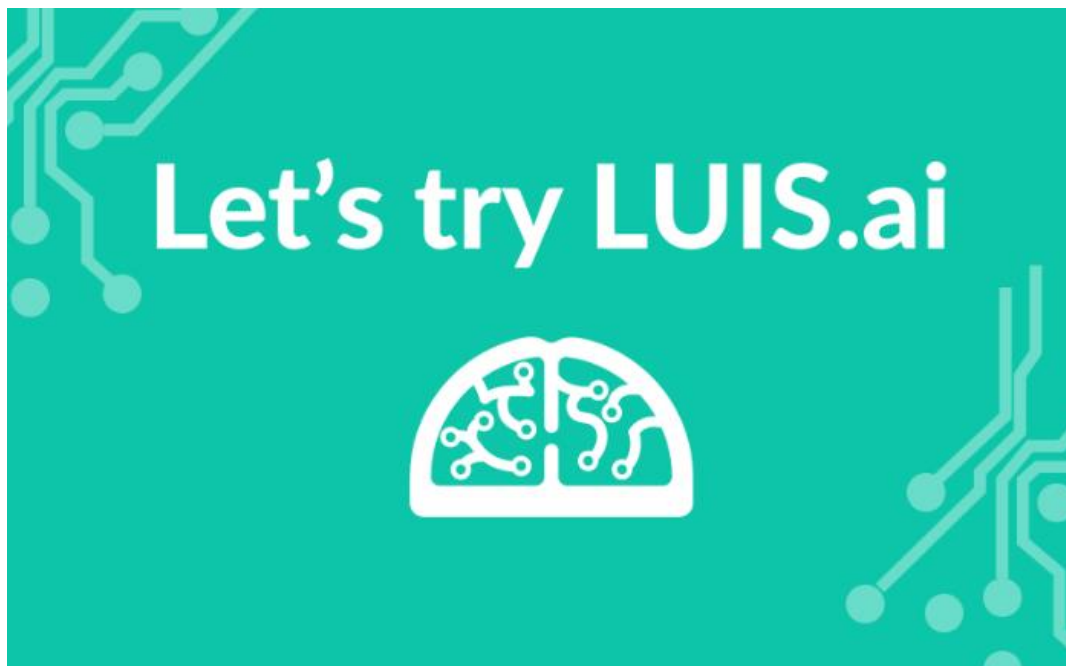




UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA



Proiect realizat de:

Péntek Tamás

Podaru Bogdan



Introducere

Procesarea limbajului natural este o metoda folosita in inteligenta artificiala pentru a comunica cu sisteme inteligente folosind limbaj natural. Intrarile unui astfel de sistem sunt textul scris sau voce. NLP se imparte in doua mari categorii: intelegerea limbajului natural (Natural Language Understanding – NLU) si generarea limbajului natural (Natural Language Generation – NLG). NLU se bazeaza pe maparea intrarii data in limbaj natural in structuri intelese de calculator si analiza lor. Pe de alta parte NLG produce fraze care au sens in limbaj natural din structuri date.

Ce este LUIS?

LUIS este un serviciu web bazat pe cloud, care foloseste intelegerea limbajului natural prin aplicarea invatarii automate pentru a extrage informatiile relevante dintr-o conversatie data in limbaj natural. Exemple de aplicatii care suporta serviciul LUIS sunt: Twitter, Cortana, Skype, Teams, Facebook Messenger si orice aplicatie care permite un mediu de conversatie.

Cum functioneaza LUIS?

Pentru a folosi LUIS este nevoie de un model, de antrenarea lui, testarea si publicarea lui. Odata publicat, un client poate trimite un text (utterance) in limbaj natural la endpoint, unde se gaseste serviciul. Acest text este prelucrat si rezultatul prelucrarii este dat in format JSON. Acest rezultat reprezinta predictia data de modelul antrenat.

Pentru a antrena modelul ce sta la baza lui LUIS e nevoie sa specificam entitatile si intenturile ce urmeaza a fi recunoscute dintr-un utterance. In primul rand trebuie sa specificam entitatile, prin crearea numelui si a tipului :

What type of entity do you want to create?

Entity name (Required)

Entity type (Required)

- Simple
- Hierarchical
- Composite
- List
- Regex
- Pattern.any

Done Cancel



Odata creata o entitate putem sa ii adaugam mai multe valori, astfel daca intr-un utterance se va regasi una din valorile declarate pentru o anumita entitate, LUIS va identifica valoarea ca fiind de tipul entitatii. Putem sa incepem prin a adauga cateva valori specifice domeniului entitatii, iar apoi LUIS va afisa niste recomandari de valori din acest domeniu, din care utilizatorul poate sa aleaga:

[Recommend](#)Add all

dual core +

i5 +

quad core +

celeron +

i3 2100 +

ultrabook +

i8 +


i4 +


core2duo +

330m +

Values

Add new sublist ...

