*Babes-Bolyai University Cluj-Napoca Romania*

*Facultatea de Matematica-Informatica*

***Plan your holiday – Holiday Planner***

***Natural language processing application***

***MIRPR Report***

***Membrii echipei:***

Galan Radu : radu.galan1@gmail.com

Hij Ancuta : ancutahij@gmai.com

Pricope Tidor : ptidor1@gmail.com

Szarvas Robert : robisz98@gmai.com

*2020*

***Holiday-Planner***

1. Descrierea problemei

Scopul proiectului este realizarea de sistem inteligent de recomandare si planificare a vacantelor. Acesta va beneficia de mai multe functionalitati responsabile cu usurarea procesului de organizare a unei excursii, vacante sau a unui concediu.

Prima functionalitate se va referi la un sistem de recomdare a unei locatii ( oras sau statiune ) de vizitat in functii de preferintele personale, preferinte care pot fi exprimate prin cuvinte sau imagini. Software-ul va primi fie o imagine, fie un text sau chiar ambele si va incerca sa ofere cea mai buna recomandare posibila bazandu-se pe o baza de date existenta continand locatii, articole despre locatii si alte revizii.

Cea de-a doua functionalitate se va ocupa cu activitatea in locatia respectiva, si anume va fi capabila sa recomande un traseu turistic bazandu-se pe preferintele utilizatorului, si desigur cunostiintele sale referitor la o varietate de locatii si obiective turistice din fiecare.

Ambele functionalitati se bazeaza foarte mult pe date, obtinute in principal din surse web folosind tehnici de extragere a textului ( scrapping ). Software-ul va beneficia de o baza de date deja existenta continand toate datele necesare metodelor de recomandare, si anume o lista de locatii, o lista de obiective in fiecare locatie, o lista de texte reprezentative din fiecare locatie, o lista de atribute pentru fiecare obiectiv turistic.

Ca un exemplu, caz de utilizare, un user introduce urmatorul text:  *„ I wish to go with my family in a warm place where my children can swim and where my husband can Gamble. Also I want this place to be in the United States. Somewhere in California should do the trick. The budget is ' around 10 thousand dollars and we want to go this summer. „* Algoritmul identifica cuvintele cheie, featurile: gamble, warm, family; locatia: California si bugetul (money) : 10k $. Cauta apoi pe tripadvisor dupa cuvintele cheie si locatie, extrage primele 2 locatii si hoteluri, planifica drumul pana acolo folosind locatia curenta. In plus, avand acum locatia, putem sa planificam vacanta o data ajuns acolo – vizite la principalele atractii turistice, parcuri, locuri de distractie.

Un alt exemplu de functionalitate, la acelasi query cu cel de mai sus, facem scrapping pe internet pentru texte relevante si comparam similaritatea (cu query-ul) pentru a gasi o locatie-hotel.

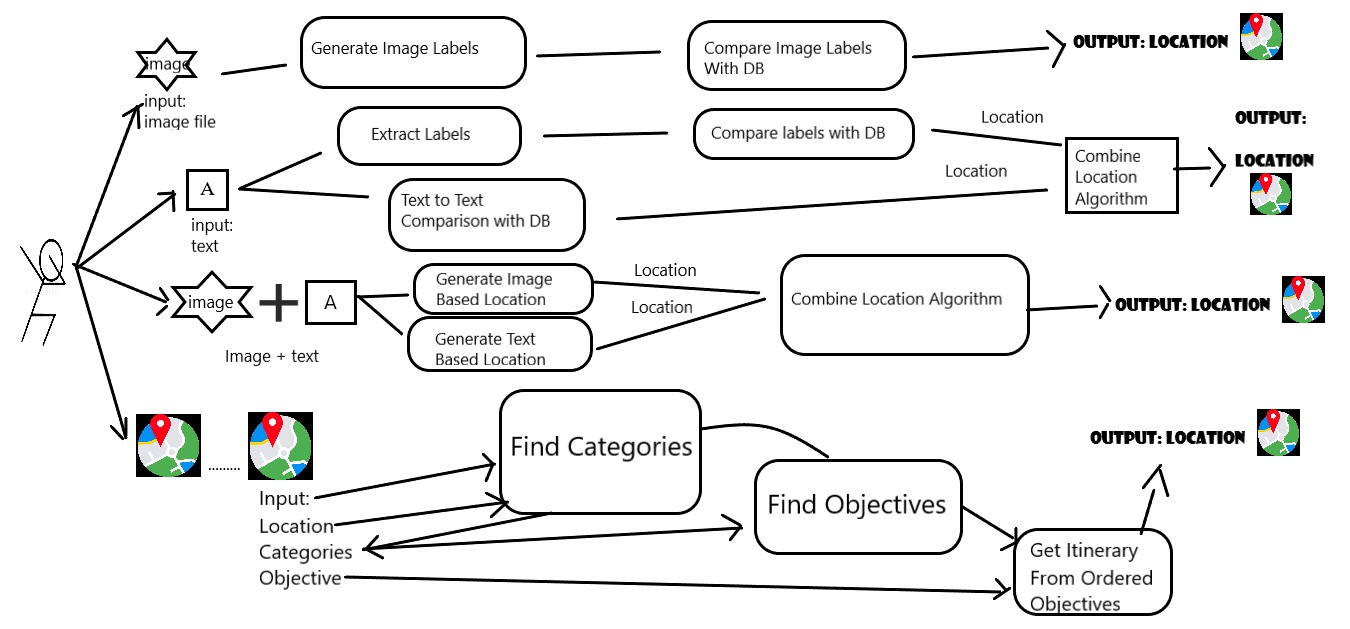
***Arhitectura applicatiei :***

Metoda de recomandare a locatiei bazata pe imagine va extrage etichete reprezentative imaginii, iar etichete le va compara apoi cu etichete existente pentru fiecare locatie obtinand o lista de locatii ordonata dupa un nivel de incredere.

