영어음성학 summary

2018130819 영어영문학과
정해민

Phonetics

- Articulatory phonetics (from mouth) 조음
- Acoustic phonetics (through air) 음향
- Auditory phonetics (to ear) 청각

Articulation

5 speech organs

1. Articulatory process

   Lips / tongue tip / tongue body: 가장 중요한 부분

2. Oro-nasal process

   Velum: nasals (m, n, ng, …)

3. Phonation process

   Larynx: voiced/voiceless (larynx=voicebox)

Control of constrictors

- 3 major constrictors: lips, tongue tip, tongue body
- Constriction Location (CL)

  Lips: bilabial / labiodental

  Tongue body: palatal / velar

  Tongue tip: dental / alveolar / retroflex / palate-alveolar
- Constriction Degree (CD)

  (upper) stops / fricatives / approximants / vowels (lower)

➔ By specifying constrictors, CD, and CL, we produce English consonants & vowels.

코딩은 '자동화'

자동화를 하는 이유: 초기 에너지를 투자하면 반복하는 데 필요한 불필요한 에너지를 절약할 수 있다.

모든 언어는 '단어'가 있고, 이를 '결합'하는 방법이 있다.

단어는 '정보'를 담는 그릇.

컴퓨터 언어에서 '단어'에 해당하는 부분이 '변수'

기계와의 '문법'은 생각보다 어렵지 않다.

1. 변수라는 그릇에 정보를 넣는 것 variable assigning
2. 자동화 If conditioning
3. 여러 번 반복 for route
4. 함수: 어떤 입력을 넣으면 출력이 나오도록 하는 것

A=1

등호 오른쪽이 정보(1) 왼쪽이 변수(a)

A: variable / 1: information

Run을 해야 입력이 됨!

python에서 모든 함수는 누가 만들어 놔야 함(직접 만들 수도 있음)

정보의 종류: 숫자, 문자

셀 지우기: x

셀 만들기: b (위로 만들기 a)

실행(run) 단축키: shift+enter

a=[1,2] ; b=[3,4] ;c=a[0]+b[0]

c

4

A[0]+b[0]에서 대괄호 안에 숫자는 '순서'를 의미 (컴퓨터에서는 1이 아니라 0부터 시작)

어떤 variable의 정보로 들어갈 때 대괄호 안에 index를 적으면 내부 정보를 데려온다.
a='123';print(type(a));print(a[1])

Dictionary: pair에서 앞 부분을 index로 쓴다. (그냥 list에서는 0,1,2,3 등 숫자를 index로 썼음) a={"a":"apple","b":"orange","c":2014} ; print(type(a)) ; print(a["a"])

String과 list는 정보의 측면에서 거의 비슷

S='abcdef' N=[100,200,300]

<len 함수>

Len(s) 6

Len(n) 3

*** this는 문장에 두개 들어 있는데 첫번째 것만 찾아주는 함수 (find)

*** rindex는 오른쪽에서 카운트하는 것

<split 함수> 아주 유용: 잡 스러운 것 지워줌

s=s.strip()

s -> 'this is a house built this year.'

<문장을 list로 만들 때: split 함수>

s.split(' ') -> space를 기준으로 잘라라 (작은 따옴표 안에 들어간 거 기준으로 잘라라)

numpy라는 package library 속에는 또 작은 package들이 들어 있을 수 있고, 그 안에 또, 그 안에 또 ...

numpy.A.B.@ 의 경우 numpy안의 A안의 B안의 @라는 함수를 뜻함 -> 이게 너무 복잡해서 쓰는 방법들이 몇 가지 있음.

from numpy import A : numpy에 있는 A를 불러오자. -> A.D.@ 로 접근할 수 있음

from numpy import A.D : (한 마디로 import를 크게도 할 수 있고 from을 사용해서 작게도 할 수 있음)

import랑 from을 잘 쓰면 됨

#numpy가 필요한 이유? numpy는 list와 아주 비슷한데 수학적인 계산 가능 앞으로 쓰게 될 모든 데이터 처리는 list가 아니라 numpy 처리

#empty는 함수 (함수는 괄호로 그 뒤에 input을 받는다고 했으니까 empty는 함수)

#numpy라는 제일 큰 library 속에 최상단에 있는 empty라는 함수

#2x3로 (옆으로 뚱뚱한 모양) 내부적으로 들어간 data type을 int(eger)로 하나 만들어라 (숫자는 random)

#float도 종류가 있음 float64의 경우 소수점 64번째자리까지 -> 아주 정밀하게, 오차를 허용하고 싶지 않을 때 사용

#단점은 많은 용량 차지

data=np.random.normal(0,1,100)

print(data)

%plt.hist(data,bins=10)

%plt.show()

#normal이라는 함수는 normal distribution을 만들어주는 함수 normal distribution은 '정규분포'를 뜻함

#-> shape을 만들어주는 게 아니라 data를 만들어 주는 것

#plt는 matplotlib라는 library 속의 pyplot을 부름.

