figure1

November 10, 2021

1 Figure1: Example sentence illustrating measures used in GLM analysis

This notebook contains all code used to set up figure 1, which illustrates the measures used in GLM analysis.

Measures of interest: next-word entropy

surprisal

Measures of no interest: word frequency

instantaneous visual change (IVC), contolling for actor movement

audio envelope, controlling for low-level auditory effects

1.1 Importing Modules and Datasets

Several dataframes will need to be set up.

d will be the complete word dataset

s will contain only word data for our sequence of interest

a will contain relevant data for the audio envelope

m will contain relevant data for the IVC (movement) envelope

[1]:
```
#Imports
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import cv2 #computer vision
import os #directories
import matplotlib.image as mpimg #matplotlib for images
import math
import matplotlib.patches as mpatches #matplotlib for legend
import matplotlib.gridspec as gridspec #advanced subplots
```

[ ]:
```
workingDirectory = os.getcwd()
workingDirectory #= 'C:/Users/gffun/OneDrive/Desktop/projects/
 ↪gesture-processing-figures'
```

Here, the dataset containing a part of the story is loaded into a variable d as a pandas dataframe. Each word has been included as a row with each column representing another measure. It is subsequently subsampled to include only the content words of the storylet (isC) and then further split into two smaller dataframes only containing a subset of content words that have been coded as either a "key" (K) or "nonkey" (NK) words. In this nomenclaure, all "key" words are accompanied by a gesture and all "nonkey" words are not. These three subsampled dataframes will later be used to construct histograms conveying the distribution of entropy and surprisal across these conditions.

Additionally, the data containing IVC and audio envelopes, form which confound regressors were calculated are loaded here.

[53]:
```
os.chdir(workingDirectory)

# loading data
d = pd.read_csv('data/samirdata_XF_1.csv')

# loading ivc and auditoy envelopes, make sure to choose the appropriate␣
 ↪storylet 01-16
ivc_env = pd.read_csv('data/04_Samir_blieb_IVC.csv')
aud_env = pd.read_csv('data/04_audio_envelope.csv')

# sub-dataframes for histograms
isC = d.loc[d.isC == 1].reset_index()[['word', 'Entropy_Roark',␣
 ↪'Surprisal_Roark','is_key', 'isC', 'is_key_and_C']]
isC_K = d.loc[d.is_key_and_C == 1].reset_index()[['word', 'Entropy_Roark',␣
 ↪'Surprisal_Roark','is_key', 'isC', 'is_key_and_C']]
isC_NK = d.loc[(d.isC == 1) & (d.is_key != 1)].reset_index()[['word',␣
 ↪'Entropy_Roark', 'Surprisal_Roark','is_key', 'isC', 'is_key_and_C']]
```

[54]:
```
#extracting columnns of interest from d
d = d[['word', 'freq', 'ivc_avg', 'Entropy_Roark', 'Surprisal_Roark',
       'env_avg','is_key','xmax', 'xmin', 'entropy']]

#adding average time column per word to d
d['xavg'] = (d['xmax']+d['xmin'])/2
```

Now, the sequence of words that will later be displayed in the figure is chosen. This can be any sequence of words from the storylet.

CONTINUE HERE

[56]:
```
#set s, can be any sequence of words from the desired storylet
s = d.iloc[614:624].reset_index()
s['freq'] = s['freq'].replace([0.0],0.2)
```

```
# new timing columns starting at 0
s['xavg0'] = s['xavg'] - s['xmin'].min()
s['xmin0'] = s['xmin'] - s['xmin'].min()
s['xmax0'] = s['xmax'] - s['xmin'].min()
#s
```

[57]:
```
#getting ivc up to some index according to our desired sentence
m = ivc_env.loc[s['xmin'].min() * 100 : s['xmax'].max()*100].reset_index()
m['time'] = m['time'] - s['xmin'].min() #correcting time to start at 0

#rolling average
window_size = 5
m['rolling_avg'] = m['ivc_standarized'].rolling(window_size).mean()
#replacing window-size nans at start
m['rolling_avg'] =  m['ivc_standarized'][:window_size-1].append(␣
 ↪m['rolling_avg'][window_size-1:])
#m
```

[60]:
```
#getting audio env up to some index according to our desired sentence
a = aud_env.loc[s['xmin'].min() * 100 : s['xmax'].max()*100].reset_index()
a['time'] = a['time'] - s['xmin'].min() #correcting time to start at 0

# rolling average
window_size = 5
a['rolling_avg'] = a['envelope_standardized'].rolling(window_size).mean()
# replacing window-size nans at start
a['rolling_avg'] =  a['envelope_standardized'][:window_size-1].append(␣
 ↪a['rolling_avg'][window_size-1:])
#a
```

1.2 Extracting and Cropping Frames of interest

For displaying the stimuli shown in the scanner, evenly spaced videoframes will get extracted from the video, cropped according to some cropping parameters and saved into the outputDirectory.

