HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

Technische Grundlagen
der angewandten Informatik

Aufbau eines einfachen Spracherkenners

Benedict Roth, David Kubatzki

Konstanz, 6. Juni 2015

Zusammenfassung (Abstract)

Thema:	Aufbau eines einfachen Spracherkenners	
Autoren:	Benedict Roth	dakuba@htwg-konstanz.de
	David Kubatzki	beroth@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz	mfranz@htwg-konstanz.de
	Jürgen Keppler	juergen.keppler@htwg-konstanz.de
	Martin Miller	martin.miller@htwg-konstanz.de

«In diesem Versuch entwickeln wir einen Spracherkenner. Zuerst nehmen wir ein Spektrum auf und dann vergleichen wir nochmals erstellte Aufnahmen mit diesem Referenzspektrum.»

Inhaltsverzeichnis

Abbildungsverzeichnis

Listingverzeichnis

1

Einleitung

In diesem Versuch wird ein einfacher Spracherkenner aufgebaut, der z.B. zur Steuerung eines Staplers in einem Hochregallager dienen könnte. Es reichen hierzu die Erkennung der vier ein- fachen Befehle ”Hoch”, ”Tief”, ”Links” und ”Rechts”. Der Aufbau des Spracherkenners folgt dem in der Vorlesung beschriebenen Prinzip des Prototyp-Klassifikators. Die zugehörigen Spektren werden mit der Windowing-Methode berechnet.

2

Versuch 1

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

2.1.1 Fragestellung

Im Ersten Versuch sollen wir das Windowing, welches wir in der Vorlesung kennen gelernt haben implementieren.

2.1.2 Messmittel

Als Messmittel dient ein 250g schweres Dynamisches Mikrofon mit einer Tauchspule (engl. moving coil). Es ist Uni-direktional und reagiert auf Frequenzen im Bereich von 70Hz bis 13 KHz und hat eine Sensitivität von -54dB ± 3dB. Die Ausgangs impendanz beträgt 500Ω±30

2.1.3 Messprinzip

Das Tauchspulenmikrofon (auch Tauchspulmikrofon) ist ein elektroakustischer Wandler, der nach dem elektroinduktiven Prinzip des dynamischen Mikrofons arbeitet. Es ist sowohl die Bauform des Druckgradientenmikrofons als auch die des Druckmikrofons üblich. Der Begriff Tauchspulenmikrofon bezieht sich auf die technische Anordnung der Bauele- mente des Wandlers: Bei dem Tauchspulenmikrofon ist die Membran fest mit einer Magnet- Spule verbunden, die durch die Membranbewegung in ein statisches dauermagnetisches Feld „eintaucht“. Siehe auch: Tauchspule. Die relative Bewegung von Spule und Magnetfeld er- zeugt per Induktion die Signalspannung. Diese ist proportional zur Membrangeschwindig- keit. 2 Tauchspulenmikrofone benötigen keine nachträgliche Impedanzanpassung und auch kei- ne Symmetrierung; beides kann allein durch die Dimensionierung und Verschaltung der Spu- le

erreicht werden. Prinzipielle Nachteile: Die Schallwelle muss die Masse der Membran mit der Spule be- wegen und elektrische Arbeit leisten. Tauchspulenmikrofone haben daher eine geringe Emp- findlichkeit und zeigen eine Trägheit im Einschwingverhalten, wodurch feinste Details nicht erfasst werden, was jedoch erwünscht sein kann: Sie liefern ein „erdiges“, kräftiges Klang- bild, hochwertige Modelle werden daher durchaus auch bei Studioaufnahmen verwendet. Tauchspulenmikrofone haben ein nicht so hohes Übertragungsspektrum wie Kondensator- mikrofone und sind aufgrund ihrer geringen Empfindlichkeit für Fernaufnahmen ungeeig- net. Die relativ hohe Masse des Membransystems lässt sie zudem empfindlich auf Körper- schall, etwa Hantierungsgeräusche, reagieren; um solche Störungen zu verringern, ist bei hochwertigen Tauchspulenmikrofonen die gesamte technische Einheit (die Mikrofonkapsel) im Mikrofongehäuse schwingfähig gelagert. Die Vorteile dieses Mikrofontyps zeigen sich darin, dass sie in der Regel gegenüber me- chanischen Belastungen recht robust sind und hohe Schalldrücke vertragen. Auch benötigen sie keine Spannungsversorgung, was im mobilen Betrieb von Vorteil sein kann. Die einfache Bauart erlaubt preisgünstige Fertigung und macht diesen Mikrofontyp nahezu unverwüstlich.

