영어 음성학

2018131028 독어독문학과 이채연

*Phonetics는 물리적으로 어떻게 소리가 나는지와 관련된 것이다. 소리가 어떻게 인지되는지에 관심을 두는 phonology와는 차이가 있다. 영어에는 44개의 다른 소리가 존재한다.
Articulatory phonetics(mouth), Acoustic Phonetics(through air), Auditory Phonetics(to ear)
*Articulation(조음)
vocal tract(nose, pharynx, larynx, ear로 구성)
upper vocal tract[lip, teeth, alveolar ridge, hard palate, softpalate(velum), uvula, pharynx wall로 구성]
lower vocal tract[lip, tip, blade, front/center/back tongue, root, epiglottis로 구성]
한국어는 음절이 반복되고, 영어에서는 stress가 반복되기 때문에 혀의 위치가 중요
폐 -> 성대(수문과 비슷한 역할); 떨림이 느껴진다는 것은 성대가 펄럭인다는 것
여자 250번 정도, 남자는 100번 정도

1) articulatory process [lips, tongue body, tongue tip]
   constriction location : where exactly -(앞뒤)
   constriction degree how much exactlly(상하)
2) phonation process[larynx]
   voiced(진동을 느낄 수 있음) - 모든 모음과 일부 유성 자음
   voiceless(진동x) - p, h, k, s, f
3) oro-nasal process[velum]
   velum이 내려가면서 소리가 지나가는 통로가 열림
   모든 모음과 비음(m,n,ng)을 제외한 자음도 velum이 열리면서 소리남

*constriction location : where exactly -(앞뒤)
lips(2개의 소리: 앞 - [ㅂ][ㅍ] /뒤 - [ㄴ][ㅁ])
tongue body(2개의 소리 - 앞 : [ㅕ],[y] [ear], [yarn] / 뒤 : 그)
tongue tip(4개의 소리 - [th(윗니)],,)
constrictor 관점에서는 같지만, 로케이션 관점에서는 다른 소리
*constriction degree how much exactly(상하): 조금 막을 건지, 많이 막을 건지, upper part를 치는 정도에 따라
stops: [p], [b], [t], [d], [k], [g]
fricatives 마찰음 [f], [v], [θ], [ð], [s], [z], [ʃ], [ʒ], [h]
approximants [r], [l], [w], [j]
vowels 모음의 정의는 막힘이 없는 것 -> upper part를 별로 안 침
tongue body, tongue tip, lip, velum, larynx의 위치로 어떤 소리가 나는지 알 수 있다.
*praat: 소리의 duration, intensity(loudness), pitch를 확인 가능
빨간 띠: formant(F1, F2에 따라 모음 구별 가능)
praat에 나타나는 큰 파동의 횟수는 larynx의 떨림 횟수와 일치
헤르츠: 1초에 성대가 떨리는 횟수

2주차

Pratt

intensity(db), pitch(hz), Formants(hz) 분석 가능

HZ(= frequency의 단위) : 1초 당 반복되는 구간이 몇 번 생겼는지

여기서, 반복되는 구간은 결국 sine wave이며, Larynx의 떨림과 일치한다.

그리고 이 세상의 모든 signal은 sine wave로 이루어졌다.

사람의 목소리도 마찬가지다. 목에다 장치를 설치해, 그 소리를 녹음하면, sine wave가 나타나지만, 마이크를 통해 사람의 말소리를 녹음하면 sine wave의 합(=complex sound)이 나옴

즉 입모양에 따라 합쳐지는 sine wave의 종류가 달라진다.

sine wave는 frequency와 magnitude 두 가지 특성에 의해 구분된다.

합쳐진 sine wave는 complex tone이라 하는데, F0과 반복되는 주기가 일치한다. 그리고 각 sinewave들의 frequency는 f0의 배수이다.

voice source(목에서 직접 측정한 소리): 스펙트럼이 점진적으로 감소하는 모양

filtered by voice tract(마이크로 측정한 소리): 점직적으로 감소하는 모양이 유지되지 않음

하지만, 둘 다 f0 배수의 frequency를 가진다는 공통점이 있다.

소리를 표현하는 방법

1)스팩트로그램

x축: 시간, y축: frequency

*위쪽으로 갈수록 옅어짐

*frequency와 amplitude는 반비례

2)스팩트럼

x축: frequency, y축: amplitude

*주파수 별로 어떤 성분이 있는지 확인 가능하다.

목소리가 정해지는 방법 = f0과 이에 대응되는 합

f0은 pitch와 일치

filter: 스팩트럼에 나타나는 mountain과 valley의 모양을 결정지음

입모양에 따라 비슷한 모양을 띔

f1은 formant와 일치

f1은 혀의 높낮이와 일치

f2는 혀의 front / back과 일치

따라서 일정한 주파수와 amplitude 수치에 따라 특정 모음과 자음의 소리를 구성할 수 있고 pure 톤들의 합으로 사람 목소리를 나타내는 것이 가능하다.