 [Import values](#)

Search ... 

☐ Normalized Value

[Synonyms ?](#)

CPU


Click here to start adding values

Intel

Click here to start adding values

i3

Click here to start adding values

 [Get Started](#)










Cu cat vom adauga mai multe valori cu atat domeniul va deveni mai specific si recomandarile lui LUIS vor fi mai exacte.


Pana acum am identificat partile importante ale unui utterance si anume entitatile. Acum trebuie sa specificam **scopul** unui utterance, care se face prin crearea unui intent. Deoarece mai multe exprimari diferite pot avea acelasi inteles si pot duce la acelasi scop, putem specifica in interfata LUIS cateva exemple de exprimari ale unui intent, iar LUIS se va orienta spre expresii cu acelasi inteles. Exemplele noi introduse de utilizator care nu au fost folosite la antrenarea lui LUIS, dar care au acelasi inteles cu cele din "baza de intents" vor fi recunoscute cu o anumita precizie. Cu cat numarul de exemple/intent folosite la antrenare este mai mare, cu atat precizia predictiei e mai mare. In interfata LUIS, se selecteaza un intent creat si se specifica cateva exemple care tin de acest intent:



Remark

Labelled entities: None

 Edit	 Reassign intent 	 Add as pattern	 Delete	 Search	 Filter	 View options
<input type="checkbox"/> Example utterance	Score 					
<input type="text" value="A new example "/>						
the component is supplied	0.95					
the component powered	0.94					
the component is connected	0.95					
the component is ready to be used	0.96					
i ' m done with the component	0.96					
i ' m done with component	0.96					

 Get Started

Exista anumite situatii in care putem avea mai multe utterance-uri care difera doar prin tipul entitatii.

Spre exemplu consideram urmatoarele exemple:

"I have placed the CPU."


"I have placed the GPU."

Pentru a elimina redundanta de mai sus introducem un **phrase list** (in acest caz numit Component) in care grupam cele doua valori (CPU si GPU), iar fraza folosita pentru antrenare se va putea rescrie sub forma:

"I have placed the component." Astfel entitati precum CPU, GPU, component devin interschimbabile si numarul de exemple folosite la antrenare se reduce semnificativ.

Phrase lists

 Create new phrase list

<input type="checkbox"/> Name 	Value	Mode	Status
<input type="checkbox"/> component	cpu,ram,motherboard,battery,screen,keybo... ...	Interchangeable	Enabled

Anatomia unei predictii in format JSON

Utterance este textul trimis de client pentru a fi procesat, numit si **query**. LUIS cauta informatiile relevante in acest query, si le prezice cu un anumit **score** care reprezinta precizia predictiei. Un **intent** reprezinta un task pe care un utilizator vrea sa realizeze, se mai numeste si scopul unei utterance. O entitate (**entity**) reprezinta informatia de interes care este extrasa din utterance, cum ar fi nume, date, un grup reprezentativ de cuvinte. Aceste entitati au nume, tip si valoare. Numele este structura identificata in utterance, tipul este declarat la constructia modelului, iar valoarea



este cuvântul propriu-zis asociat structurii. În imaginea următoare scopul este indentificarea unor structuri în următorul query: „I have placed the CPU.” care se referă la o remarcă pe care un utilizator, de pildă un tehnician IT, o transmite unui asistent bazat pe LUIS pentru a fi analizată. Serviciul returnează un topIntent cu score-ul de 0.99 care reprezintă intentul cel mai potrivit în acest context. Totodată se observă că s-a identificat o entitate de tip „CPU”, cu valoarea „CPU”.

```
{
  "query": "I have placed the CPU.",
  "topScoringIntent": {
    "intent": "Remark",
    "score": 0.997268856
  },
  "intents": [
    {
      "intent": "Remark",
      "score": 0.997268856
    },
    {
      "intent": "Question",
      "score": 0.00508717541
    },
    {
      "intent": "None",
      "score": 0.0008576008
    },
    {
      "intent": "ListCommand",
      "score": 0.000779882132
    }
  ],
  "entities": [
    {
      "entity": "cpu",
      "type": "CPU",
      "startIndex": 18,
      "endIndex": 20,
      "resolution": {
        "values": [
          "CPU"
        ]
      }
    }
  ]
}
```



Tema proiectului

Aplicatia noastra reprezinta un asistent pentru asamblarea unui laptop. Asistentul comunica cu utilizatorul prin intermediul unui chat (text si voce) si il ghideaza in acest proces prin comenzi. Aplicatia suporta trei tipuri de comenzi pe care utilizatorul i le poate comunica asistentului: utilizatorul face o remarca despre ceea ce a realizat la un anumit pas, pune o intrebare atunci cand vrea sa stie ce trebuie sa faca la un anumit pas si o comanda pentru afisarea componentelor nefolosite. Aplicatia a fost realizata cu ajutorul Microsoft Bot Framework si se bazeaza pe doua servicii: LUIS si Cognitive Speech Service (Azure).