2.1.4 Aufbau

Das Mikrofon ist mithilfe eines Audiokabels mit der Soundkarte des Laborrechners verbunden.

2.2 Messwerte

Zuerst schreiben wir uns ein python skript, in welchenm wir mithilfe des Moduls Pyaudio ein Signal aufnehmen. Zur Hilfe haben wir hier das Skript von J. Keppler aus der Moodle einführung. Dieses Signal speichern wir dann mit der Funktion numpy.save() ab. Nun erweitern wir das Programm noch um eine Triggerfunktion. Wir haben uns dafür entschieden, dass das Signal erst ab einem bestimmten Amplitudenwert aufnimmt. So ist sichergestellt, dass erst sobald man in das Mikrofon reinspricht, das Programm auch mit der Aufnahme beginnt. Nun stellen wir die Aufnahmelänge noch auf eine Sekunde ein.

2.3 Auswertung

Nachdem wir das Signal aufgenommen haben, errechnen wir nun noch mithilfe der Funktion numpy.fft.fft() das Amplitudenspektrum. Nun wenden wir noch das Windowing, welches

wir in der Vorlesung kennengelernt haben an. Hierzu zerlegen wir das Signal in kleine Fenster von einer Länge von 512 samples. Diese Fenster müssen sich auch überlappen, hierzu nehmen wir die Hälfte des Vorherigen bzw nächsten Fensters. Auf das jeweilige Fenster multiplizieren wir jetzt noch die Gaußchen Fensterfunktion bei welcher die Fensterbreite der 4fachen Standardabweichung entspricht Hieraus berechnen wir nun wieder das Spektrum.