3주차

코딩이란?

자동화 - 계속 같은 것이 반복되기 때문에

언어 - 단어를 어떻게 결합하는지(단어와 문법)

단어 - 의미/정보를 담는 그릇

어떤 단어를 이용해서 어떻게 합칠 것인가

컴퓨터 언어에서 단어의 역할을 하는 것은 변수(variable), 변수에 담기는 것은 문자와 숫자가 전부

1. Variable assignment 변수에다 정보를 지정하는 것
2. if Conditioning 자동화, 기계화 - 조건문 문법 if
3. 자동화에 중요한 것은 여러 번 반복하는 것 for
4. 함수: 어떤 것을 입력했을 때 원하는 값이 나오는 것이 함수

   1,2,3,4가 함수 안에 들어가는 것

   Ex> 모든 사람이 함수다 : 먹고 싸는 거, 기름으로 자동차가 달리는 것

재사용과 반복을 위해 코딩을 한다.

= : 같다는 뜻이 아니라, 오른쪽에 있는 정보를 왼쪽에 있는 변수로 assign한다.

함수(입력)

각각의 셀이 따로 실행 가능

Variable은 정보를 override 함

한 변수에 여러 개를 정보를 넣고 싶을 때 - list[], tuple(), dictionary={'key':'value'} 표제어와 설명의 쌍

차이 - 튜플이 더 보안에 강하다(바꾸기 힘들다)

Type() : 변수가 어떤 특성을 가지고 있는지 알려줌

4주차
하나의 variable에 여러 정보를 담으면, 최신 정보로 업데이트 됨
어떤 variable 속에 정보가 들어 있을 때,
retrieve 정보를 가지고 온다.

변수 간의 덧셈, data acess
a=[0번째, 1번째,,,]

리스트에서 일부분의 정보를 가져올 때, 대괄호를 이용함

타입종류()어떤 변수의 type을 다른 타입으로 바꿔주는 함수
어떤 변수의 내부 정보를 가져오고 싶으면, 대괄호를 사용하여, 인덱스를 적으면 됨

dict의 acess
pair set에서의 key 부분을 인덱스로 사용
string과 list는 정보의 접근 차원에서 비슷

```
In [47]: s='abcdef'

In [48]: print(s[0],s[5],s[-1],s[-6])
         a f f a
```

왼쪽으로 가면 음수
print(s[1:3]) - 1에서부터 3까지 가져와라
print(s[1:]) - 첫 번재부터 끝까지
print(s[:3]) - 처음부터 2번째까지

리스트도 같은 방식
len() 함수 - 정보의 길이를 알려줌
s.upper() - 대문자로 변화

for i in a:
in 뒤에 있는 것을 하나한씩 돌려서 I가 그 하나하나를 받고 어떤 일을 해라
if conditionint
function 입력이 들어가고 출력이 나오는 것들(이미 있는 것도 있고, 내가 만드는 것도 있음)
ipynb 확장자; 코드와 필기를 한번에

In [18]:
```
a=['red', 'green', 'blue', 'purple']
b=[0.2, 0.3, 0.1, 0.4]

# enumerate도 함수명 - 번호 매기는 함수, 두개의 변수가 들어갈 수  있음
for i,s in enumerate(a):
    #i에다가는 번호(index 값), s에다가는 red 이런 애들
    print("{}:{}%".format(s, b[i]*100))
```

```
red:20.0%
green:30.0%
blue:10.0%
purple:40.0%
```

In [19]:
```
a=['red', 'green', 'blue', 'purple']
b=[0.2, 0.3, 0.1, 0.4]

# enumerate도 함수명 - 번호 매기는 함수, 두개의 변수가 들어갈 수  있음
for i,s in enumerate(b):
    #i에다가는 번호(index 값), s에다가는 red 이런 애들
    print("{}:{}%".format(a[i], s*100))
```

```
red:20.0%
green:30.0%
blue:10.0%
purple:40.0%
```

In [21]:
```
a=['red', 'green', 'blue', 'purple']
b=[0.2, 0.3, 0.1, 0.4]

# enumerate도 함수명 - 번호 매기는 함수, 두개의 변수가 들어갈 수  있음
for s,i in zip(a, b):
    #a와 b는 반드시 길이가 같은 리스트가 되어야 함, zip - 둘을 합친다는 뜻, 딕셔너리처럼 바뀌는 것
    #s에는 a의 값, i에는 b의 값이 들어가는 것, index 값은 들어가지 않음
    print("{}:{}%".format(s, i*100))
```

```
red:20.0%
green:30.0%
blue:10.0%
purple:40.0%
```

