A):
```

```
                for y in range(num_entities_B):
                    text_color_spk = 'black'
                    text_color_lsn = 'black'
                    ax_1.text(y, x, f'{probabilities_matrix_1[x, y]:.5f}', ha='c
                    ax_2.text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='c

            # Customize the plots for spk
            ax_1.set_xticks(np.arange(num_entities_B))
            ax_1.set_xticklabels(entitiesB, rotation=90)
            ax_1.set_yticks(np.arange(num_entities_A))
            ax_1.set_yticklabels(entitiesA)
            ax_1.set_xlabel(f"{expression_choiceB} lsn")
            ax_1.set_ylabel(f"{expression_choiceA} spk")

            # Customize the plots for lsn
            ax_2.set_xticks(np.arange(num_entities_B))
            ax_2.set_xticklabels(entitiesB, rotation=90)
            ax_2.set_yticks(np.arange(num_entities_A))
            ax_2.set_yticklabels(entitiesA)
            ax_2.set_xlabel(f"{expression_choiceB} spk")
            ax_2.set_ylabel(f"{expression_choiceA} lsn")

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im_spk, ax=axs, label='Probability')

    # Add a common title for the heatmaps of each database
    fig.suptitle(f'Mean mimicry probability heatmaps filtered by the role in the

    # Show the plot
    plt.show()
```







Results:

For A/B with delta = 0:

- CCDB: In the case of laughs mimicking smiles, the spk mimic more the listener. Laughs mimicking are rarely high.

Smiles mimicking smiles will often be of higher intensities, and more frequent in lsn mimicking spk. A laugh in the lsn will often be mimicked by a smile of greater intensity but by an equivalent laugh in the spk. Conversely, the lsn will more rarely mimic the spk's laughs or with less intensity.

- IFADV: In the case of a smile mimicking another smile, those of the lsn mimicking spk will be of less intensity. On the other hand, spk will have stronger smiles when

mimicking smiles of lsn. For laughs imitating smiles, the role has little impact (often low laughs). Lsn will mimic spk's smiles with laughs more frequently. Mimicked laughs are rare but a little more common from spk mimicking lsn.

- NDC: Spk mimicking lsn's smiles will produce smiles of higher intensity. As for the other datasets, the roles do not have much impact on a laugh mimicking a smile. They will be a bit more expressive in spk mimicking lsn.

The laughs are mimicked by stronger intensities in the case of spk imitating lsn.

For A/B with delta = 1000:

- The results are somewhat bizarre since they are very similar to each other. Hence the very similar graphics. However, we can see that most speakers mimic listeners more than the other way around. The intensities will generally be similar or higher intensities. We just have for IFADV the case of smiles and laughs mimicking smiles which is different and give use higher intensity of mimic for the listener (mimicking the speaker).

In general, we observe that laughs mimic laughs when spk mimics lsn. Therefore, a possible interpretation for this exception is that spk is producing an utterance, expecting a reaction from lsn, so spk will mimic the laughs with real laughs and with a similar level as the lsn's. However, when lsn mimics spk, lsn could be producing fake laughs, and so laughs of lower levels.

b) A mimicking B

Now we take a look at person A mimicking B in an interaction (so all first files in the pairs files mimicking second files in database order):

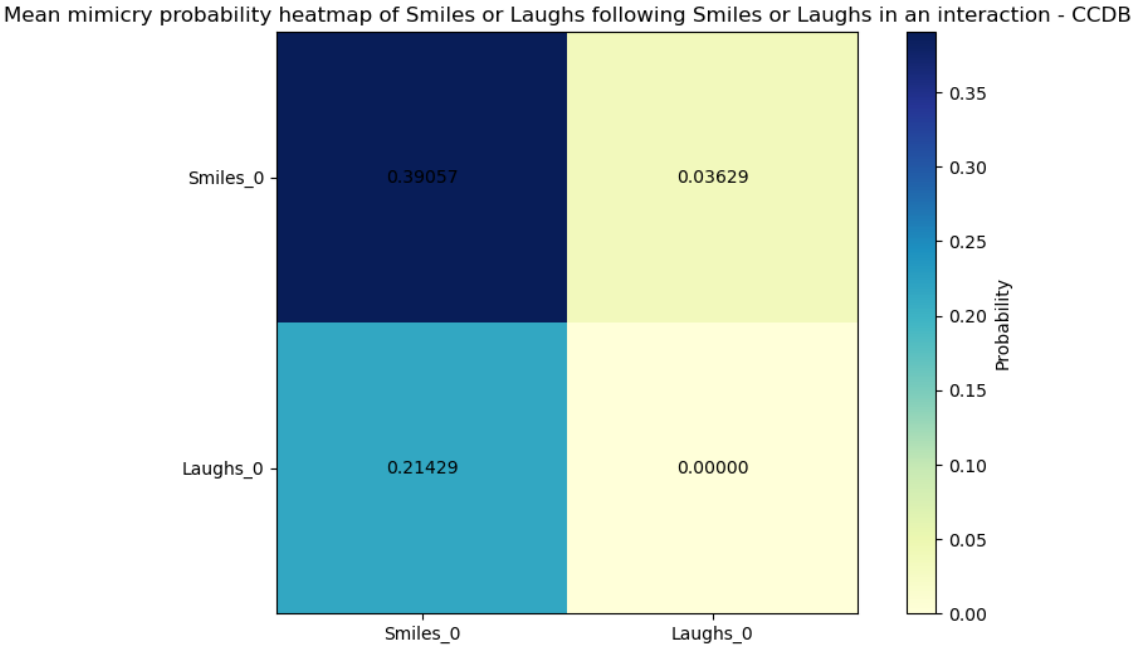
In [ ]:
```
# B/A -> person A mimics person B
mimic_choice2 = 'B/A'
```

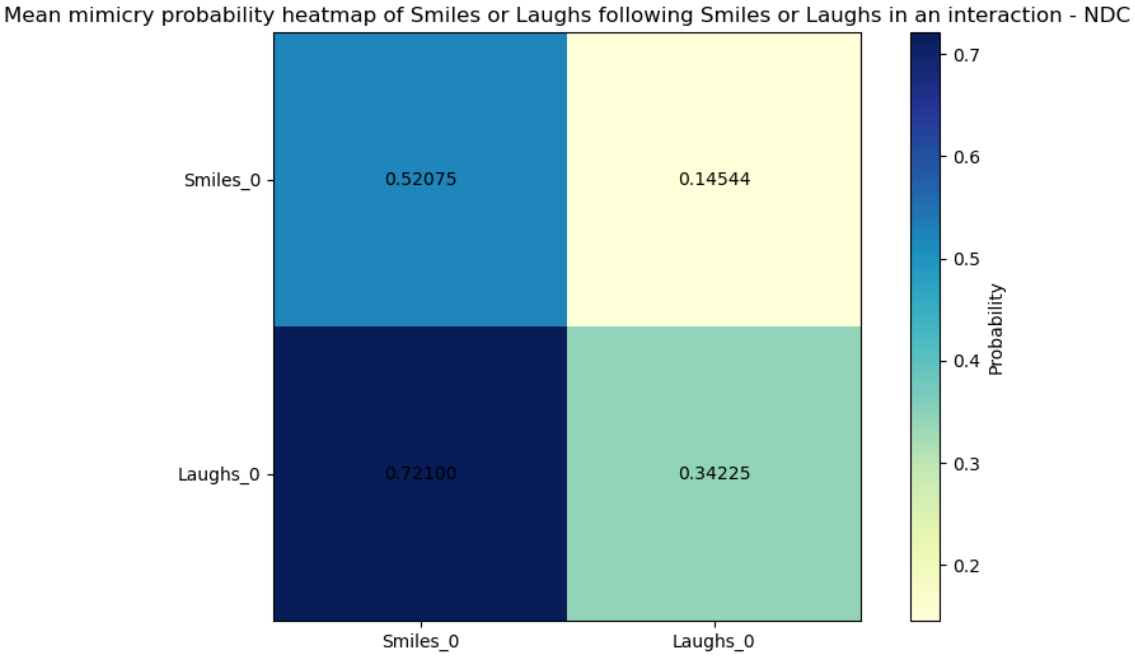
Mimicry Smiles vs Laughs (all entities) for each database

Extraction of the stats :

In [ ]:
```
probabilities_matrix = []
moyenne_prob_interaction = 0
for i, database in enumerate(databases_name):
    if database==databases_pairs[i].replace('_pairs','').upper():
                databases_list=databases_pair_paths[databases_pairs[i]]
    # Create a 2x2 matrix with zeros
    probabilities_matrix = np.zeros((len(expressions), len(expressions)))
    for j in range(len(expressions)):
        expression_choiceA = expressions[j]
        for k in range(len(expressions)):
            expression_choiceB = expressions[k]
            list_mimicry_SL = give_mimicry_folder2(databases_list, database.lowe
            for item in list_mimicry_SL:
                 moyenne_prob_interaction += item[1]
            moyenne_prob_interaction = moyenne_prob_interaction/len(list_mimicry
```

```
            probabilities_matrix[j, k] = moyenne_prob_interaction
            moyenne_prob_interaction = 0

    # Create a new figure for each database
    fig, ax = plt.subplots(figsize=(10, 6))

    # Create the heatmap plot
    im = ax.imshow(probabilities_matrix, cmap='YlGnBu', interpolation='nearest')

    # Add the probability values within each square
    for x in range(len(expressions)):
        for y in range(len(expressions)):
            text_color = 'black'  # Default text color
            ax.text(y, x, f'{probabilities_matrix[x, y]:.5f}', ha='center', va='

    # Customize the plot
    fig.colorbar(im, ax=ax, label='Probability')
    ax.set_xticks(range(len(expressions)))
    ax.set_yticks(range(len(expressions)))
    ax.set_xticklabels(expressions)
    ax.set_yticklabels(expressions)
    ax.set_title(f'Mean mimicry probability heatmap of Smiles or Laughs followin