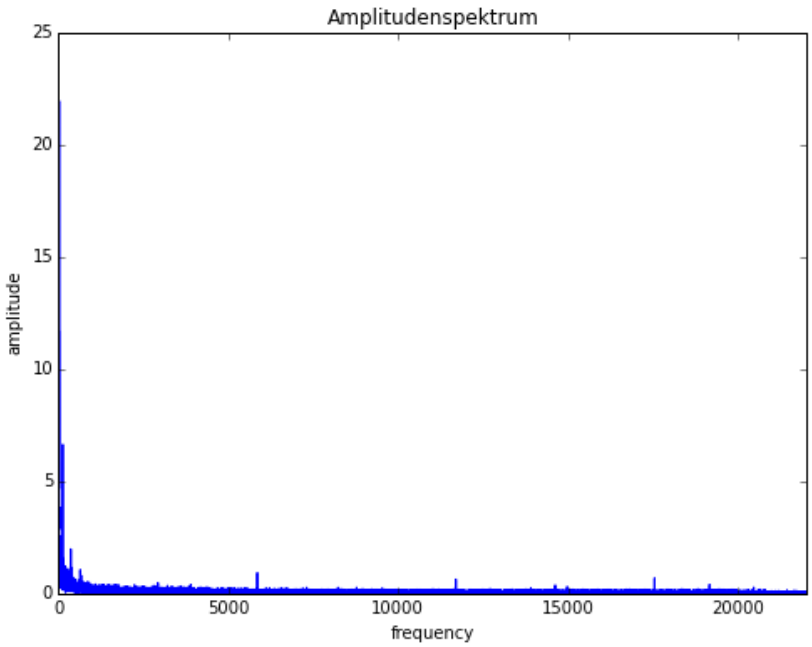


Abbildung 2.1: Amplitudenspektrum

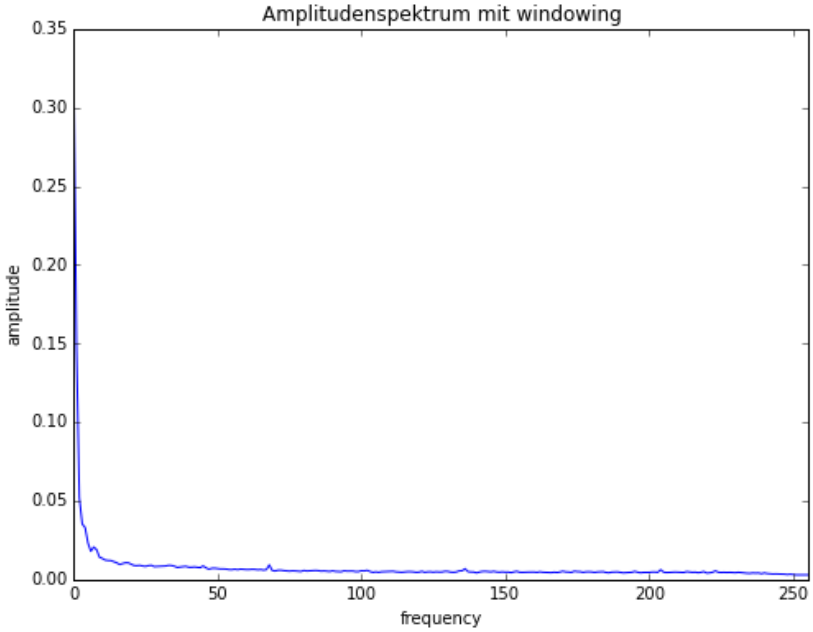


Abbildung 2.2: Windowing

2.4 Interpretation

Nun haben wir eine Aufnahmefunktion, um die Referenzen für Aufgabe 2 aufzunehmen. Nebenbei haben wir noch das Windowing angewandt, mithilfe des Windowing bekommen wir eine genauere Auflösung des Frequenzbandes. Siehe Komplementarität von Frequenz und Zeit.

3

Versuch 2

3.1 Fragestellung, Messprinzip, Aufbau, Messmittel

3.1.1 Fragestellung

In Versuch 2 erstellen wir nun die Spektren für die Wörter Hoch, Tief, Links, Rechts. Anschließend nehmen wir noch 5 Beispiele für diese 4 Wörter auf und vergleichen diese mit den Spektren. Anschließend nimmt ein anderer Sprecher nochmals je 5 mal die 4 Wörter auf und diese werden anschließend wieder mit dem Spektrum verglichen.

3.1.2 Messprinzip

Das Messprinzip ist das gleiche wie in Versuch 1.

3.1.3 Aufbau

Der Aufbau entspricht auch dem von Versuch 1.

3.1.4 Messmittel

Messmittel ist wieder das Mikrofon aus Versuch 1.

3.2 Messwerte

Um eine Referenz für jedes der Wörter Hoch, Runter, Links, Rechts zu erstellen nehmen wir jetzt jeweils 5 mal das jeweilige Wort mithilfe des Aufnahmeskriptes von Aufgabe 1

auf. Danach werden nochmals 5 mal das jeweilige Wort aufgenommen um zu verhindern, dass der Spracherkenner zu gut funktioniert. Anschließend nimmt ein anderer Sprecher auch nochmal 5 mal das jeweilige Wort auf.

3.3 Auswertung

Für jede der 5 Aufnahmen errechnen wir nun das Spektrum und ermitteln aus diesem dann den Mittelwert und erhalten so unsere Referenz. Nun schreiben wir uns noch eine Funktion um die Korrelation zu ermitteln. Hierzu nutzen wir die Funktion scipy.stats.pearsonr(). Nun vergleichen wir welches Spektrum die Höchste Korrelation mit der jeweiligen Aufnahme hat und können so ermitteln um welches Wort es sich handelt.

