Prepoznavanje govora

End pointing

End pointing

- Problem pronalaženja početaka i krajeva reči u govornom signalu.

Pristupi

- Zasnovani na energiji
- Zasnovani na karakteristikama zvuka

- Mi ćemo se za sada baviti samo zasnovanim na energiji.

Napomene

- Uvek ćemo koristiti apsolutne vrednosti signala (ili kvadratne)
- Treba uzeti u obzir činjenicu da imamo dva kanala. Ovde ima više strategija, koje se manje-više slično ponašaju:
  - Tretirati dva kanala kao dva sempla (imamo duplo više semplova)
  - Uzeti srednju vrednost dva kanala kao trenutni sempl
  - Uzeti maksimum dva kanala kao trenutni sempl

Naivan algoritam (A)

- Empirijski odrediti nivo šuma na snimku
  - Ovo može da ima smisla ako je okruženje izuzetno stabilno, i znamo da će snimci uvek doći iz takvog okruženja
- Dok ima semplova
  - Pročitati sempl
  - Ako je energija sempla veća od nivoa energije šuma
    - Ulazimo u stanje govora
  - Ako je energija sempla manja od nivoa energije šuma
    - Izlazimo iz stanja govora

Problem

- Priroda zvučnog signala je takva da će (čak i pri konstantnom vikanju) definitivno u toku izgovaranja reči u nekom trenutku nivo signala pasti ispod nivoa šuma.

  →

- Treba posmatrati prozor i usrednjavati.

Malo manje naivan algoritam (B)

- Empirijski odrediti nivo šuma na snimku
  - Ovo može da ima smisla ako je okruženje izuzetno stabilno, i znamo da će snimci uvek doći iz tog okruženja
- Dok ima prozora (širine 10ms)
  - Pročitati prozor
  - Odrediti srednju vrednost energije na nivou prozora
  - Ako je srednja energija prozora veća od nivoa energije šuma
    - Ulazimo u stanje govora
  - Ako je srednja energija prozora manja od nivoa energije šuma
    - Izlazimo iz stanja govora

Šumšum

- Ako nam okruženje nema konstantan nivo šuma, ne možemo držati tu granicu kao konstantu.
- Za početak pretpostavimo da je nivo šuma konstantan na nivou jednog snimka.
- Možemo programski da odradimo merenje šuma umesto ručno. Na prvih 100ms snimka (koje zahtevamo da nemaju govor) izmerimo srednji nivo signala, i to postavljamo kao granicu šuma.

Šumšum (poboljšanje)

- Prosto posmatranje srednje vrednosti često nije dovoljno restriktivno.
- Statistički je opravdano da kao granicu uzmemo srednju vrednost udaljenu za dve standardne devijacije.
- Granica – L

  $L = \mu + 2 \cdot \sigma$

- (definicije za μ i σ date na narednom slajdu)

Šumšum (poboljšanje)

- Broj semplova u prozoru – N
- Vrednosti semplova: s_1, s_2, … s_N
- Srednja vrednost – μ
- Standardna devijacija – σ

$$\mu = \frac{s_1 + s_2 + \ldots + s_N}{N}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i - \mu)^2}$$

Merenje šuma (C)

- Isčitati prvih 100ms snimka
  - Postaviti nivo šuma na srednju vrednost signala u ovom delu snimka
- Dok ima prozora (širine 10ms)
  - Pročitati prozor
  - Odrediti granicu energije na nivou prozora
  - Ako je srednja energija prozora veća od granice šuma
    - Ulazimo u stanje govora
  - Ako je srednja energija prozora manja od granice šuma
    - Izlazimo iz stanja govora

Spike / ponor

- Postoji mogućnost da se u signal uvuče zvuk koji nije govor, ali liči na njega. U zvučnom signalu će to biti spajkovi visoko iznad šuma, koji mogu da „prevare“ algoritam. (spike)
- Isto tako, može da se desi da u periodu koji je zapravo govor bude kratko zatišje, koje želimo da ostavimo markirano kao govor. (ponor)
- Obe ove anomalije želimo da izravnamo.

Spuštanje spike-ova

- Prozor obeležen kao šum ćemo predstavljati sa 0, a prozor obeležen kao signal sa 1.
- Jednostavna taktika je da zahtevamo izvestan broj 1 pre nego što ih stvarno prihvatimo. Ako ih nema dovoljno, retroaktivno ih spustimo.
- Primer (zahtevamo tri uzastopne 1):
  - Ulaz:
    - `00110000011111111000100000`
  - Izlaz:
    - `00000000011111111000000000`

Podizanje ponora

- Sličnu taktiku možemo da primenimo da podignemo ponore.
- Zahtevamo da postoji neki fiksan uzastopni broj 0, u suprotnom ih podižemo:
- Primer (zahtevamo tri uzastopne 0):
  - Ulaz:
    - `00000011111011101111111100000`
  - Izlaz:
    - `00000011111111111111111100000`
- Primetite da je bitno kojim redosledom se rade ove dve operacije.