    # Show the plot
    plt.show()
```





Analysis of the stats:

For B\A: We observe that smiles are mimicked by smiles, same for laughs mimicked by smiles (except for NDC in which laughs are mimicked by smiles and also laughs).

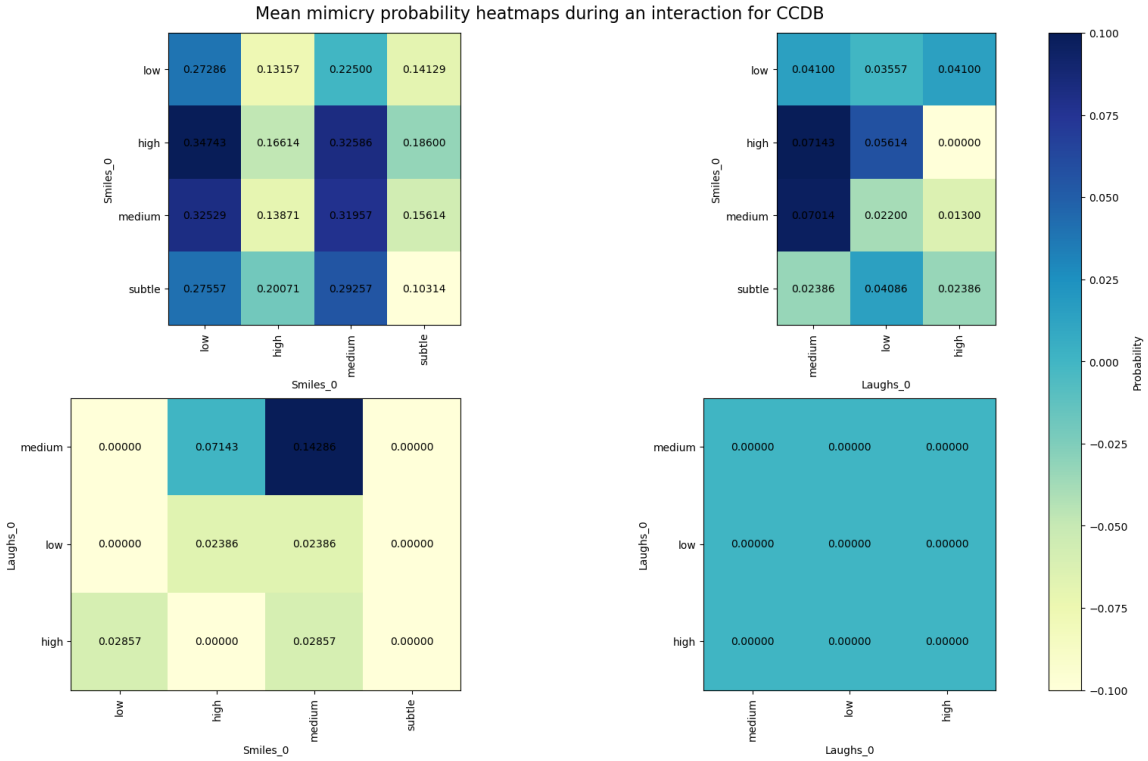
Mimicry Smiles vs Laughs (per entity) for each database

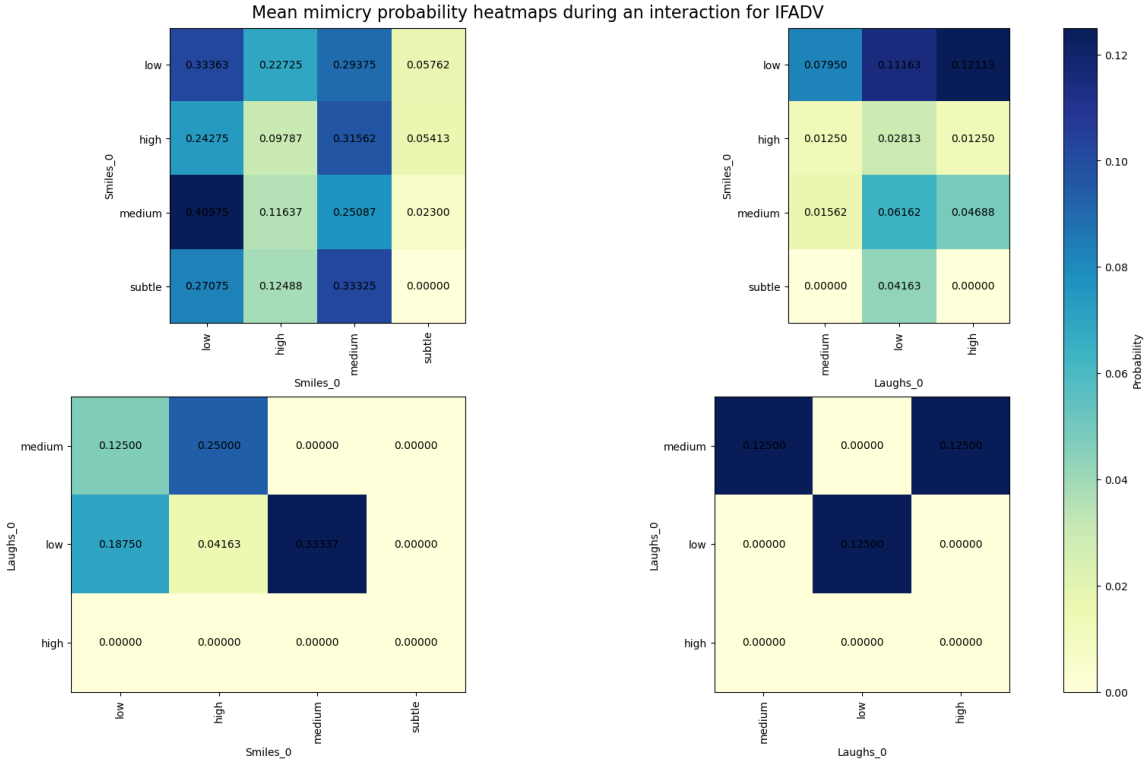
Extraction of the stats:

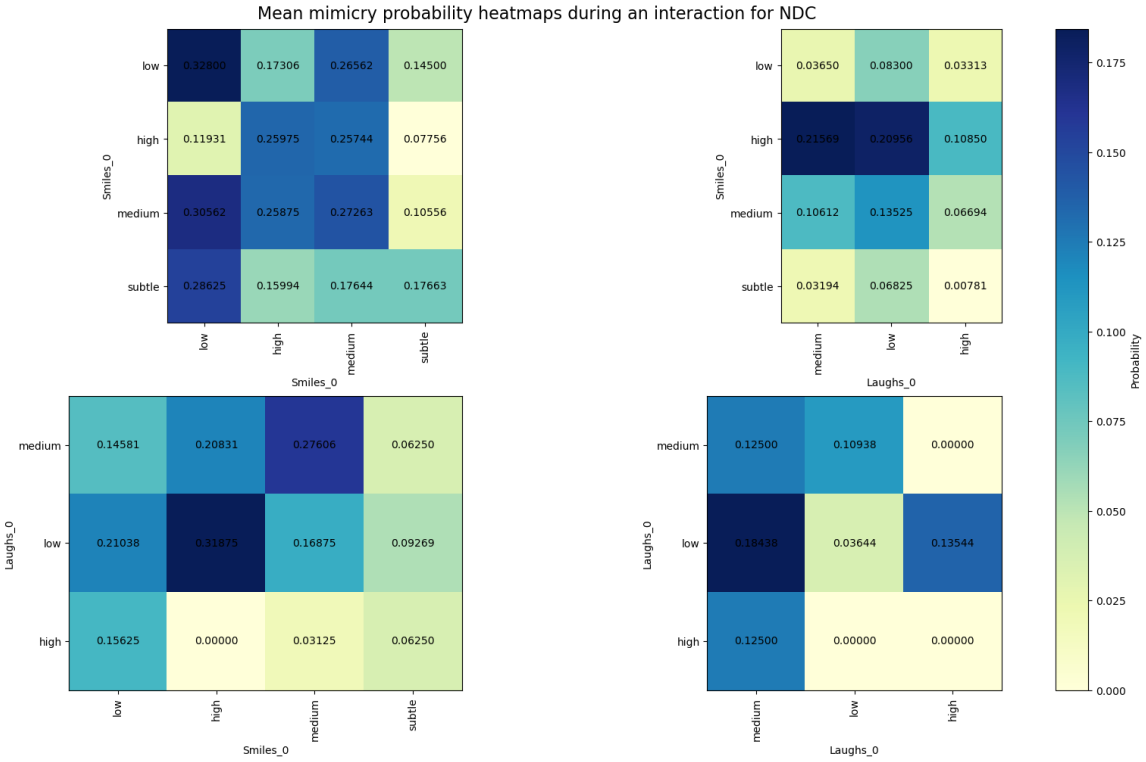
In [ ]:
```
probabilities_matrix_2 = []
moyenne_prob_interaction_2 = 0

# Iterate over each database
for i, database in enumerate(databases_name):
    if database == databases_pairs[i].replace('_pairs', '').upper():
        databases_list = databases_pair_paths[databases_pairs[i]]

    # Create a new figure and axes
```

```
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 10), constrained_layc

    # Create a new dictionary to store entities grouped by expressions for each
    entities_by_expression = {}

    # Group entities by expressions
    for expression_choiceA in expressions:
        entitiesA = tier_lists[expression_choiceA]
        for entityA in entitiesA:
            if expression_choiceA not in entities_by_expression:
                entities_by_expression[expression_choiceA] = []
            entities_by_expression[expression_choiceA].append(entityA)

    # Iterate over expression pairs and entities for each database
    for j, (expression_choiceA, entitiesA) in enumerate(entities_by_expression.i
        for k, (expression_choiceB, entitiesB) in enumerate(entities_by_expressi
            # Create a matrix with zeros
            num_entities_A = len(entitiesA)
            num_entities_B = len(entitiesB)
            probabilities_matrix_2 = np.zeros((num_entities_A, num_entities_B))

            # Track the current row and column index
            current_row = 0
            current_col = 0

            for entityA in entitiesA:
                for entityB in entitiesB:
                    list_mimicry_SL_entity = give_mimicry_folder2(databases_list

                    for item in list_mimicry_SL_entity:
                        moyenne_prob_interaction_2 += item[1]

                    moyenne_prob_interaction_2 = moyenne_prob_interaction_2 / le
                    probabilities_matrix_2[current_row, current_col] = moyenne_p
                    moyenne_prob_interaction_2 = 0

                    current_col += 1

                current_row += 1
                current_col = 0

            # Select the appropriate subplot for each heatmap
            ax = axs[j, k]

            # Create the heatmap plot
            im = ax.imshow(probabilities_matrix_2, cmap='YlGnBu', interpolation=

            # Add the probability values within each square
            for x in range(num_entities_A):
                for y in range(num_entities_B):
                    text_color = 'black'
                    ax.text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='cer

            # Customize the plot
            ax.set_xticks(np.arange(num_entities_B))
            ax.set_xticklabels(entitiesB, rotation=90)
            ax.set_yticks(np.arange(num_entities_A))
            ax.set_yticklabels(entitiesA)
            ax.set_xlabel(expression_choiceB)
            ax.set_ylabel(expression_choiceA)
```

```
# Add a common colorbar for the heatmaps of each database
fig.colorbar(im, ax=axs, label='Probability')

# Add a common title for the heatmaps of each database
fig.suptitle(f'Mean mimicry probability heatmaps during an interaction for {