Reguli recunoscute

Aplicatia recunoaste trei tipuri de propozitii: Remark, Question si List Command, folosite si la antrenarea modelului din LUIS.

- a.) *Remark*: o afirmatie de forma: „I have placed the component.”, caz in care sistemul proceseaza componenta respectiva. Sistemul recunoaste fraze ale caror scop au montarea unei componente (cel de mai sus este doar un exemplu).
- b.) *Question*: recunoasterea unor forme de interogare, precum: „What is the next component?” sau altele cu acelasi sens, caz in care sistemul verifica unde am ajuns cu procesul de constructie a laptopului si anunta componenta cea mai potrivita pentru a fi montata in pasul respectiv.
- c.) *List Command*: recunoasterea de tipul : „List all” sau alte propozitii asemanatoare, iar atunci asistentul va lista toate componentele care sunt necesare pentru ca laptopul sa fie complet asamblat.

Entitatile extrase dintr-un utterance reprezinta componentele folosite in procesul de constructia unui laptop. Modelul a fost antrenat cu urmatoarele entitati: Chassis, Motherboard, CPU, GPU, Cooler, RAM, Battery, Hard disk, Wifi, Speakers, Screen, Keyboard, Webcam, Trackpad.

Procesul de constructie presupune ca piesele sa fie utilizate intr-o anumita ordine (de exemplu Cooler-ul va veni intotdeauna dupa CPU si nu invers). Totusi exista anumite componente a caror montare poate fi efectuate daca ele sunt localizate in zone separate din carcasa. De aceea am introdus ideea utilizarii unor clase de echivalenta care sa pastreze atat o relatie de prioritate (de exemplu, Cooler-ul are o prioritate mai mare decat CPU), cat si o relatie de egalitate (de exemplu, CPU-ul poate fi montat dupa GPU, dar si invers). Astfel am realizat urmatoarele clase de echivalenta cu prioritatea data ca si index, iar clasele de la acelasi index vor fi echivalente in sensul ca nu necesita ordine de montare. Componentele din clasa de echivalenta $i + 1$ au o prioritate mai mica decat componentele din clasa i , in sensul ca cele din clasa i trebuie montate primele.



- 1. Chassis**
- 2. Motherboard**
- 3. CPU / GPU**
- 4. Cooler**
- 5. RAM / Battery / Hard disk / Wifi**
- 6. Speakers / Screen**
- 7. Keyboard / Webcam**
- 8. Trackpad**

Flow-ul datelor

Pentru a controla ordinea in care trec datele, am folosit un automat cu stari finite (Finite State Machine).

Automatul are mai multe atribute:

```
eqClass = {0:{"Chassis":0},
            1:{"Motherboard":0},
            2:{"CPU":0,"GPU":0},
            3:{"Cooler":0},
            4:{"RAM":0,"Battery":0,"Hard disk":0,"WiFi":0},
            5:{"Speakers":0,"Screen":0},
            6:{"Keyboard":0,"Webcam":0},
            7:{"Trackpad":0}
            }

- implementarea claselor de echivalenta de mai sus si gestiunea starilor (automatul are 8 stari)

gg = {0: "Looks fine.",
      1: "Nice move.",
      2: "Great!",
      3: "Cool! It's a pleasure to work with you.",
      4: "You have skills",
      5: "Amazing",
      6: "You've placed that perfectly",
      7: "Good job!",
      8: "Very accurate setup."
      }
```



- dictionarul din care asistentul alege un raspuns pentru un „Remark”
topIntent = regula cea mai potrivita
entity = tipul entitatii extrase din utterance
value = valoarea entitatii extrase din utterance
response = raspunsul dat de automat pentru o comanda
stopped = flag pentru pornirea / oprirea automatului
state = starea curenta a automatului

Automatul porneste cand utilizatorul introduce comanda de start a asistentului: „start bot”. Dupa aceea utilizatorul are posibilitatea de a introduce cele trei tipuri de comenzi: Remark, Question si List Components, iar la final automatul se va opri utilizand comanda „stop bot”.