Then, they will be loaded, appended and plotted.

[45]:
```
#cropping parameters
y = 75
x = 260
h = 500
w = 230

#even spacing of frames in time
n = 13 # number of frames
frametimes = np.arange(s.xmin.min(), s.xmax.max(), s.xmax0.max()/n) * 1000
```

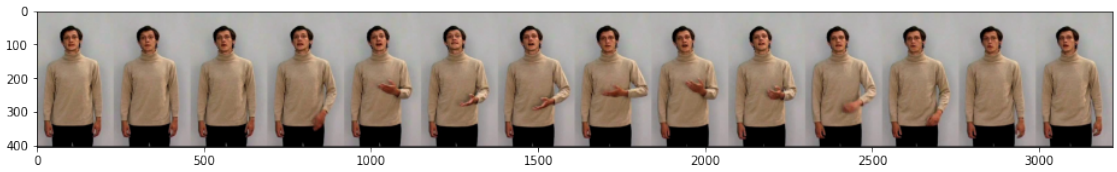
```
#set output directory
outputDirectory = workingDirectory + '/framesCropped'

# initialize capture object with cv2
# make sure to include the appropriate file 01-16
cap = cv2.VideoCapture('data/04_Samir_blieb.mpeg')

# initialize output directory, if not already existing
if not os.path.exists(outputDirectory):
    os.mkdir(outputDirectory)

os.chdir(outputDirectory)

# clear output directory
for f in os.listdir(outputDirectory):
    os.remove(os.path.join(outputDirectory, f))

# get, crop and save frames
for i in range(len(frametimes)):
    cap.set(0, frametimes[i]) # set msec of video here
    ret, frame = cap.read() # get frame
    crop_img = frame[y:y+h, x:x+w] # crop frame
    if i < 10:
        cv2.imwrite('frame0'+str(i)+'.jpeg', crop_img) # save frame
    else:
        cv2.imwrite('frame'+str(i)+'.jpeg', crop_img) # save frame

os.chdir(workingDirectory)
```

[45]:
```
"\nwhile(cap.isOpened()):\n    ret, frame = cap.read()\n\n    #gray =
cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)\n    crop_img = frame[y:y+h, x:x+w]\n
\n    cv2.imshow('frame',crop_img)\n    \n    if cv2.waitKey(0) & 0xFF ==
ord('q'): #waitKey 0 waits indefinately for key press to show next pic\n
break\n"
```

[51]:
```
# loading and appending frames
#outputDirectory = workingDirectory + '/framesCropped'
os.chdir(outputDirectory)

framelist = [f for f in os.listdir(outputDirectory)] # get frames from outputdir
frames = [mpimg.imread(frame) for frame in framelist] # load all frames

os.chdir(workingDirectory)

# append frames, this might be done more elegantly
longframe = frames[0]
```

```
for frame in frames[1:]:
    longframe = np.append(longframe,frame, axis = 1)

# plotting frames
fig, ax = plt.subplots(1,1, figsize = (16,5))
plt.imshow(longframe)
```