# Show the plot
plt.show()
```







Analysis of the stats:

For B/A :

- For CCDB:
  - Smiles mimicking smiles have similar intensities or lower
  - Laughs mimicking smiles have a lower intensity
  - Smiles mimicking laughs are generally higher in intensity
  - Laughs mimicking laughs, we have 0 probabilities (maybe the value of delta is too low for this one)
- For IFADV:
  - Smiles mimicking smiles are of the same intensity or higher
  - Laughs mimicking smiles, we don(t have high probabilities but the intensity of lower seems to be higher)
  - Smiles mimicking laughs are mostly with higher intensity
  - Laughs mimicking laughs have the same intensity or higher
- For NDC:
  - Smiles mimicked by smiles are of the same intensity or higher
  - Laughs mimicking smiles are mostly of the same or lower intensity
  - Smiles mimicking laughs are mostly higher in intensity
  - Laughs mimicking laughs have the same intensity or higher

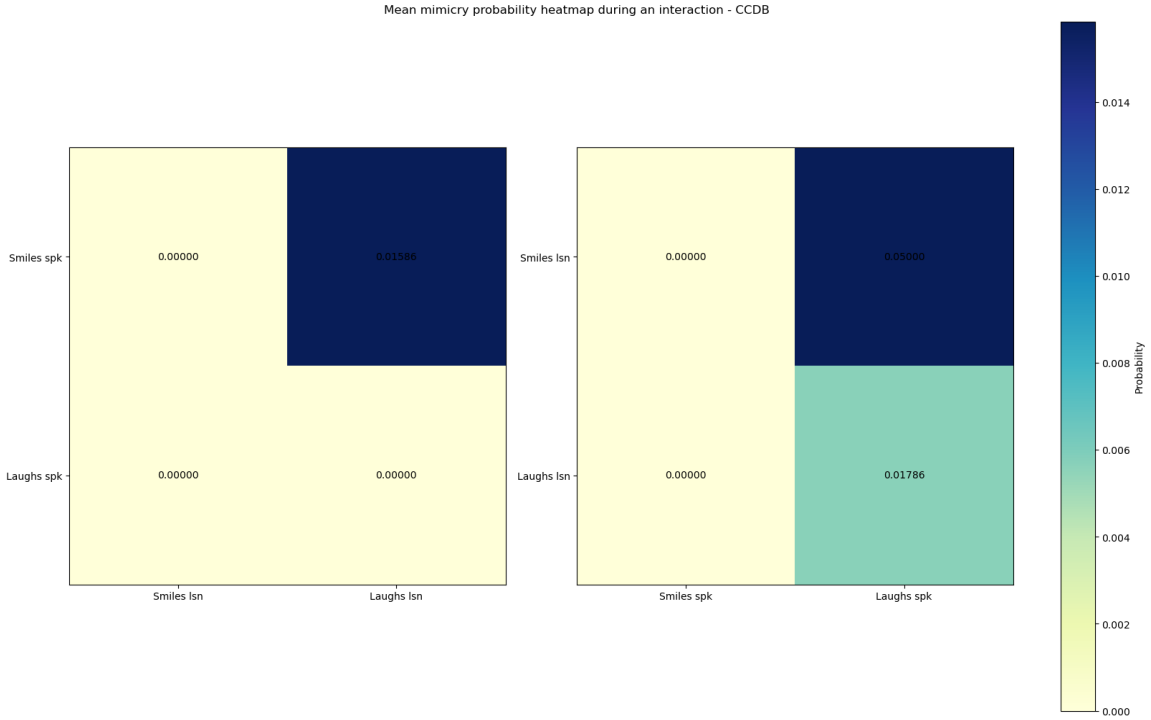
Lower levels of laughs mimicking smiles and higher levels of smiles mimicking laughs are in favor of the smile-laugh continuum theory mentioned earlier with smiles being on the low arousal side and laughs on the high side of a common arousal level scale for both S&L.

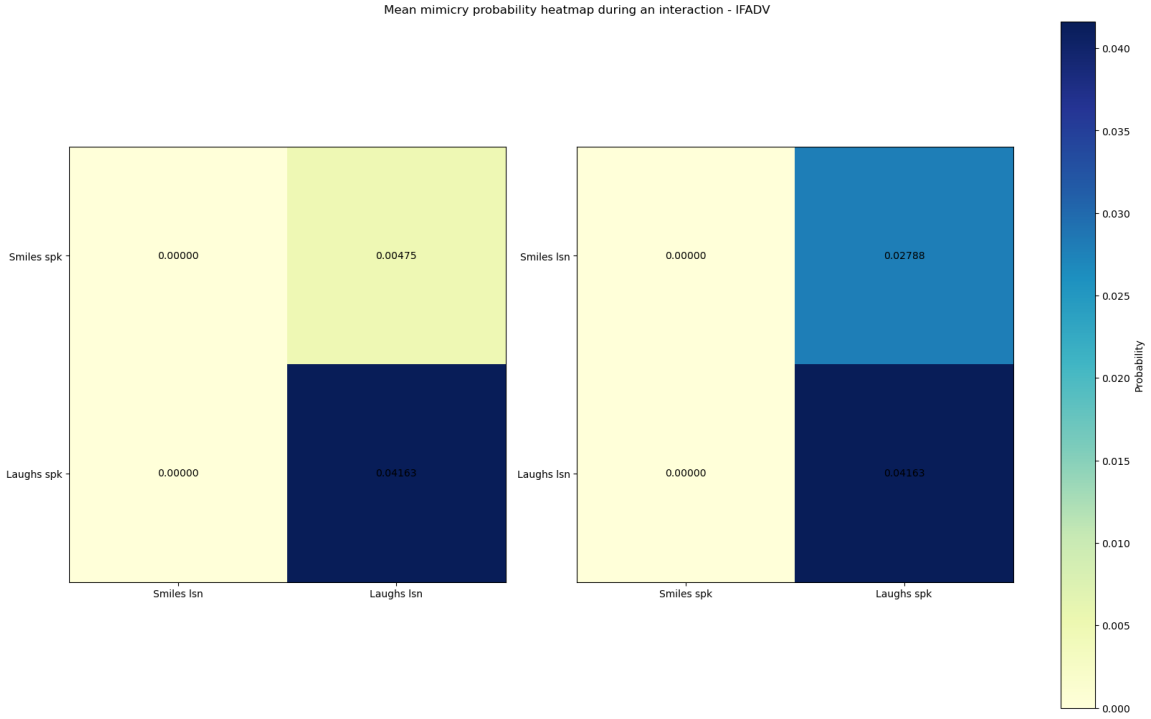
Filtered by role

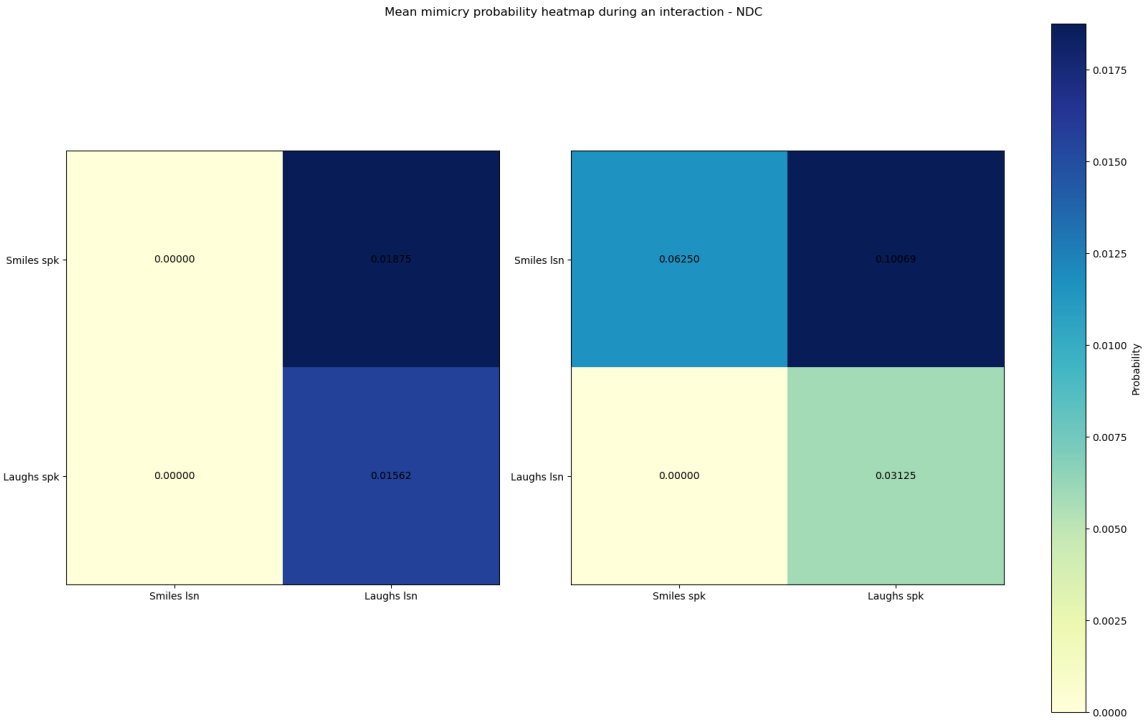
For all entities:

In [ ]:
```
probabilities_matrix = []
moyenne_prob_interaction = 0
# Define the pairs of expressions for the heatmaps
entity_pairs = [("spk", "lsn"), ("lsn", "spk")]
for i, database in enumerate(databases_name):
    if database==databases_pairs[i].replace('_pairs','').upper():
                databases_list=databases_pair_paths[databases_pairs[i]]
    # Create a new figure and axes
    fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16, 10), constrained_layc

    # Track the current row and column index
    current_row = 0
    current_col = 0

    # Create a 2x2 matrix with zeros
    probabilities_matrix_1 = np.zeros((len(expressions), len(expressions)))
    probabilities_matrix_2 = np.zeros((len(expressions), len(expressions)))
    for j in range(len(expressions)):
        expression_choiceA = expressions[j]
        for k in range(len(expressions)):
            expression_choiceB = expressions[k]
            for pair_index, (entityA, entityB) in enumerate(entity_pairs):
                # Get the statistics for each entity of Role (spk or lsn)
                list_mimicry_SL_by_role = give_mimicry_folder4(databases_list, c
                moyenne_prob_interaction = 0
                for item in list_mimicry_SL_by_role:
                    moyenne_prob_interaction += item[1]
                moyenne_prob_interaction = moyenne_prob_interaction/len(list_mim
                if entityA == "spk":
                    probabilities_matrix_1[current_row, current_col] = moyenne_p
                elif entityA == "lsn":
                    probabilities_matrix_2[current_row, current_col] = moyenne_p

            current_col += 1

        current_row += 1
        current_col = 0
    # Create the heatmaps for spk and lsn
    im_spk = axs[0].imshow(probabilities_matrix_1, cmap='YlGnBu', interpolation=
    im_lsn = axs[1].imshow(probabilities_matrix_2, cmap='YlGnBu', interpolation=

    # Add the probability values within each square for spk
    for x in range(len(expressions)):
        for y in range(len(expressions)):
            text_color_spk = 'black'
            axs[0].text(y, x, f'{probabilities_matrix_1[x, y]:.5f}', ha='center'

    # Add the probability values within each square for lsn
    for x in range(len(expressions)):
        for y in range(len(expressions)):
            text_color_lsn = 'black'
            axs[1].text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='center'

    # Customize the plots for spk and lsn
    axs[0].set_xticks(range(len(expressions)))
    axs[0].set_yticks(range(len(expressions)))
    axs[0].set_xticklabels(['Smiles lsn', 'Laughs lsn'])
```

```
    axs[0].set_yticklabels(['Smiles spk', 'Laughs spk'])


    axs[1].set_xticks(range(len(expressions)))
    axs[1].set_yticks(range(len(expressions)))
    axs[1].set_xticklabels(['Smiles spk', 'Laughs spk'])
    axs[1].set_yticklabels(['Smiles lsn', 'Laughs lsn'])

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im_spk, ax=axs, label='Probability')