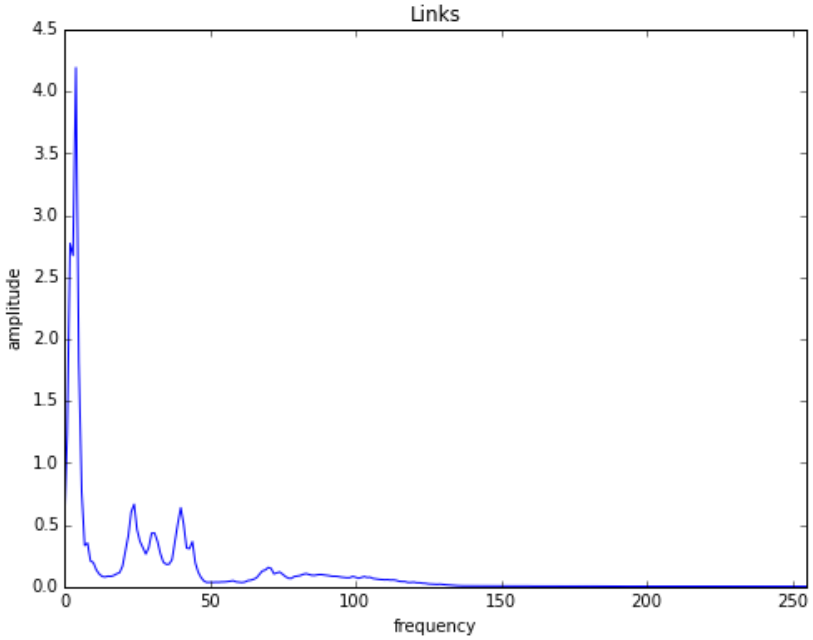


Abbildung 3.1: Links

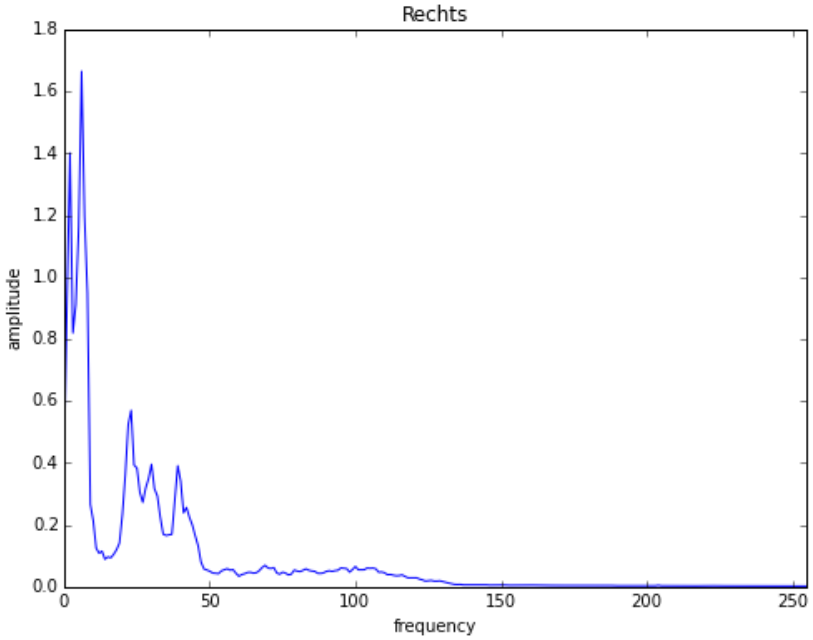


Abbildung 3.2: Recht

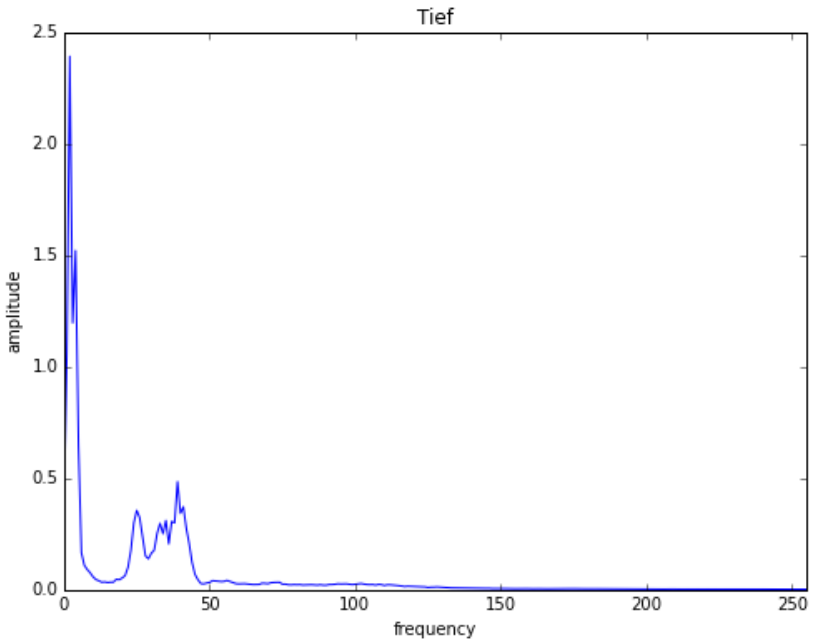


Abbildung 3.3: Tief

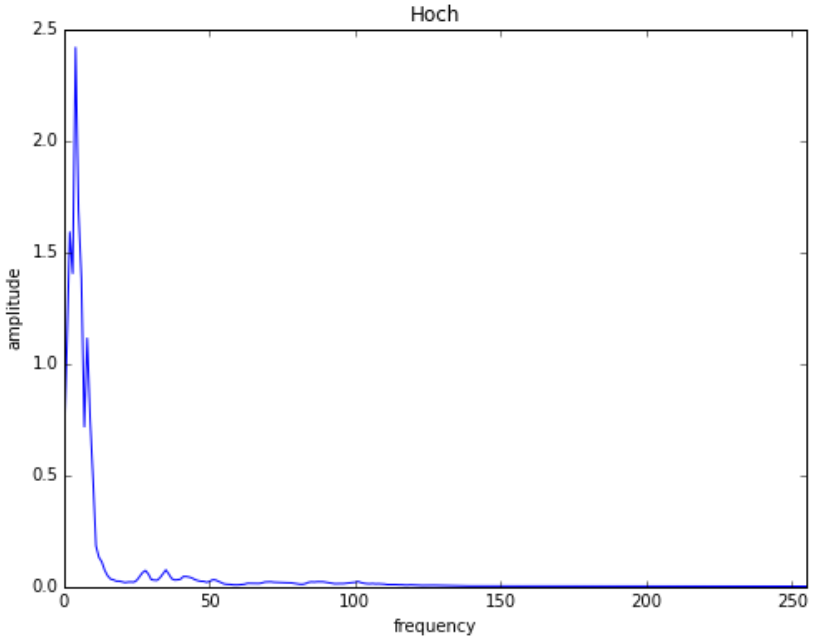


Abbildung 3.4: Hoch

3.4 Interpretation

Wir haben nun einen Funktionstüchtigen Spracherkenner. Mithilfe der Korrelation können wir nun die gesprochenen Wörter ermitteln. So kommen wir auf eine übereinstimmung von 50% bei dem Sprecher der die Referenz eingesprochen hat und auf 55% beim anderen Sprecher. Der Grund für die Ungenauigkeit könnte darin bestehen dass sich die Spektren der einzelnen Wörter, insbesondere Links und Rechts sehr ähneln.