Strategija poravnanja

- Postoje dve strategije koje možemo da uzmemo ovde:
  - Konzervativna – čuvamo što više možemo od signala (prvo podižemo ponore, potom spuštamo spike-ove)
  - Reduktivna – sečemo što više možemo od signala (prvo spuštamo spike-ove, potom podižemo ponore)
- Za prepoznavanje govora, konzervativna strategija je često poželjna, jer nam je često bolje da imamo višak nego manjak informacija.

Strategija poravnanja

- Ulaz:
  - `00000110111111111101000000`
- Konzervativni rezultat:
  - `00000111111111111111000000`
- Reduktivni rezultat:
  - `00000000111111111100000000`

Prozori prozora (D)

- Isčitati prvih 100ms snimka
  - Postaviti nivo šuma na srednju vrednost signala u ovom delu snimka
- Dok ima prozora (širine 10ms)
  - Pročitati prozor
  - Odrediti granicu energije na nivou prozora i obeležiti trenutni prozor kao 1 (signal) ili 0 (šum).
  - Ako smo naišli na 1, i pre toga je bilo manje od X 0 → te 0 podižemo na 1.
- U drugom prolazu se radi dodatni smoothing:
  - Ako naiđemo na niz 1 dužine manje od Y → te 1 spuštamo na 0.
- X i Y su parametri sistema.

Spuštanje ponora (ponovo)

- Samo posmatranje dužine sekvence često nije dovoljno dobro. Recimo da imamo konzervativnu strategiju i sledeći primer:
- Ulaz:
  - `000101100000011111101111110000000`
- Izlaz:
  - `000111100000011111111111110000000`

Spuštanje ponora (ponovo)

- Da bismo ispravili prethodno opisani problem, u prvom prolazu nećemo da jednostavno očekujemo nekoliko uzastopnih istih vrednosti, već ćemo da zahtevamo da se tom izmenom dobije nova sekvenca izvesne dužine.
- Na primer, kod konzervativnog pristupa, ako naiđemo na sekvencu 0 kraću od 3, privremeno ih podižemo, potom proveravamo da li tako dobijena sekvenca 1 kraća od 6. Ako jeste, ignorišemo promenu.

Spuštanje ponora (ponovo)

- Ispravljeni algoritam:
- Ulaz:
  - `000101100000001111110111111000000`
- Privremeni rezultat:
  - `000111100000001111111111111000000`
- Konačni rezultat:
  - `000000000000001111111111111000000`

Prozori prozora od prozora (E)

- Isčitati prvih 100ms snimka
  - Postaviti nivo šuma na srednju vrednost signala u ovom delu snimka
- Dok ima prozora (širine 10ms)
  - Pročitati prozor
  - Odrediti granicu energije na nivou prozora i obeležiti trenutni prozor kao 1 (signal) ili 0 (šum).
  - Ako smo naišli na 1, i pre toga je bilo manje od X 0 → te 0 podižemo na 1, pod uslovom da tako dobijen niz 1 bude duži od Y.
- U drugom prolazu se radi dodatni smoothing:
  - Ako naiđemo na niz 1 dužine manje od Z → te 1 spuštamo na 0.
- X, Y i Z su parametri sistema.

Šta je š a šta je šum?

- Frikativi i nazali na početku ili kraju govora mogu lako da se pomešaju sa šumom ako gledamo samo nivoe energije.
- Postoji konzervativna mera koja u nekim slučajevima može da nam pomogne da pronađemo (posebno zvučne) glasove.

Zero-crossing rate

- ZCR se definiše kao broj prelaska nivoa energije iz pozitivnog u negativni opseg na fiksnom vremenskom periodu (npr. 10ms).
- Kod zvučnih glasova postoji tendencija da ZCR naglo poraste.
- Ako ZCR pređe unapred postavljen threshold (ZCT), možemo bezbedno da pretpostavimo da je u pitanju glas, a ne šum.

ZCT

- Moguća strategija za odabiranje ZCT – slično kao kod šuma
- Za prvih 100ms (tišine) merimo ZCT, i srednju vrednost postavljamo kao granicu (opet, dobro je da dodamo 2*σ).
- Za slučaj da snimak poseduje neprirodan šum, nije loše postaviti fiksnu vrednost preko koje ZCT ne može da ode
  - 25 prelazaka / 10ms.

Šumovi i š-ovi (F)

- Algoritam identičan kao E, nakon čega:
- Ispitamo ZCT za fiksne intervale (recimo 250ms) pre i posle govora.

Vežba

- Pretpostavićemo da već postoji program koji radi kao algoritam C, i proizvodi niz koji se sastoji od 0 i 1, pritom:
  - 0 predstavlja prozor koji je označen kao šum
  - 1 predstavlja prozor koji je označen kao signal
- Napisati program koji kao ulaz uzima ovaj niz, a kao izlaz daje rezultat rada algoritma D (konzervativni smoothing, prosto brojanje).
- Primeri ulaza dati na narednom slajdu.

Vežba

- Primer 1:
  - Ulaz:
    - `00000110111110100000`
  - Očekivani izlaz:
    - `00000111111111100000`
- Primer 2:
  - Ulaz:
    - `00010100011111011110100001000`
  - Očekivani izlaz:
    - `00011100011111111111100000000`