#plt속의 hist라는 함수를 쓸 것. hist는 '히스토그램' 히스토그램의 option에는 늘 bins라고 해서 바구니를 총 몇개를 할 건가를 정함

#x축에 바구니를 10개 만드는 것 (바구니 5개 다음 0(mean)나오고 5개 나오고)

#range속에 들어가는 위의 값들이 y값. y값에 들어가는 것들은 절대 소수값이 나올 수 없음. 값들을 다 더하면 100개

Phasor

theta값만 있으면 sine,cosine 그래프는 만들어지지만 실제 소리는 만들 수 없다.

소리라고 명명하는 순간 -> 시간의 개념이 반드시 들어가야 함

sine, cosine같은 것들을 phasor라고 하는데

phase, theta, radian?, + time을 반드시 연동시켜야 play할 수 있다.

#parameter setting

amp=10 #range [0.0,1.0]

sr=10000 #samplint rate, Hz (얼마나 정보를 촘촘하게 할건가, 숫자(점)들이 1초동안 몇번 나오는가)

dur=0.5 #in seconds (얼마나 길게 할건가)

freq=440.0 #sine frequency, Hz (그 사인웨이브가 1초동안 몇번 왔다갔다(반복)할건가)

#sr과 freq는 unit은 같이 쓰지만 내용이 완전 다르니까 조심

sampling rate이 100Hz라고 생각해보자

우리가 표현할 수 있는 숫자의 개수가 1초에 100개라는 뜻

이 100개의 숫자를 가지고 1Hz frequency를 표현할 수 있다? 없다? -> 있다 (그냥 한번의 사인 웨이브 주기를 표현하면 됨)

2Hz 가능? -> 가능

10000Hz 가능? -> 불가능

sampling rate이 충분해야 그만큼의 주파수를 표현할 수 있다

너무너무 중요한 개념이니까 꼭 이해해야 한다

★Nyquist Frequency★

표현할 수 있는 주파수 max: 주어진 sapling rate 숫자의 반

ex. sampling rate 10hz 이면, 표현할 수 있는 frequency는 5가 maximum

CD음질이라고 할 때, CD가 갖는 것은 44100Hz(Sampling rate)

이것의 nyquist frequency는 22050hz (아주 높은 소리까지 표현 가능)

근데 왜 CD음질을 44100으로 잡았을까? 사람이 들을 수 있는 가청주파수가 20000hz

<Linear algebra>

- Matrices 행렬

  행렬의 크기를 차원이라고 하는데 차원을 이야기 할 때는 M by N 행렬 (M:행, N:열)

- Vectors

  벡터는 행렬 중에서도 Nx1이나 1XN 행렬을 말하는데, sequence of numbers임

  세로로 긴 행렬을 column vector라고 하고, 가로로 긴 행렬을 row vector라고 함

- Vector multiplication / addition

  Ex. (3,4) + (2,1) = (5,5) 기하 상에서 평행사변형을 그려서 만나는 점이 (5,5) – 3차원에서도 동일

- Vector spaces

  *linear combinations (중요)

  cv+dw에서 v와 w는 차원이 같은 두 벡터고 c,d는 벡터에 곱해지는 단순한 숫자(scalars)

  linear combination을 했을 경우, 벡터들이 어디로 튈지 모르기 때문에 x차원 vector space는 x차원의 모든 공간이 되어 야지, 일부분은 vector space라고 볼 수 없다.

- Column space

  Independent한 column vector들을 linear combination할 수 있는 모든 vector들이 이루는 space

  Column space와 row space의 차원은 항상 같아야 한다. Ex. 6x7행렬에서 column space가 3차원일 때, row space도 3차원이어야 함

  Column space의 차원은 whole space의 차원을 넘을 수 없다.