    # Add a common title for the figure
    fig.suptitle(f'Mean mimicry probability heatmap during an interaction - {dat

    # Show the plot
    plt.show()
```







Results:

For B/A:

- For CCDB: The listener mimic the smiles of the speaker by laughs mostly. On the other hand, the speaker mimic the smiles and laughs of the listener by laughs. (The smiles are more mimicked)
- For IFADV: The listener mimic smiles and laughs of the speaker by laughs but he mimics mostly the laughs. Same things of the speaker mimicking the listener.
- For NDC: The listener mimic the smiles and laughs of the speaker equally. The speaker tend to mimic more the smiles of the listener by laughs (+) and smiles.

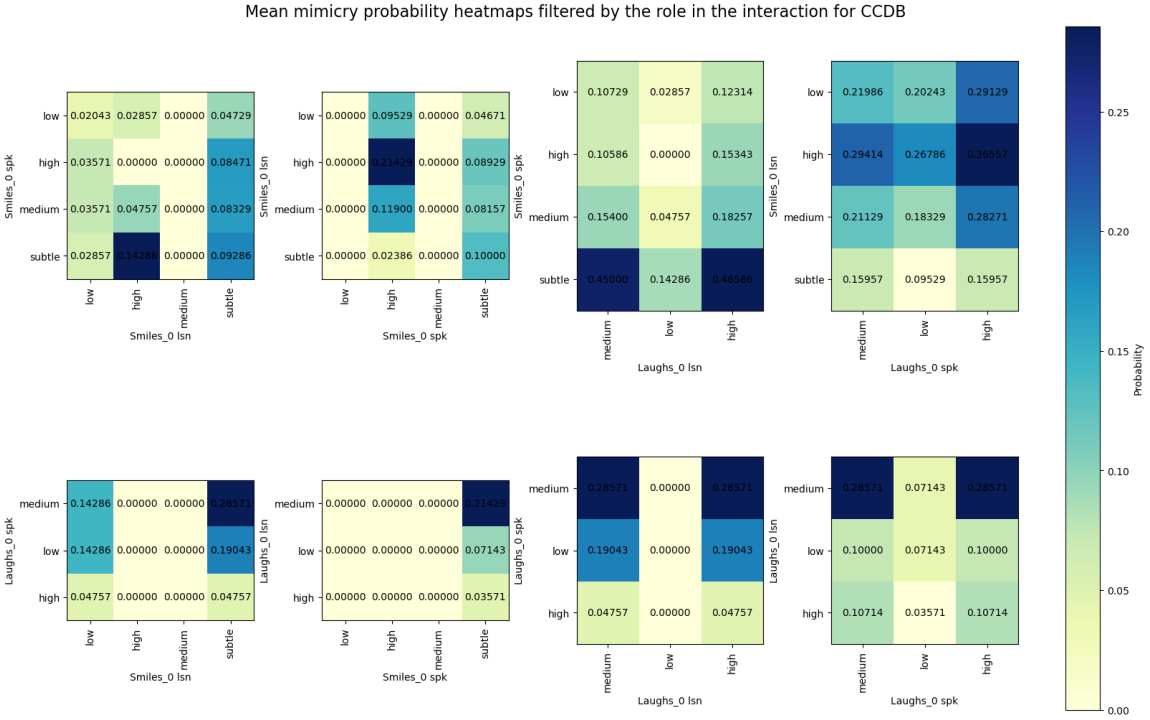
In general, the speaker mimic more the listener in this case.

For each entity:

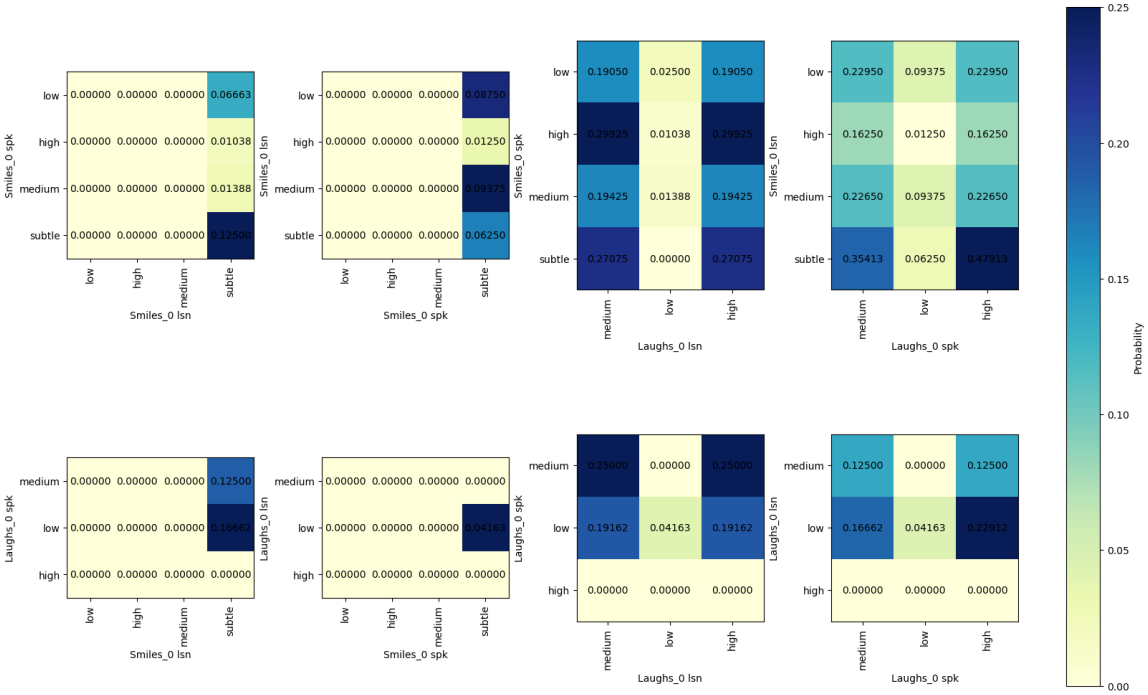
In [ ]:
```
probabilities_matrix_3 = []
moyenne_prob_interaction_3 = 0
# Define the pairs of expressions for the heatmaps
entity_pairs = [("spk", "lsn"), ("lsn", "spk")]

# Iterate over each database
for i, database in enumerate(databases_name):
    if database == databases_pairs[i].replace('_pairs', '').upper():
        databases_list = databases_pair_paths[databases_pairs[i]]

    # Create a new figure and axes
    fig, axs = plt.subplots(nrows=2, ncols=4, figsize=(16, 10), constrained_layc

    # Create a new dictionary to store entities grouped by expressions for each
    entities_by_expression = {}

    # Group entities by expressions
    for expression_choiceA in expressions:
```

```
        entitiesA = tier_lists[expression_choiceA]
        for entityA in entitiesA:
            if expression_choiceA not in entities_by_expression:
                entities_by_expression[expression_choiceA] = []
            entities_by_expression[expression_choiceA].append(entityA)

    # Iterate over expression pairs and entities for each database
    for j, (expression_choiceA, entitiesA) in enumerate(entities_by_expression.i
        for k, (expression_choiceB, entitiesB) in enumerate(entities_by_expressi
            # Create a matrix with zeros for spk and lsn categories
            num_entities_A = len(entitiesA)
            num_entities_B = len(entitiesB)
            probabilities_matrix_1 = np.zeros((num_entities_A, num_entities_B))
            probabilities_matrix_2 = np.zeros((num_entities_A, num_entities_B))

            # Track the current row and column index
            current_row = 0
            current_col = 0

            for entityA in entitiesA:
                for entityB in entitiesB:
                    for pair_index, (entity1, entity2) in enumerate(entity_pairs
                        # Get the statistics for each entity of Role (spk or lsn
                        list_mimicry_SL_by_role = give_mimicry_folder4(databases
                        moyenne_prob_interaction_3 = 0
                        for item in list_mimicry_SL_by_role:
                            moyenne_prob_interaction_3 += item[1]

                        moyenne_prob_interaction_3 /= len(list_mimicry_SL_by_rol

                        if entity1 == "spk":
                            probabilities_matrix_1[current_row, current_col] = m
                        elif entity1 == "lsn":
                            probabilities_matrix_2[current_row, current_col] = m

                    current_col += 1

                current_row += 1
                current_col = 0

            # Select the appropriate subplots for each heatmap
            ax_1 = axs[j, k * 2]
            ax_2 = axs[j, k * 2 + 1]

            # Create the heatmaps for spk and lsn
            im_spk = ax_1.imshow(probabilities_matrix_1, cmap='YlGnBu', interpol
            im_lsn = ax_2.imshow(probabilities_matrix_2, cmap='YlGnBu', interpol

            # Add the probability values within each square for spk and lsn
            for x in range(num_entities_A):
                for y in range(num_entities_B):
                    text_color_spk = 'black'
                    text_color_lsn = 'black'
                    ax_1.text(y, x, f'{probabilities_matrix_1[x, y]:.5f}', ha='c
                    ax_2.text(y, x, f'{probabilities_matrix_2[x, y]:.5f}', ha='c

            # Customize the plots for spk
            ax_1.set_xticks(np.arange(num_entities_B))
            ax_1.set_xticklabels(entitiesB, rotation=90)
            ax_1.set_yticks(np.arange(num_entities_A))
```

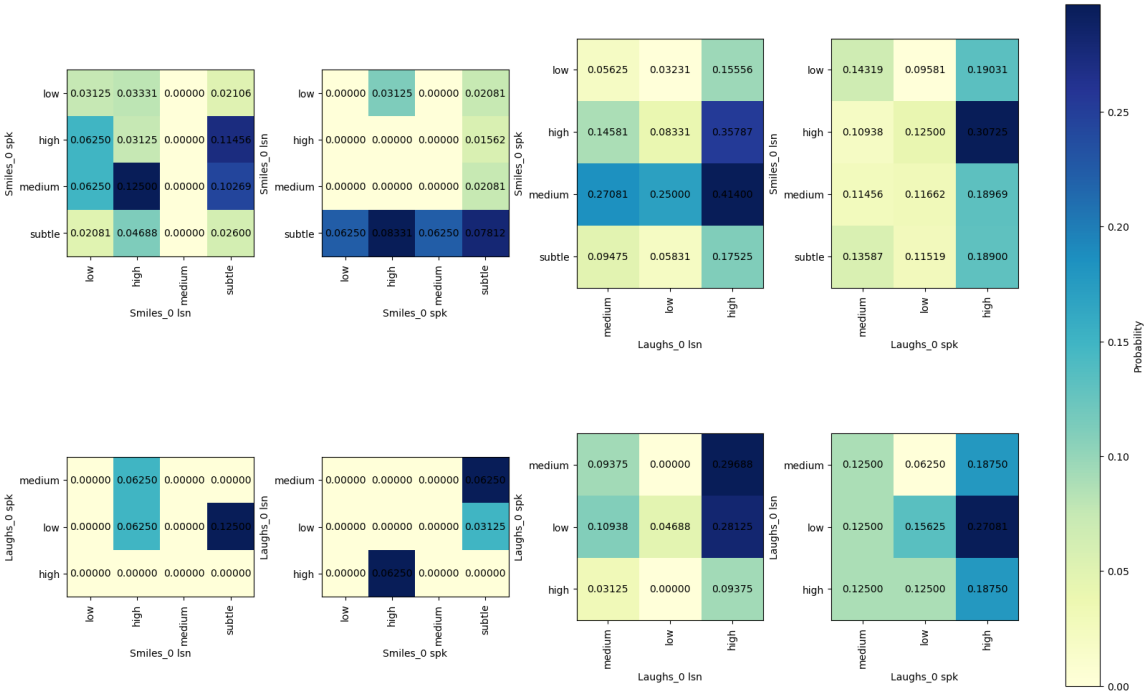
```
            ax_1.set_yticklabels(entitiesA)
            ax_1.set_xlabel(f"{expression_choiceB} lsn")
            ax_1.set_ylabel(f"{expression_choiceA} spk")

            # Customize the plots for lsn
            ax_2.set_xticks(np.arange(num_entities_B))
            ax_2.set_xticklabels(entitiesB, rotation=90)
            ax_2.set_yticks(np.arange(num_entities_A))
            ax_2.set_yticklabels(entitiesA)
            ax_2.set_xlabel(f"{expression_choiceB} spk")
            ax_2.set_ylabel(f"{expression_choiceA} lsn")

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im_spk, ax=axs, label='Probability')

    # Add a common title for the heatmaps of each database
    fig.suptitle(f'Mean mimicry probability heatmaps filtered by the role in the