[51]: <matplotlib.image.AxesImage at 0x25db5473a08>



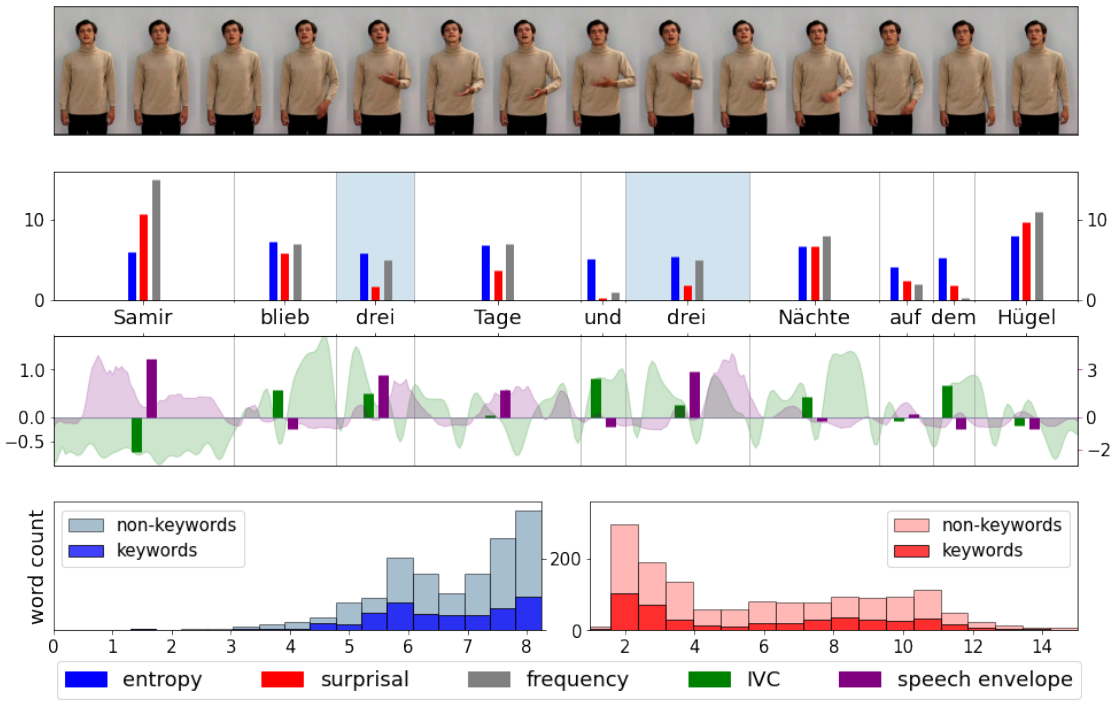
1.3 Plotting

[61]:
```
# setting up subplot structures
fig = plt.figure(figsize = (16,10))

# gridspec tool
gs = fig.add_gridspec(4, 2, hspace = 0.28, wspace = 0.1)

# 5 axes
ax1 = fig.add_subplot(gs[0, :]) # frames
ax2 = fig.add_subplot(gs[1, :]) # word-wise entropy, surprisal, frequency
ax3 = fig.add_subplot(gs[2, :]) # ivc and audio envelopes
ax4 = fig.add_subplot(gs[3, 0]) # entropy distribution histogram
ax5 = fig.add_subplot(gs[3, 1]) # surprisal distribution histogram

# plotting the frames
ax1.imshow(longframe)

# remove ticks for ax1
ax1.tick_params(axis='x', which='both', bottom=False, top=False,␣
 ↪labelbottom=False)
ax1.tick_params(axis='y', which='both', left=False, right=False, labelleft=False)

w1 = 7 # linewidth for entropy, surprisal, frequency
w2 = 9 # linewidth for ivc, audio

# positions for lines relative to word averages
# entropy, surprisal, frequency
left = s['xavg0'] - 0.025
```

```
right = s['xavg0'] + 0.025
middle = s['xavg0']

# ivc, audio
left2 =  s['xavg0'] - 0.04
right2 =  s['xavg0'] + 0.04
middle2 =  s['xavg0']

# plotting lineplots for entropy and surprisal
ax2.vlines(left2, [0], s.iloc[:,4], color = "blue", linewidth=w1) #entropy
ax2.vlines(middle2, [0], s.iloc[:,5], color = "red", linewidth=w1) #surprisal

# setting up twin axis for frequency
axfreq = ax2.twinx()
axfreq.vlines(right2, [0], s.iloc[:,2], color = "grey", linewidth=w1) # frequency

# ax2 cosmetics
ax2.set_xticks(s['xavg0'], minor = False )
ax2.set_xticks(s['xmax0'], minor = True) # onsets/offsets of words
ax2.xaxis.grid(True, which='minor')
ax2.set_xticklabels(s['word'], fontsize = 18)
ax2.tick_params(axis='y', labelsize = 15)
ax2.set_ylim(0, 16)
ax2.axis(xmin= s['xmin0'].min(), xmax= s['xmax0'].max())
axfreq.tick_params(axis='y', labelsize = 15)
axfreq.set_ylim(0, 16)

# plotting lineplots for ivc and audio
ax3.vlines(left, [0], s.iloc[:,3], color = "green", linewidth=w2) # ivc
ax3.vlines(right, [0], s.iloc[:,6], color = "purple", linewidth=w2) # aud

# ax3 cosmetics
ax3.xaxis.tick_top()
ax3.hlines(0, -0.5, len(s)-0.5, linewidth= 0.5, alpha = 0.5) # a line a 0
ax3.set_xticklabels([])
ax3.set_xticks(s['xavg0'], minor = False)
ax3.set_xticks(s['xmin0'], minor = True)
ax3.xaxis.grid(True, which='minor')
ax3.set_ylim(-1, 1.7)
ax3.set_yticks([-0.5, 0, 1], minor = False)
ax3.tick_params(axis='y', labelsize = 15)
ax3.axis(xmin= s['xmin0'].min(), xmax= s['xmax0'].max())

# twin axis for audio envelope rolling average
axenv = ax3.twinx()

# plotting rolling averages
```

```
ax3.fill_between(m['time'], m['rolling_avg'], alpha = 0.2, color = 'green')
axenv.fill_between(a['time'], a['rolling_avg'], alpha = 0.2, color = 'purple')

# axenv cosmetics
axenv.set_ylim(-1*3, 1.7*3)
axenv.set_yticks([-2, 0, 3], minor = False)
axenv.tick_params(axis='y', labelsize = 15, color = 'purple')