Anhang

A.1 Quellcode

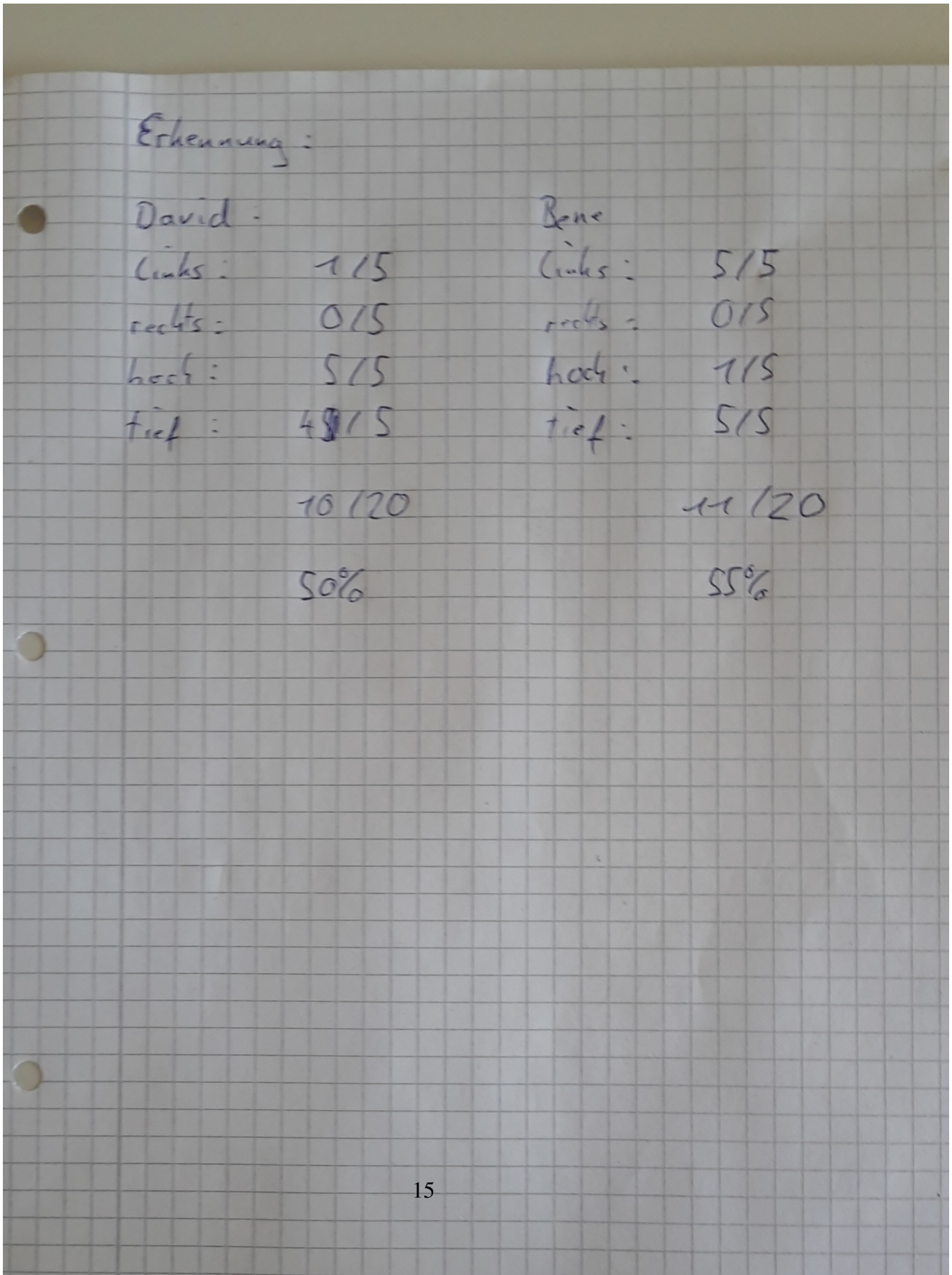
```
#-*-coding:utf-8-*-
"""
SpyderEditor

Thisisatemporaryscriptfile.
"""

import pyaudio
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal
import scipy.stats

FORMAT=pyaudio.paFloat32
SAMPLEFREQ=44100
FRAMESIZE=1024
NOFFRAMES=int(SAMPLEFREQ/FRAMESIZE)
WINDOW=512
def aufnahme(name):
    p=pyaudio.PyAudio()
    print('running')
    stream=p.open(format=FORMAT, channels=1, rate=SAMPLEFREQ,
    input=True, frames_per_buffer=FRAMESIZE)
    reg = True
```

```
    while reg:
        sample = stream.read(FRAMESIZE)
        sample = np.mean(np.fromstring(sample, 'Float32'))
        print(sample)
        if  sample > 0.005 or sample < -0.005:
            reg = False
    data=stream.read(NOFFRAMES*FRAMESIZE)
    decoded=np.fromstring(data,'Float32');
    stream.stop_stream()
    stream.close()
    p.terminate()
    print('done')
    plt.plot(decoded)
    plt.show()
    np.save(name,decoded)
    return decoded

def amplitude(file):
    file = np.load(file)
    spec = np.abs(np.fft.fft(file))
    fig, ax = plt.subplots(figsize=(800/100, 600/100), dpi=100)
    ax.plot(spec[:int(len(spec)/2)])
    ax.set_xlabel('frequency')
    ax.set_ylabel('amplitude')
    ax.set_title('Amplitudenspektrum')
    ax.autoscale(enable=True, axis='x', tight=True)
    #plt.show()

def windowing(file):
    counter = 0
    res = np.zeros(WINDOW)
    gauswin = scipy.signal.gaussian(WINDOW, WINDOW/4)
    cnt = 0
    while counter <= len(file):
        tmp = file[counter:counter+WINDOW]
```

```
            if len(tmp) is not WINDOW:
                vals = tmp
                tmp = np.zeros(WINDOW)
                for i in range(0,len(vals)):
                    tmp[i] = vals[i]
            tmp = np.multiply(tmp, gauswin)
            tmp = np.abs(np.fft.fft(tmp))