    # Show the plot
    plt.show()
```



Mean mimicry probability heatmaps filtered by the role in the interaction for IFADV



Mean mimicry probability heatmaps filtered by the role in the interaction for NDC



Results:

For B/A:

- For CCDB: The speaker mimic the listener with the same or higher intensity mostly. The listener mimic more rarely or with lower intensity except for laughs mimicking laughs and high smiles mimicking subtle smiles.
- For IFADV: There is less probabilities here maybe due to the dataset or the value of delta. Nevertheless, we can see than the spk is more likely to mimic the smiles of the listener. The listener seems to mimic more the laughs.
- For NDC: The mimic of the speaker are higher in intensity than the one from the listener. But we see than the listener mimic more the smiles of the speaker than the other way.

In general, we observe that laughs mimic laughs when spk mimics lsn and generally mimic more than the listener (with higher intensities than lsn mimicking spk). Therefore, a possible interpretation for this exception is that spk is producing an utterance, expecting a reaction from lsn, so spk will mimic the laughs with real laughs and with a similar level as the lsn's. However, when lsn mimics spk, lsn could be producing fake laughs, and so laughs of lower levels.

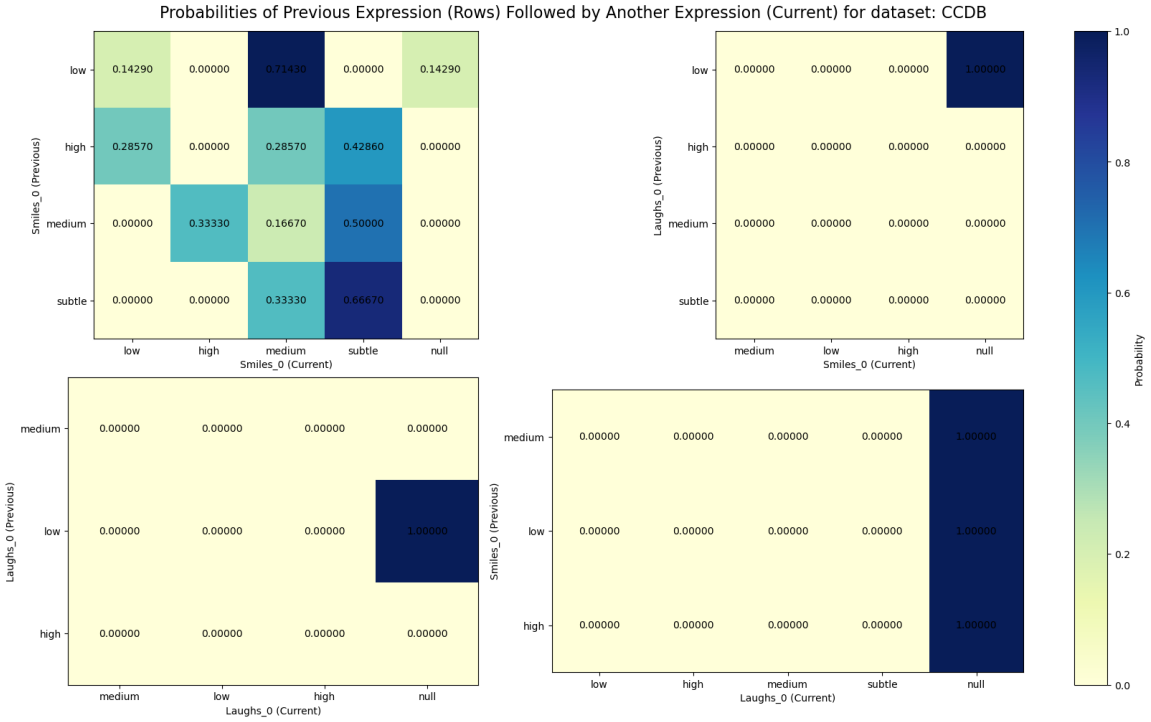
3-Smile and laugh sequences (intra)

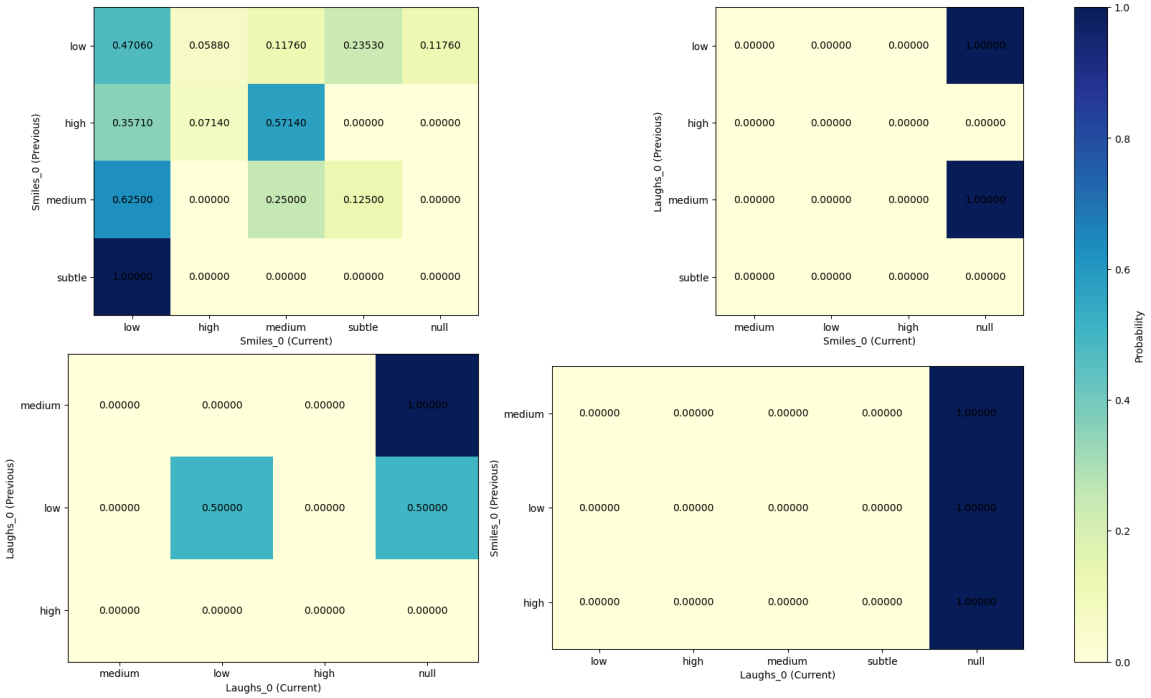
In this section we study the temporal sequence patterns of S&L produced during an entire interaction (i.e., one's own S&L sequence pattern, not considering their interlocutor's). For this, we consider the S&L directly following any smile or laugh.

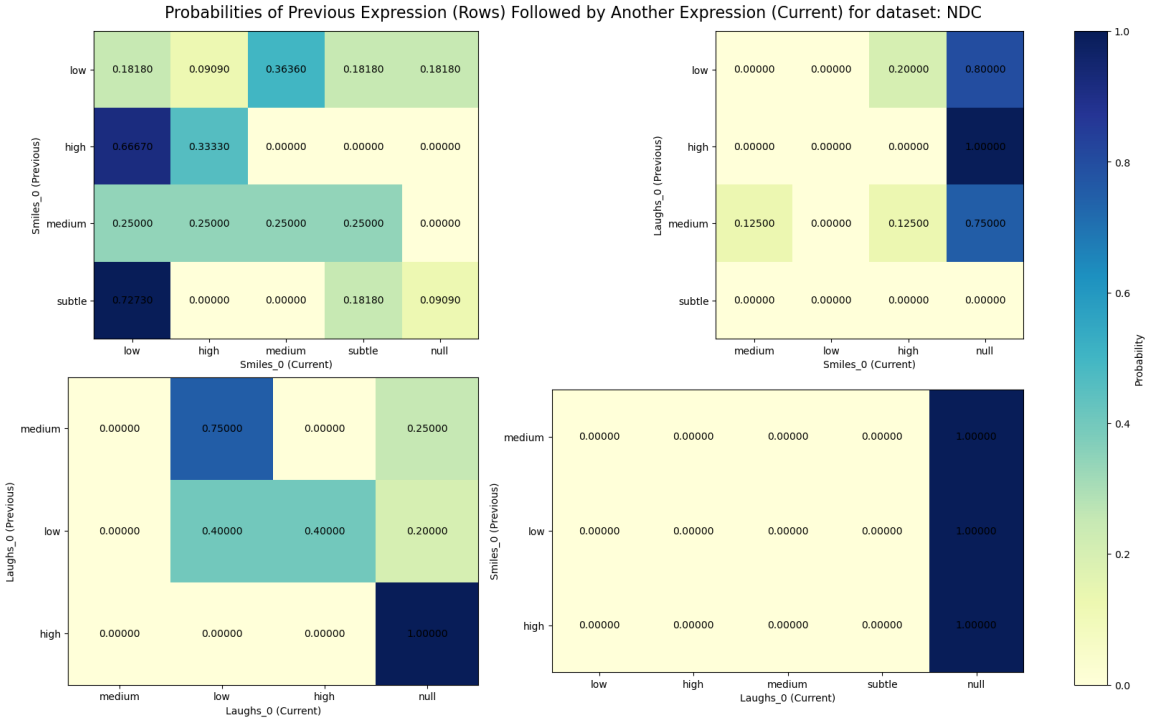
Track of previous expression for each dataset of each individual:

In [ ]:
```
warnings.filterwarnings("ignore")
probabilities_matrix = []
moyenne_prob_interaction = 0
# Define the pairs of expressions for the heatmaps
expression_pairs = [("Smiles_0", "Smiles_0"), ("Smiles_0", "Laughs_0"), ("Laughs

# Iterate over the datasets
for dataset_index, database in enumerate(databases_name):
    # Create a new figure and axes for the current dataset
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 10), constrained_layc

    # Track the current row and column index
    current_row = 0
    current_col = 0

    # Iterate over the expression pairs for the heatmaps
    for pair_index, (expression_choiceA, expression_choiceB) in enumerate(expres
        # Get the appropriate intensity labels based on the expressions
        intensity_labelsT = copy.deepcopy(smiles_intensities) if expression_choi
        intensity_labelsC = copy.deepcopy(smiles_intensities) if expression_choi
        intensity_labelsC.append("null")

        # Create a 2x2 matrix with zeros
```

```
        probabilities_matrix_intensity = np.zeros((len(intensity_labelsT), len(i

        list_mimicry_TC = expression_track_byI(expression_choiceA, expression_ch
        list_mimicry_TC_prev = list_mimicry_TC[0]  # Previous expression DataFra

        # Create the heatmap for previous expression
        ax = axs[current_row, current_col]
        for i, intensityC in enumerate(intensity_labelsC):
            for j, intensityT in enumerate(intensity_labelsT):
                try:
                    filtered_list_prev = list_mimicry_TC_prev[
                        (list_mimicry_TC_prev['Intensityp'] == intensityC) &
                        (list_mimicry_TC_prev[f'Current_level_{expression_choice
                    ]
                    percentage_prev = filtered_list_prev['Percentagep'].values[0
                    probabilities_matrix_intensity[j, i] = percentage_prev / 100
                except:
                    pass

        # Plot the heatmap
        im = ax.imshow(probabilities_matrix_intensity, cmap='YlGnBu', interpolat
        # Add the text for each cell
        for i in range(len(intensity_labelsC)):
            for j in range(len(intensity_labelsT)):
                text = ax.text(i, j, f"{probabilities_matrix_intensity[j, i]:.5f

        # Set the title for the current heatmap
        ax.set_xticks(range(len(intensity_labelsC)))
        ax.set_yticks(range(len(intensity_labelsT)))
        ax.set_xticklabels(intensity_labelsC)
        ax.set_yticklabels(intensity_labelsT)
        ax.set_xlabel(f"{expression_choiceA} (Current)")
        ax.set_ylabel(f"{expression_choiceB} (Previous)")

        # Update the current row and column index
        current_col += 1
        if current_col == 2:
            current_row += 1
            current_col = 0

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im, ax=axs, label='Probability')

    # Set the title for the figure based on the dataset
    fig.suptitle(f"Probabilities of Previous Expression (Rows) Followed by Anoth