# coloring keywords
for index, row in s.iloc[np.where(s['is_key']==1)].iterrows():
    ax2.axvspan(row['xmin0'], row['xmax0'], alpha = 0.2)

# histograms for entropy and surprisal distributions
alpha = 0.5

bins = np.linspace(0, isC_NK['Entropy_Roark'].max(), 20)
bins2 = np.linspace(0, isC_NK['Surprisal_Roark'].max(), 20)

# plot histograms
ax4.hist(isC_NK['Entropy_Roark'], alpha = alpha, color = '#517ea0', bins = bins,␣
 ↪align = 'mid', edgecolor = 'black') # light blue
ax4.hist(isC_K['Entropy_Roark'],  alpha = 0.75, color = 'blue', bins = bins,␣
 ↪align = 'mid', edgecolor = 'black')

ax5.hist(isC_NK['Surprisal_Roark'],  alpha = alpha, color = '#ff726f', bins =␣
 ↪bins2, align = 'mid', edgecolor = 'black') # light red
ax5.hist(isC_K['Surprisal_Roark'], alpha = 0.75, color = 'red', bins = bins2,␣
 ↪align = 'mid', edgecolor = 'black')

# axes 4 and 5 cosmetics
ax4.axis(xmin = 0, xmax=isC_NK['Entropy_Roark'].max())
ax5.axis(xmin = 1, xmax=isC_NK['Surprisal_Roark'].max())

# set ticks
ax4.tick_params(axis='y', labelsize = 15)
ax4.tick_params(axis='x', labelsize = 15)
ax4.set_ylim(0, 360)

ax5.tick_params(axis='y', labelsize = 15)
ax5.tick_params(axis='x', labelsize = 15)
ax5.set_ylim(0, 360)

# set labels
sz = 18
ax4.set_ylabel('word count', size = sz)
ax4.yaxis.tick_right()
```

```
ax4.set_yticklabels([])

# adding legends to axes 4 and 5
ax4.legend(["non-keywords","keywords"],fontsize = 15)
ax5.legend(["non-keywords", "keywords"],fontsize = 15)

# artist proxies for legend
blue_patch = mpatches.Patch(color='blue', label='entropy')
red_patch = mpatches.Patch(color='red', label='surprisal')
grey_patch = mpatches.Patch(color='grey', label='frequency')
green_patch = mpatches.Patch(color='green', label='ICV')
purple_patch = mpatches.Patch(color='purple', label='speech envelope')

# adding legend
fig.legend([blue_patch, red_patch, grey_patch, green_patch,␣
 ↪purple_patch],["entropy", "surprisal",'frequency', "IVC", "speech envelope"],␣
 ↪bbox_to_anchor = (0.12, 0.03, 0.79, .02), ncol = 5, fontsize = 18, loc =␣
 ↪'lower left', mode = 'expand')

#plt.savefig("figure1.png")
```



1.4 Figure 1

An example sentence from the utilized story, illustrating measures used in GLM analysis. Top: frames cropped from the stimulus video in equidistant intervals. Upper center: word-wise measures of interest entropy and surprisal (in bits) and frequency as a measure of no interest (in arbitrary units). Words accompanied by a gesture (keywords) are colored blue. Lower center: z-scored measures of no interest relating to the actor's movement (IVC, left scale) and auditory speech envelope (right scale), both word-wise and as continuous measures (sampling frequency: 100 Hz). For illustrative purposes, a running average with window size 5 for continuous measures was used. Bottom: Distributions of both key- and non-keywords across surprisal and entropy measures.