  만약 vector들이 서로 dependent할 경우 (2차원 공간에서) column space는 1차원이 되고 나머지 1차원은 null space라고 부름

  요약하자면, whole space – row의 개수 / column space – independent한 column vector의 개수

- Four spaces in a matrix

  Column vector의 관점처럼 row vector의 관점에서도 space를 이야기할 수 있음

  Four space는 column vector 관점의 whole/column space와 row vector관점의 whole/row space로 구성

  Column space에서 나오는 null space를 left null space라고 하고 row space에서 나오는 null space를 (right) null space라고 함 (null space x가 A의 왼쪽에 붙느냐 오른쪽에 붙느냐는 left/right null space에 따라 달라짐

- Linear transformation

  Ax=b에서 x: 입력 벡터, b: 출력 벡터

  입력 벡터와 출력 벡터의 차원은 반드시 같을 필요가 없는 게, A에 따라서 b의 차원이 달라지기 때문 – x를 b로 바꿔주는 transformation matrix는 A

  Basis vector: (1,0), (0,1)

- Detransformation: Inverse matrix (역함수)

  Dependent한 vector들은 invertible할 수 없다

- Eigenvector

  Av=b에서 v의 eigenvector는 transformation 후 결과 값 b와 원점이 v와 동일 선 상에 모두 있을 때

- Null space

  Null space의 정의에서 어떤 방향으로의 변화는 출력에 영향을 미치고, 어떤 방향으로의 변화는 출력에 영향을 미치지 않는다. -> 이를 확실히 구분하는 것이 null space에 대한 이해에 도움이 된다.

  Ex. Ax=b를 인공지능 사진인식이라고 생각해보자. 강아지 이미지가 x로 들어갔을 때, 수많은 강아지 이미지들이 다 다름에도 출력값 b는 '강아지'라는 개념으로 동일하다. 이 경우 많은 종류의 강아지 '이미지'들은 출력값에 영향을 미치지 않는 null space를 따라서 이동하는 것이라고 할 수 있다.

  본래 null space의 목적은 더 artistic하게 보이기 위해서가 아니라, 어떤 task를 이루는 데 방해 받을 만한 요인을 피해 가기 위함이다.

  또 하나는, 기계에서 무언가를 인식할 때, 입력(아까 예에서는 그림)이 많이 변해도 같은 해석이 나오는 경우가 null space를 사용한 것이다.

- Eigenvector

  어떤 행렬의 eigenvector는 하나의 값이 아니라 eigenvector space로 봐야 한다. (값이 아니라 '방향'으로 이해할 것)

  Eigenvalue: 원점에서 입력값까지와 출력값까지의 비율

- Correlation 상관 관계

  상관 관계는 r이라는 수치로 표현할 수 있고 -1에서 1까지의 범주

  -1, +1이 정확하게 나오는 순간은 모든 벡터들이 완벽하게 한 선 상에 있을 때 (기울기는 상관 없음) -> 얼마나 그 '선'에 가깝냐 가 r의 절대값이라고 보면 된다. 0일 때는 동그라미 모양

  Ex. 85명의 학생들의 국어, 영어, 수학 성적으로 85차원의 국어, 영어, 수학 벡터를 만듦

  원래는 축이 85개지만 다른 축을 다 지우면 아무리 차원이 높아도 삼각형을 이룰 수가 있음

  각각의 축으로 cos값을 정의하면, 두개가 거의 붙어 있을 때는 cos0=1 / 떨어져 있을 때는 cos90=0 (1,0은 r값)

  그럼 대체 85차원 안에서 각도 값을 어떻게 구하지? -> inner product

- Inner product (dot product)

  두 벡터(a,b)가 원점과 이루는 삼각형에서 a를 b에 수직으로 내린 길이(a길이 * cos세타) * b의 길이

  a길이는 루트 안에 a를 이루는 모든 값의 제곱을 더하면 됨

  그렇다면 inner product가 왜 필요하느냐? 어떤 sound signal이 있을 때, 이것이 어떤 frequency의 wave들로 이루어져 있는지 알려주는 것이 spectrogram인데, signal vector에 여러 개의 sine wave를 만든 후 inner product하면 correlation에 따라 inner product 값이 다르게 나오기 때문이다.

- Cosine Similarity

  두 벡터가 얼마나 비슷한 지를 수치적으로 알려주는 방법

  똑 같은 성분이 많이 있으면 inner product 값이 크게 나오고, 똑 같은 성분이 많이 없으면 같은 성분에만 response를 주기 때문에 값이 적게 나옴

  *맹점: 한 sine wave와 같은 frequency의 cosine wave(옆으로 90도 이동)을 만들면 inner product 값이 00이 나옴

  기하상에서도 두 벡터가 이루는 각도가 90도일 때, inner product가 0인데 그래프 상의 90도와 기하상의 90도가 공교롭게도 일치한다는 점이 매우 신기함

  ➔ 이처럼 sine, cosine은 phasor shift에 대한 민감도가 안 좋기 때문에 complex phasor를 사용