    # Show the figure
    plt.show()
```


Probabilities of Previous Expression (Rows) Followed by Another Expression (Current) for dataset: CCDB
Smiles_0 (Previous)
Smiles_0 (Current)
low
high
medium
subtle
null
0.14290 0.00000 0.71430 0.00000 0.14290
0.28570 0.00000 0.28570 0.42860 0.00000
0.00000 0.33330 0.16670 0.50000 0.00000
0.00000 0.00000 0.33330 0.66670 0.00000
Laughs_0 (Previous)
Laughs_0 (Current)
0.00000 0.00000 0.00000 1.00000
0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000
Probability
1.0
0.8
0.6
0.4
0.2
0.0

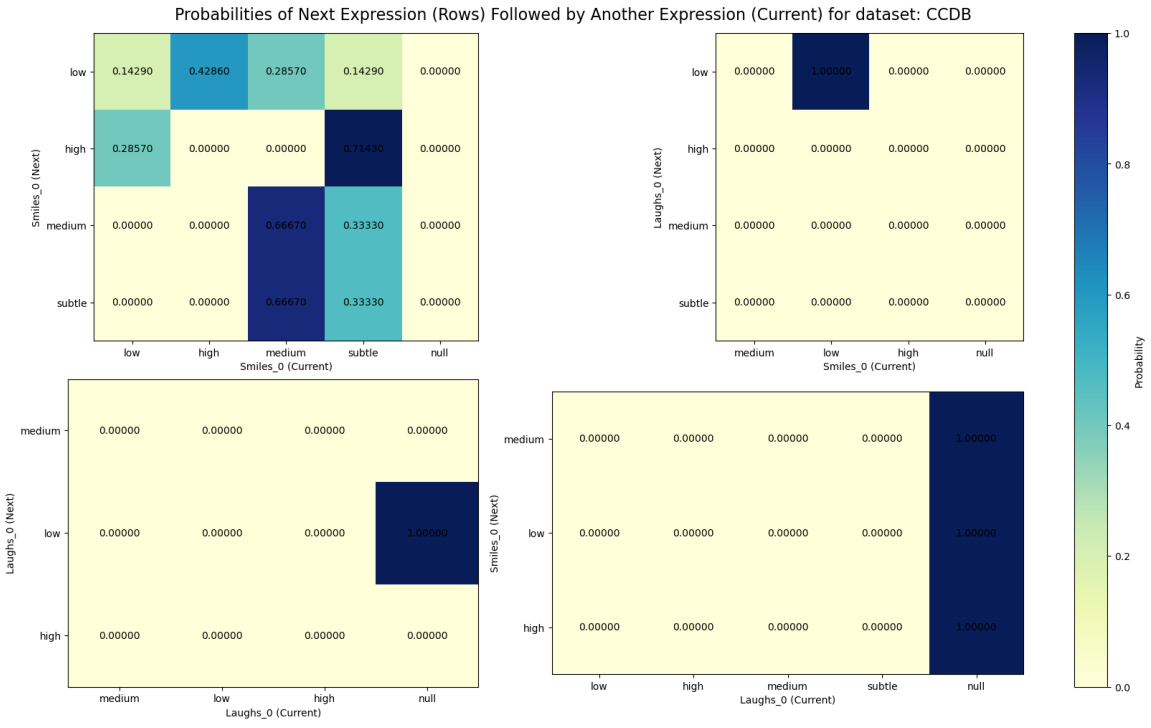

Probabilities of Previous Expression (Rows) Followed by Another Expression (Current) for dataset: IFADV
Smiles_0 (Previous)
Smiles_0 (Current)
low
high
medium
subtle
null
0.47060 0.05880 0.11760 0.23530 0.11760
0.35710 0.07140 0.57140 0.00000 0.00000
0.62500 0.00000 0.25000 0.12500 0.00000
1.00000 0.00000 0.00000 0.00000 0.00000
Laughs_0 (Previous)
Laughs_0 (Current)
0.00000 0.00000 0.00000 1.00000
0.00000 0.50000 0.00000 0.50000
0.00000 0.00000 0.00000 0.00000
Probability
1.0
0.8
0.6
0.4
0.2
0.0

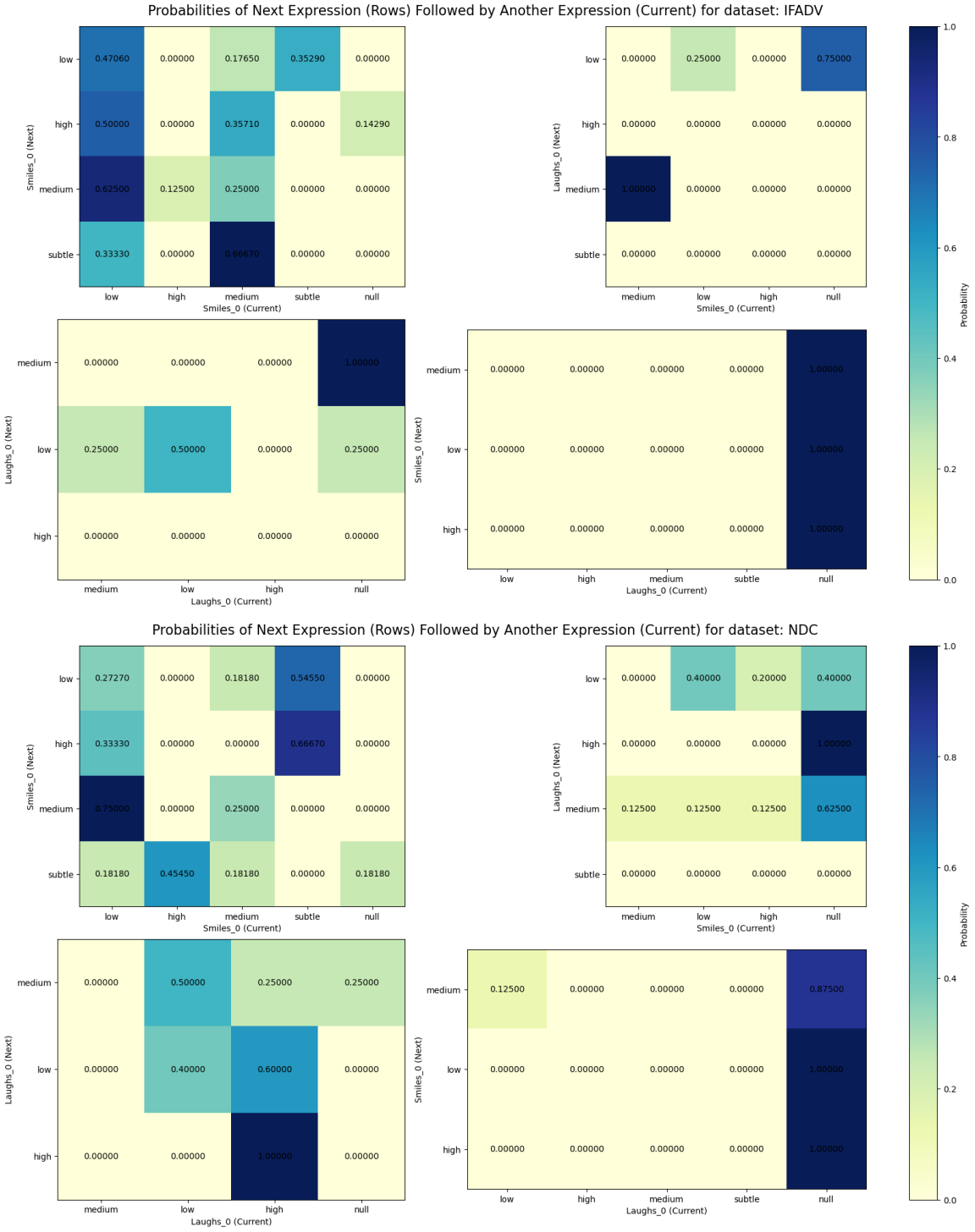


Track of next expression for each dataset of each individual:

In [ ]:
```
probabilities_matrix = []
moyenne_prob_interaction = 0
# Define the pairs of expressions for the heatmaps
expression_pairs = [("Smiles_0", "Smiles_0"), ("Smiles_0", "Laughs_0"), ("Laughs

# Iterate over the datasets
for dataset_index, database in enumerate(databases_name):
    # Create a new figure and axes for the current dataset
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 10), constrained_layc

    # Track the current row and column index
    current_row = 0
    current_col = 0

    # Iterate over the expression pairs for the heatmaps
    for pair_index, (expression_choiceA, expression_choiceB) in enumerate(expres
        # Get the appropriate intensity labels based on the expressions
        intensity_labelsT = copy.deepcopy(smiles_intensities) if expression_choi
        intensity_labelsC = copy.deepcopy(smiles_intensities) if expression_choi
        intensity_labelsC.append("null")

        # Create a 2x2 matrix with zeros
        probabilities_matrix_intensity = np.zeros((len(intensity_labelsT), len(i

        list_mimicry_TC = expression_track_byI(expression_choiceA, expression_ch
        list_mimicry_TC_next = list_mimicry_TC[1]  # Next expression DataFrame

        # Create the heatmap for next expression
        ax = axs[current_row, current_col]
        for i, intensityC in enumerate(intensity_labelsC):
            for j, intensityT in enumerate(intensity_labelsT):
                try :
                    filtered_list_next = list_mimicry_TC_next[
                        (list_mimicry_TC_next['Intensityf'] == intensityC) &
```

```
                        (list_mimicry_TC_next[f'Current_level_{expression_choice
                    ]
                    percentage_next = filtered_list_next['Percentagef'].values[0
                    probabilities_matrix_intensity[j, i] = percentage_next / 100
                except:
                    pass

        # Plot the heatmap
        im = ax.imshow(probabilities_matrix_intensity, cmap='YlGnBu', interpolat
        # Add the text for each cell
        for i in range(len(intensity_labelsC)):
            for j in range(len(intensity_labelsT)):
                text = ax.text(i, j, f"{probabilities_matrix_intensity[j, i]:.5f

        # Set the title for the current heatmap
        ax.set_xticks(range(len(intensity_labelsC)))
        ax.set_yticks(range(len(intensity_labelsT)))
        ax.set_xticklabels(intensity_labelsC)
        ax.set_yticklabels(intensity_labelsT)
        ax.set_xlabel(f"{expression_choiceA} (Current)")
        ax.set_ylabel(f"{expression_choiceB} (Next)")

        # Update the current row and column index
        current_col += 1
        if current_col == 2:
            current_row += 1
            current_col = 0

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im, ax=axs, label='Probability')

    # Set the title for the figure based on the dataset
    fig.suptitle(f"Probabilities of Next Expression (Rows) Followed by Another E