Tranzitia automatului dintr-o stare in alta se face intr-o metoda de actualizare starii, atunci cand anumite conditii sunt indeplinite: automatul nu este oprit (stopped = False), entity apartine clasei de echivalenta corespunzatoare clasei curente si componenta nu a mai fost pusa deja.

Daca utilizatorul introduce o comanda de tip Remark, pe baza propozitiei introduse, LUIS returneaza un rezultat de tip JSON, din care este obtinut topIntent-ul si entitatea. Dupa aceea, daca a fost introdusa o propozitie corecta, adica utilizatorul a pus o componenta potrivita, automatul trece in starea urmatoare si afiseaza un raspuns in functie de componenta introdusa: daca componenta este corecta, afiseaza un mesaj pozitiv (de exemplu, „You've placed that perfectly”), iar in celalalt caz afiseaza „Component is not expected at this step”.

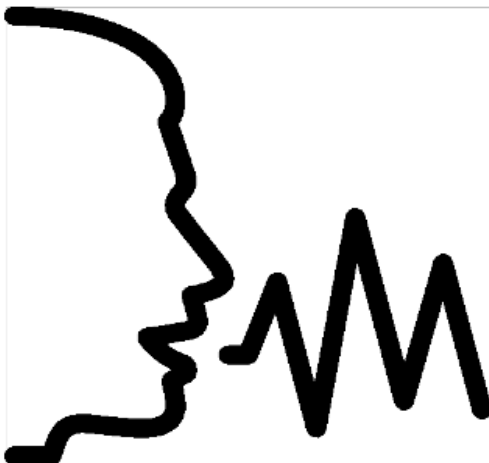
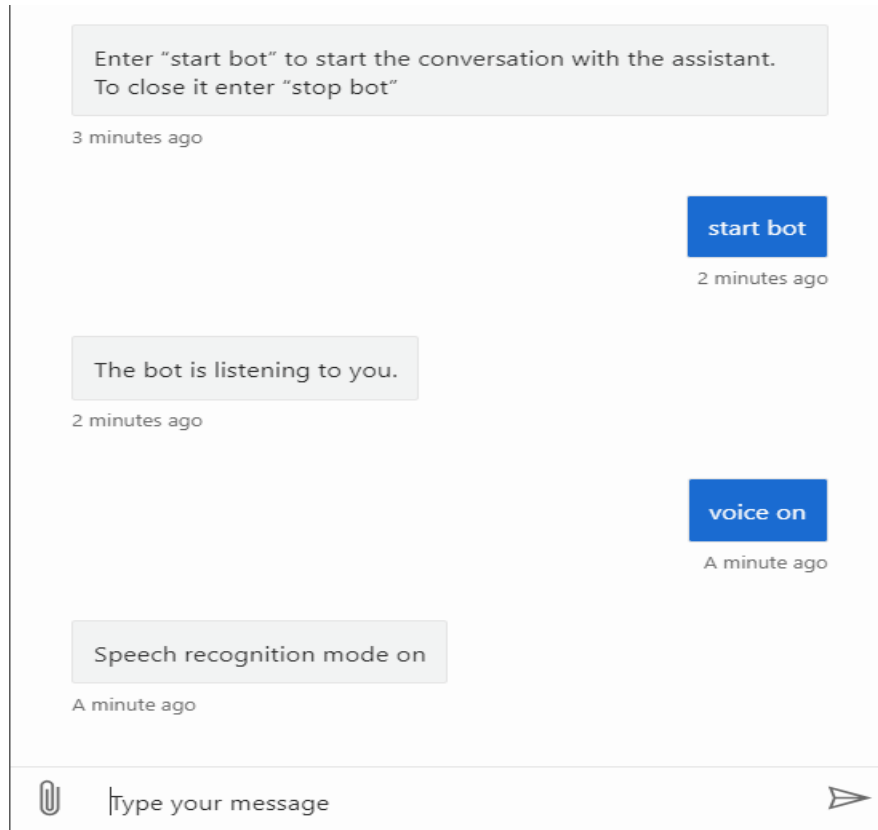
Daca input-ul utilizatorului este o comanda de tip Question, automatul verifica care a fost ultima componenta introdusa si pe baza acestui component afiseaza un mesaj de tip: „At this step you have to place the following:”.

Daca utilizatorul are nevoie de o lista cu toate componentele ramase si introduce o comanda de tip List Components, automatul verifica intr-un dictionar toate componentele care nu au fost puse si afiseaza o lista cu piesele ramase.

Pentru feedback-ul pozitiv, dat de asistent, am folosit un dictionar in care avem mai multe propozitii. La fiecare feedback pozitiv se genereaza un numar random si pe baza acestui numar se alege o propozitie din dictionar si se afiseaza in chat.



Exemple



“Voice off”