In [28]:
```
a = 3
#(크거나 같으면)
if a >= 0:
    print('yy')
else :
    print('nn')
```

```
yy
```

In [31]:
```
for i in range(1,3):
    for j in range(3,5):
        if j >=4:
        print(i+j)
        #!!!인덴테이션!!!
    #계층적인 개념을 잘 알기!!
```

```
  File "<tokenize>", line 4
    print(i+j)
    ^
```

11/3 - 5주차
컴퓨터가 처리하는 모든 정보들은 데이터화되어야지만, 가치 있는 정보들이 된다.
영상, 이미지 - 각 픽셀에 할당된 값으로 표현
소리 - 수치로 표현 가능
벡터, 행렬로 표현해야지 수학적인 처리가 가능하다.
그것을 가능하게 해 주는 것이 numpy이다.
import numpy as np : numpy라는 라이브러리를 np라는 이름으로 가져와라
1. array() : 계산이 가능한 벡터 형태로 바꿔주는 함수(적은 메모리로 데이터를 빠르게 처리)
<리스트와 다른 점>
모든 원소가 같은 자료형이다 / 원소의 개수를 바꿀 수 없다.
리스트의 원소들은 계산이 불가능하지만, 배열의 원소는 가능하다.

2. 2차원 배열인 행렬을 만들 수 있다.
X = np.array([[1,2,3], [4,5,6]])
print(X) : array([1,2,3], [4,5,6]]) / X.shape : (2,3)

3. numpy 속 subpackage를 불러오고 싶을 때
imprt numpy.A.D.F = from numpy import A, D
import matplotlib.pyplot as plt = from matplotlib import pyplot as plt

4. 다차원 배열 생성 :np,empty/ones/zeros(만들고자 하는 배열을 튜플로 표현, 변수 종류)

5. np. arrange : 특정한 규칙에 따라 증가하는 수열을 만듦
np.arrange(시작, 끝, 증가하는 수) 시작과 끝을 포함하지 않음

6. np. linspace(시작, 끝, 원소의 개수, dtype = ~)
시작 ~ 끝까지의 수들 중 n개만큼의 원소의 벡터를 만드는 데, 숫자들 사이의 거리가 같다.

7. 하나의 행렬에 대해서 속성을 알아낼 수 있는 다양한 함수가 존재한다.
X = np.array([4,5,6], [1,2,3])
X.ndim #배열의 차원 / X.shape #행렬의 형태 / X.dtype #행렬 속 데이터 정보를 알려줌
X.astype(np.float64) #행렬 속 데이터 정보를 바꿔줌 /np.zeros_like(X) = X*0

8. 정규 분포
data = np.random.normal(표준편차, 평균, 데이터 수) #정규분포를 만들어 주는 함수
plt.hist(data, bins = 10)
#위의 데이터로 10개 범위의 히스토그램을 만들어라 = 100개 데이터를 10개의 범위로 표현
plt.show() #그 히스토그램을 보여줘

9. 3차원의 행렬
X = np.ones([차원의 수, 가로줄의 수(행의 수), 세로줄의 수(열의 수)])

#차원의 개념 :줄은 일차원 / 직사각형은 2차원 / 3차원은 그 직사각형이 두개인 것
Y = X.reshape(-1, 바꾸고 싶은 만큼의 가로의 수, 바꾸고 싶은 세로의 수)
# Y의 원소의 수는 X의 원소의 수와 반드시 일치해야 한다.
즉 차원 * 행 * 열의 값이 같아야 한다. 이 계산에 맞게 알아서 계산해주는 것이 바로 -1

10. np.allclose(X.reshape(-1, 3, 2), Y)
두 개의 배열이 서로 호환 가능하면, True를 출력

11. np.random.randint(시작, 끝, [행, 열])
시작을 포함하여 끝-1까지의 수 중, 균일 분포의 정수 난수(random number)
np.random.random(받고 싶은 자료의 크기)
ex> np.random.random(5) ->1*5의 랜덤 함수/ np.random.random((2,2)) -> 2*2의 함수

12. np.save("test",a,b) -> 실제 파일에 다음의 변수를 저장 / who : 지금 avaialble한 변수
np.load() : .npy 파일을 배열로 불러오기! / npzfiles.files : 현재 available한 파일
/npzfiles['이름'] -> 해당 배열

16. arr = np.random.random([5,2,3]) -> 5차원의 2개의 행 3개의 열
type(arr) : numpy.ndarray / len(arr): 5 / arr.shape: (5,2,3) / arr.ndim: 3
arr.size: 30 #원소의 개수 arr.dtype : floast64 #원소의 데이터 타입