    # Show the figure
    plt.show()
```



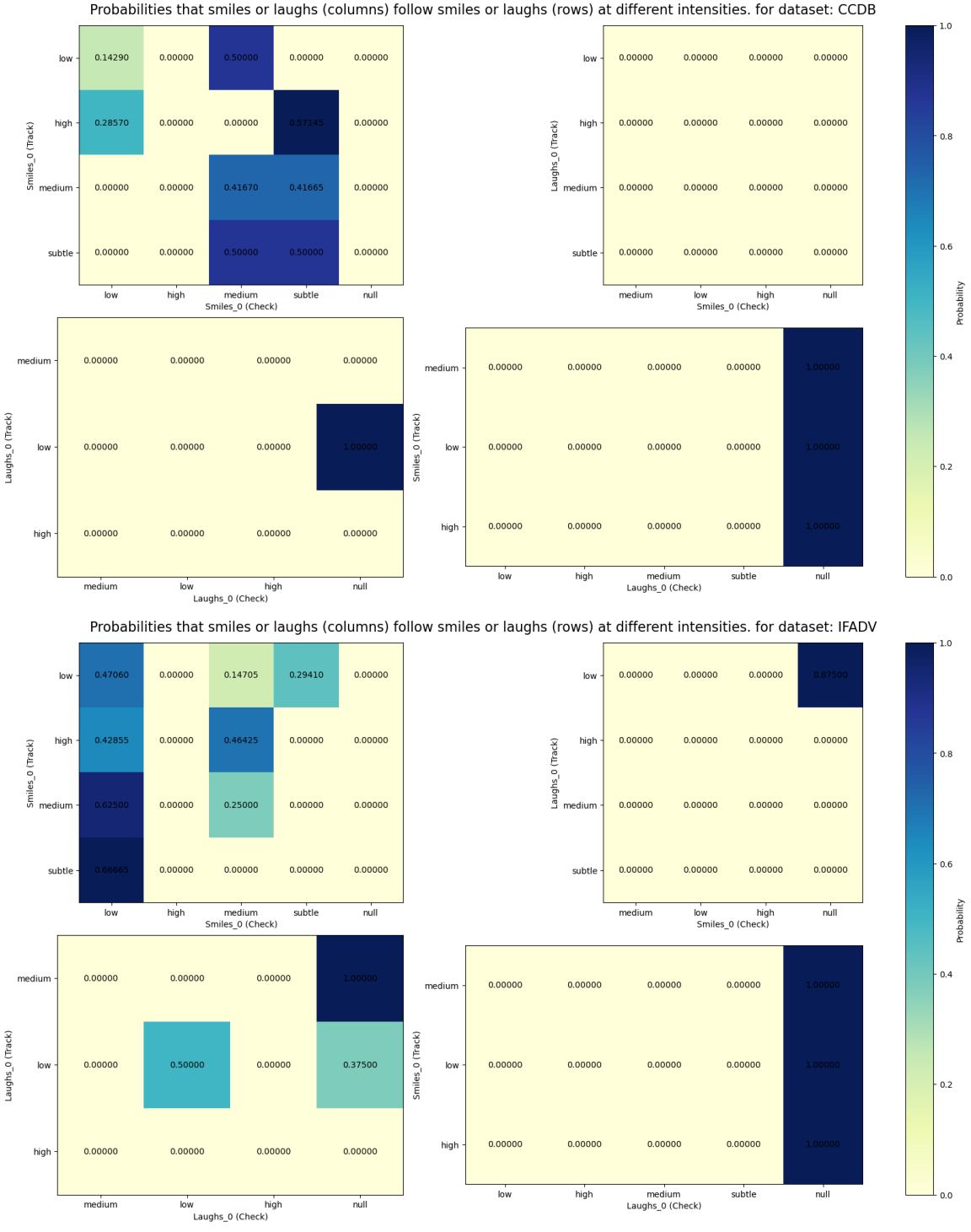


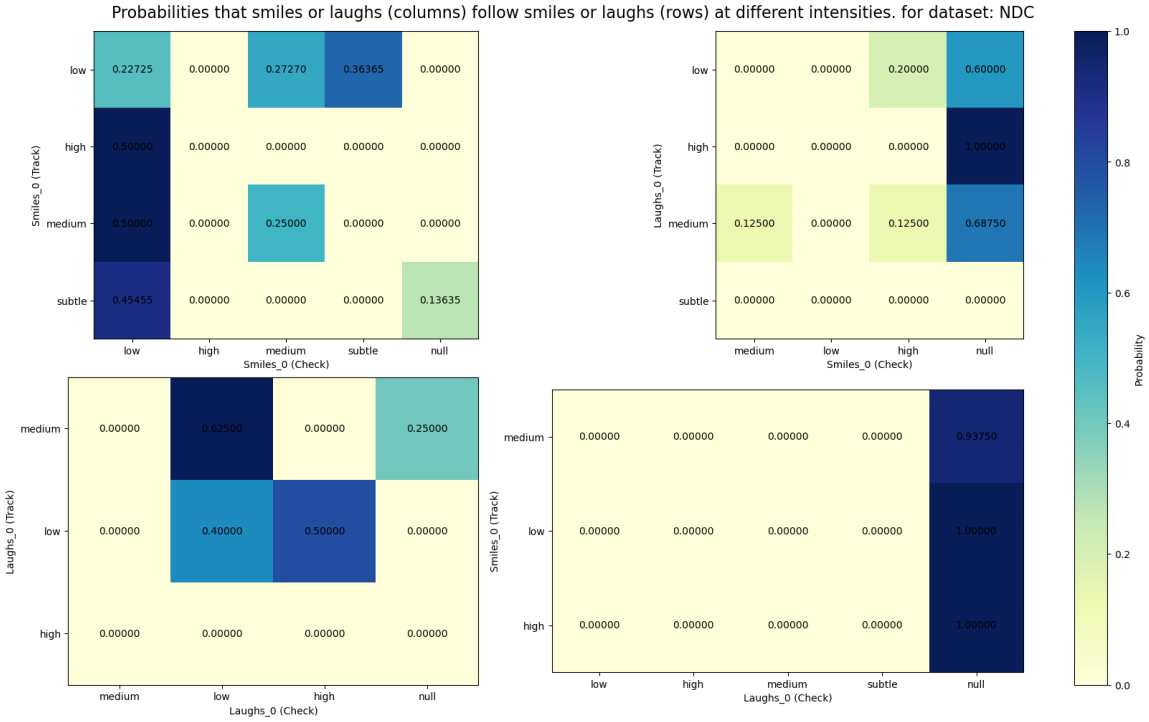
Track of expression for each dataset of each individual:

In [ ]:
```
probabilities_matrix = []
moyenne_prob_interaction = 0
# Define the pairs of expressions for the heatmaps
expression_pairs = [("Smiles_0", "Smiles_0"), ("Smiles_0", "Laughs_0"), ("Laughs

# Iterate over the datasets
for dataset_index, database in enumerate(databases_name):
    # Create a new figure and axes for the current dataset
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 10), constrained_layc

    # Track the current row and column index
    current_row = 0
    current_col = 0
```

```
    # Iterate over the expression pairs for the heatmaps
    for pair_index, (expression_choiceA, expression_choiceB) in enumerate(expres
        # Get the appropriate intensity labels based on the expressions
        intensity_labelsT = copy.deepcopy(smiles_intensities) if expression_choi
        intensity_labelsC = copy.deepcopy(smiles_intensities) if expression_choi
        intensity_labelsC.append("null")

        # Create a 2x2 matrix with zeros
        probabilities_matrix_intensity = np.zeros((len(intensity_labelsT), len(i

        list_mimicry_TC = expression_track_byI(expression_choiceA, expression_ch
        list_mimicry_TC_prev = list_mimicry_TC[0]  # Previous expression DataFra
        list_mimicry_TC_next = list_mimicry_TC[1]  # Next expression DataFrame

        # Create the heatmap for combined expressions
        ax = axs[current_row, current_col]
        for i, intensityC in enumerate(intensity_labelsC):
            for j, intensityT in enumerate(intensity_labelsT):
                try :
                    filtered_list_prev = list_mimicry_TC_prev[
                        (list_mimicry_TC_prev['Intensityp'] == intensityC) &
                        (list_mimicry_TC_prev[f'Current_level_{expression_choice
                    ]
                    percentage_prev2 = filtered_list_prev['Percentagep'].values[

                    filtered_list_next = list_mimicry_TC_next[
                        (list_mimicry_TC_next['Intensityf'] == intensityC) &
                        (list_mimicry_TC_next[f'Current_level_{expression_choice
                    ]
                    percentage_next2 = filtered_list_next['Percentagef'].values[
                    # Average the percentages for previous and next expressions
                    combined_percentage = (percentage_prev2 + percentage_next2)
                    probabilities_matrix_intensity[j, i] = combined_percentage /
                except:
                    pass

        # Plot the heatmap
        im = ax.imshow(probabilities_matrix_intensity, cmap='YlGnBu', interpolat
        # Add the text for each cell
        for i in range(len(intensity_labelsC)):
            for j in range(len(intensity_labelsT)):
                text = ax.text(i, j, f"{probabilities_matrix_intensity[j, i]:.5f

        # Set the title for the current heatmap
        ax.set_xticks(range(len(intensity_labelsC)))
        ax.set_yticks(range(len(intensity_labelsT)))
        ax.set_xticklabels(intensity_labelsC)
        ax.set_yticklabels(intensity_labelsT)
        ax.set_xlabel(f"{expression_choiceA} (Check)")
        ax.set_ylabel(f"{expression_choiceB} (Track)")

        # Update the current row and column index
        current_col += 1
        if current_col == 2:
            current_row += 1
            current_col = 0

    # Add a common colorbar for the heatmaps of each database
    fig.colorbar(im, ax=axs, label='Probability')
```

```
# Set the title for the figure based on the dataset
fig.suptitle(f"Probabilities that smiles or laughs (columns) follow smiles o