            res += tmp
            counter = counter + WINDOW/2
            cnt += 1
        res = np.divide(res, cnt)
        return res

def middle(name):
    value = np.zeros(len(np.load(name+str(1)+".npy")))
    counter = 0
    for i in range(0, 4):
        value = value + np.load(name+str(i+1)+".npy")
        counter = counter + 1
    value = np.divide(value, counter)
    plt.plot(value)
    np.save(name+"␣middle",value)

def korrelation(name):
    namearr = windowing(np.load(name+".npy"))
    hoch = windowing(np.load("hoch␣middle.npy"))
    tief = windowing(np.load("tief␣middle.npy"))
    links = windowing(np.load("links␣middle.npy"))
    rechts = windowing(np.load("rechts␣middle.npy"))
    corrList = []
    corrhoch = scipy.stats.pearsonr(namearr, hoch)[0]
    corrList.append(corrhoch)
    corrtief = scipy.stats.pearsonr(namearr, tief)[0]
    corrList.append(corrtief)
    corrlinks = scipy.stats.pearsonr(namearr, links)[0]
```

```
    corrList.append(corrlinks)
    corrrechts = scipy.stats.pearsonr(namearr, rechts)[0]
    corrList.append(corrrechts)
    correlation = max(corrList)
    if correlation == corrtief:
        return "tief"
    elif correlation == corrhoch:
        return "hoch"
    elif correlation == corrlinks:
        return "links"
    elif correlation == corrrechts:
        return "rechts"

#spec = windowing(decoded)
#amplitude(spec)

window = windowing(np.load('links␣middle.npy'))
fig, ax = plt.subplots(figsize=(800/100, 600/100), dpi=100)
ax.plot(window[:int(len(window)/2)])
ax.set_xlabel('frequency')
ax.set_ylabel('amplitude')
ax.set_title('Links')
ax.autoscale(enable=True, axis='x', tight=True)
```

A.2 Messergebnisse



Erkennung:

David:		Bene	
links:	1/5	links:	5/5
rechts:	0/5	rechts:	0/5
hoch:	5/5	hoch:	1/5
tief:	4/5	tief:	5/5
	10/20		11/20
	50%		55%

[**?**, S.21]