# Show the figure
plt.show()
```





Results:

We first note that, in general, smiles are mostly followed by smiles. For NDC, we see that smiles are also followed by laughs. We can see that laughs are rarely followed by no laughs and mostly followed by laughs of similar levels. Due to the rare occurrence of laughs of high levels compared to lower levels, we rarely observe laughs following higher levels of smiles. But we can observe with high probability, laughs with higher levels follow smiles with higher levels. When smiles follow smiles, we can see that the lower levels (subtle and low) are mostly followed by the levels directly above them (low and medium respectively) and the higher levels (medium and high) by the ones directly below them (medium and low respectively, even subtle). When laughs follow laughs, the successive laughs have similar levels or higher. The smile-laugh continuum suggests that S&L can be represented on the same scale. S&L (especially laughs) rarely occur without any surrounding S&L, and that a relationship exists between the S&L intensity levels when these form sequences. In all cases, the most probable levels following other levels are the closest ones. There is thus, in a sequence of S&L, no sudden variation across levels but rather gradual variation.

4-Questions studied

Mimicry:

Is the mimicry recurring? (For example, a smile always leads to a smile?)

- Here are the main observations concerning the recurrence of mimicry:
  - For person B who imitates person A (A/B):
    - In general, smiles are mainly mimicked by smiles, while laughs are mimicked by laughs and smiles.

        - Smiles imitating smiles are of the same intensity or greater intensity.
        - Laughs imitating smiles have a lower intensity.
        - Smiles imitating laughs are mainly of lower intensity, but the probabilities are very low (lack of data?).
        - Laughs imitating laughs have the same intensity or a lower intensity.
    - For person A who imitates person B:
        - Smiles are mainly mimicked by smiles, just like laughs mimicked by smiles (except for NDC where laughs are also mimicked by smiles and laughs).
        - Smiles mimicking smiles have similar or lower intensities.
        - Laughs imitating smiles have a lower intensity.
        - Smiles imitating laughs are generally of higher intensity.
        - Laughs imitating laughs have the same intensity or a higher intensity.
- Overall, there is a trend of recurrence of mimicry, with smiles being mostly mimicked as smiles and laughs being mostly mimicked as laughs. However, there are variations depending on the role of the speaker and the listener.

- It is also interesting to note that, in many cases, the role of the speaker has an influence on mimicry. For example, the speaker tends to imitate the listener's smiles more frequently with laughs, while the listener imitates less frequently or with lower intensity.

Are laughs mostly mimicked by laughs? Would the smiles be mimicked by both?

- We can observe the following patterns:

    - For the person B mimicking person A (A/B):
        - Laughs are mostly mimicked by laughs.
        - Smiles are mimicked by both smiles and laughs, although the intensity of smiles mimicking laughs is mostly lower.
    - For the person A mimicking person B:
        - Laughs are mimicked by both laughs and smiles.

      -Smiles are mimicked by both smiles and laughs, with the intensity of smiles mimicking laughs generally being higher.
- Therefore, we can conclude that laughs are mostly mimicked by laughs, while smiles can be mimicked by both smiles and laughs.

When laughs mimic smiles, are laughs levels lower? When smiles mimic laughs, are smile levels higher?

- We can make the following observations regarding the levels of laughs and smiles when they mimic each other:

    - When laughs mimic smiles:
        - In general, the intensity of laughs mimicking smiles is lower.
        - For CCDB, laughs mimicking smiles tend to have a lower intensity.
        - For IFADV, laughs mimicking smiles are mostly low laughs.
        - For NDC, laughs mimicking smiles have the same or lower intensity.
    - When smiles mimic laughs:

        - In general, the intensity of smiles mimicking laughs is higher.
        - For CCDB, smiles mimicking laughs are mostly of lower intensity, but there are small probabilities for this category.
        - For IFADV, smiles mimicking laughs are mostly of the same intensity or higher.
        - For NDC, smiles mimicking laughs are mostly higher in intensity.
- Therefore, we can conclude that when laughs mimic smiles, the levels of laughs are generally lower. On the other hand, when smiles mimic laughs, the levels of smiles are generally higher. These findings support the concept of a smile-laugh continuum, where smiles are on the low arousal side and laughs are on the high arousal side of a common arousal level scale for both speaker and listener.

When the listener imitates the speaker, are levels of laughs imitation lower?

- We can observe the following patterns when the listener imitates the speaker:
    - laughs imitation by the listener:
        - In general, the levels of laughs imitation by the listener are lower compared to when the speaker imitates the listener.
        - For CCDB, when laughs mimic smiles, the speaker (listener imitating the speaker) mimics more frequently and with lower intensity laughs.
        - For IFADV, the role of the listener has less impact, but the probabilities indicate that the listener tends to mimic the speaker's laughs.
        - For NDC, the listener imitates the speaker's laughs less frequently or with lower intensity, except for laughs mimicking laughs and high smiles mimicking subtle smiles.
- Therefore, we can conclude that when the listener imitates the speaker, the levels of laughs imitation are generally lower. This could be due to factors such as the listener producing fake laughs or imitating with lower intensity compared to the speaker.

Is there an influence of the role on the imitation of expressions?

- We can observe some influence of the role (speaker or listener) on the imitation of expressions. Here are the observations:
    - Influence of the role on the speaker's imitation of the listener:
        - In general, the speaker tends to mimic the listener more frequently.
        - The intensity of mimicry by the speaker is often the same or higher compared to the original expression of the listener.
        - The speaker tends to mimic the listener's smiles and laughs with laughs, and the intensity is generally similar or higher.
        - The speaker mimics the listener's expressions, particularly smiles, more frequently with laughs.
    - Influence of the role on the listener's imitation of the speaker:
        - The listener tends to mimic the speaker's expressions, particularly smiles, more frequently.
        - The intensity of mimicry by the listener is sometimes lower compared to the original expression of the speaker.

    - The listener mimics the speaker's smiles with laughs more frequently.
    - The listener imitates the speaker's expressions, particularly laughs, with less frequency or lower intensity.
- Therefore, we can conclude that the role does have an influence on the imitation of expressions. The speaker tends to mimic the listener more frequently and with similar or higher intensity, while the listener imitates the speaker, especially with smiles, more frequently but sometimes with lower intensity. These observations suggest that the speaker's expressions may have a stronger influence on the listener's imitation.

Dynamics of intra/inter influences:

How do intra and inter influences combine to shape facial expressions in social interactions?

- The data suggests that there are both intra-influences (within an individual's own sequence of expressions) and inter-influences (between the expressions of the speaker and listener) that shape facial expressions in social interactions.
- Intra-influences can be seen in the patterns of smiles being mostly followed by smiles and laughs being rarely followed by no laughs but mostly followed by laughs of similar levels.
    - Intra-influences refer to the influence of an individual's own sequence of expressions on subsequent expressions.
    - The data shows that smiles are mostly followed by smiles within an individual's own sequence, indicating a self-reinforcing pattern of positive affect.
    - Similarly, laughs are rarely followed by no laughs, suggesting a tendency for laughs to be sustained or continued in one's own expression sequence.
    - These intra-influences contribute to the coherence and continuity of an individual's emotional expression, as subsequent expressions tend to align with and reinforce the preceding expressions.
- Inter-influences can be observed when smiles of one participant (e.g., the speaker) are followed by laughs of another participant (e.g., the listener), indicating some level of synchronization or response between the expressions of the speaker and listener.
    - Inter-influences refer to the influence between the expressions of different participants in a social interaction, such as the speaker and listener.
    - The data indicates that there is synchronization or response between the expressions of the speaker and listener.
    - Specifically, smiles of the speaker are followed by laughs of the listener, suggesting a reciprocal interaction where the listener responds to the speaker's positive affect with laughs.
    - This inter-influence reflects the interpersonal dynamics and emotional contagion within social interactions, as individuals tend to mimic and synchronize their expressions with each other.
- Mutual Adaptation:

    - The combined effect of intra and inter influences leads to a process of mutual adaptation in facial expressions during social interactions.
    - Intra-influences shape an individual's own expression sequence, reinforcing certain emotional states and patterns.
    - Inter-influences between participants further shape and modulate facial expressions, creating a feedback loop of mutual adaptation.
    - This mutual adaptation enhances the interpersonal connection and emotional resonance between individuals, as they respond and adapt to each other's expressions.
- Co-Construction of Emotional Dynamics:
    - The combination of intra and inter influences contributes to the co-construction of emotional dynamics in social interactions.
    - Intra-influences establish a foundation for individual emotional expression patterns, while inter-influences introduce variability and responsiveness based on the interaction with others.
    - Together, these influences shape the unfolding emotional expressions, creating a dynamic and reciprocal exchange of affective cues between participants.
- In summary, intra and inter influences combine to shape facial expressions in social interactions. Intra-influences contribute to the coherence and continuity of an individual's own expression sequence, while inter-influences reflect the synchronization and response between the expressions of participants. This mutual adaptation and co-construction of emotional dynamics enhance interpersonal connection, emotional resonance, and the shared experience of social interactions.

Do the expression sequences influence each other?

- Yes, the expression sequences do influence each other. The data indicates that the occurrence and intensity levels of one expression can influence the occurrence and intensity levels of the following expression.
- For example, laughs with higher levels are more likely to follow smiles with higher levels, suggesting a relationship between the intensity levels of smiles and laughs in the sequence.
    - Influence of Smile Sequences:
        - The data suggests that smile sequences influence the subsequent expressions, particularly laughs.
        - When smiles of higher levels occur in a sequence, they are more likely to be followed by laughs with higher levels. This indicates a relationship between the intensity levels of smiles and laughs in the sequence.
        - The occurrence of a higher intensity smile may elicit a stronger emotional response, leading to a corresponding increase in the intensity of the subsequent laugh.
    - Gradual Variation and Continuum:
        - The data shows that expression sequences exhibit a gradual variation in intensity levels, with lower levels often followed by slightly higher levels, and vice versa.

        - This gradual variation aligns with the concept of the smile-laugh continuum, where smiles are on the low arousal side and laughs are on the high arousal side of a common intensity scale.
        - The influence of expression sequences lies in maintaining the continuity and smooth progression of emotional states, as subsequent expressions tend to align with the intensity levels of preceding expressions.
    - Reciprocal Influence:
        - Expression sequences exhibit a reciprocal influence, with each expression influencing the occurrence and intensity of the subsequent expression.
        - For example, a smile of a particular intensity level can influence the subsequent laugh by either increasing or decreasing its intensity level, depending on the context and the emotional response it elicits.
        - Similarly, the intensity level of a laugh can be influenced by the preceding smile, leading to a continuation or modulation of the emotional response.
    - Emotional Contagion and Synchronization:
        - The influence of expression sequences on each other is closely related to emotional contagion and synchronization.
        - Emotional contagion refers to the tendency of individuals to mimic and adopt the emotional states of others. In this case, the observed expressions in a sequence can elicit similar expressions in the observer, creating a feedback loop of emotional synchronization.
        - The influence of expression sequences contributes to a shared emotional experience and enhances the interpersonal connection between individuals.
- In summary, the data suggests that expression sequences exert an influence on each other. The occurrence and intensity levels of one expression, such as smiles, can impact the subsequent expression, such as laughs. This reciprocal influence contributes to the gradual variation and continuity of emotional states in the sequence. It plays a role in emotional contagion, synchronization, and the establishment of shared emotional experiences during social interactions.

Relationship of intensities:

What is the impact of intensities in facial expression sequences? in mimicry ?

- The data suggests that the intensity levels of facial expressions have an impact on the subsequent expressions in the sequence.
- In general, there is a gradual variation in intensity levels within the sequence, following a pattern where lower levels are mostly followed by slightly higher levels and higher levels are mostly followed by slightly lower levels.
- This pattern supports the concept of a smile-laugh continuum, where there is a common arousal level scale for both smiles and laughs. The intensities in facial expression sequences contribute to the overall dynamics and flow of the interaction.
    - Matching Intensities in Mimicry:
        - The data suggests that there is a relationship between the intensity levels of facial expressions in sequences, with subsequent expressions often

matching or aligning with the intensity of preceding expressions.
    - When mimicking facial expressions, individuals tend to match the intensity level of the observed expression. For example, higher intensity smiles are more likely to be followed by higher intensity mimicry, and laughs with higher levels are more likely to follow smiles with higher levels.
    - Matching intensities in mimicry contributes to a more accurate representation of the emotional state and reinforces the perceived connection between individuals.
  - Gradual Variation and Continuum of Intensities:
    - The data indicates that facial expression sequences exhibit a gradual variation of intensity levels, where lower levels are mostly followed by slightly higher levels and higher levels are mostly followed by slightly lower levels.
    - This pattern aligns with the concept of a smile-laugh continuum, where smiles are on the low arousal side and laughs are on the high arousal side of a common intensity scale for both S&L.
    - Mimicking gradual variations in intensity levels helps maintain the smooth flow and natural progression of expressions, contributing to the overall coherence and authenticity of the interaction.
  - Emotional Synchronization and Empathy:
    - Mimicry of intensities in facial expression sequences facilitates emotional synchronization and empathy between individuals.
    - When individuals accurately mimic the intensity levels of facial expressions, it enhances the perception of shared emotional experiences and fosters a deeper sense of understanding and connection.
    - By mirroring the intensity of the observed expressions, individuals demonstrate empathy and create a shared emotional space, leading to stronger emotional bonds and social cohesion.
  - Influence on Social Dynamics:
    - Intensity-matched mimicry in facial expression sequences influences the dynamics of social interactions.
    - Matching the intensity levels of facial expressions helps establish a mutual rhythm and synchrony between individuals, creating a sense of resonance and rapport.
    - Intensity-matched mimicry can enhance communication effectiveness, as it conveys the appropriate level of emotional response and facilitates mutual comprehension and alignment of emotional states.
- In summary, the impact of intensities in facial expression sequences, as observed in the data, is highly relevant in the context of mimicry. Matching intensities during mimicry promotes emotional synchronization, empathy, and social bonding. It contributes to the natural progression and coherence of expressions, enabling effective communication and fostering a deeper connection between individuals.

Overall, the data indicates that both intra-influences and inter-influences play a role in shaping facial expressions during social interactions. The sequential patterns, intensity

levels, and mutual influences between expressions contribute to the dynamics and coordination of facial expressions in social communication.