Metoda de recomandare a locatiei bazata pe text va folosi doua metode pentru a-si spori eficienta: una se va baza pe algoritmi de comparare text cu text, iar cealalta pe algoritmi de comparare label cu label ( din nou pe baza etichetelor deja existente pentru locatii ) si va obtine tot o lista de locatii ordonata.

Metoda de recomandare a locatiei bazata pe text si imagini se va folosi de ambele metode si luand in considerare increderea pentru fiecare locatiei va efectua o pseudo-interclasare a rezultatelor pentru a obtine lista finala a locatiilor.

Metoda de recomandare a unui traseu turistic va avea mai multe etape de interogare a utilizatorului: in primul rand va solicita locatia in care traseul va avea loc. Sistemul va returna o lista de categorii disponibile la locatia respectiva si va solicita selectia catorva ( minim 1 ) categorii pentru a continua. In ultima etapa sistemul va returna o lista de obiective turistice ordonate dupa popularitatea lor, toate avand o mica descriere, un program de activitate, o durata de vizitare si o scurta descriere. Utilizatorul va selecta obiectivele de care este cel mai interesat, iar sistemul va returna pentru o ultima oara itinerariul calatoriei si o imagine cu traseu acestuia luand in considerare traficul optim, programul fiecarei locatii, timpul de deplasare de la un obiectiv la altul, timpul mediu de vizitare al fiecarui obiectiv si importanta pe care i-am oferit-o.



Figură Design-ul aplicatiei

**Lista functionalitati (non technical):**

-scrapping pentru a extrage informatii relevante de pe internet

-cu informatiile obtinute testam similaritatea cu query-ul userului

-extragere feature-uri dintr-un text in limbaj  natural folosind aylien

-mijloc de transport pana la destinatie, in cazul transportului cu masina, planificam ruta

-extragerea de pe internet a principalelor activitati/atractii turistice disponibile in zona

-extragerea feature-urilor dintr-o imagine

**Lista functionalitati (technical):**

-scrapping pentru a extrage informatii relevante de pe internet folosind **UiPath**

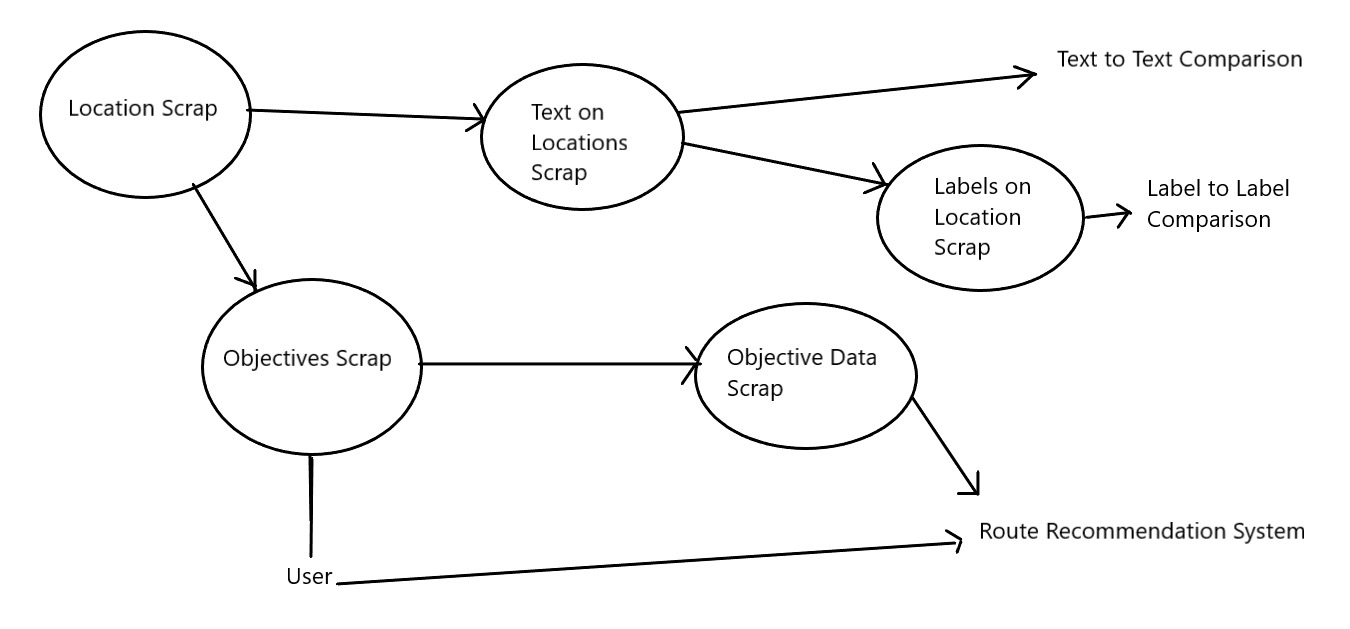
UiPath este un software de automatizare a proceselor business pe care l-am utilizat in special pentru scrappuirea/extractia link-urilor de pe site-urile web cu pagini multiple. ( Extragerea locatiilor si Extragerea link-urilor mai multor obiective turistice ).



Figură Python Scrapping [sursa](https://www.google.com/search?biw=1396&bih=657&tbm=isch&sxsrf=ACYBGNTsfCFdWJFaLxLxHCfrOkHtbwvvNw%3A1578964624778&sa=1&ei=kBYdXuCQL4HbsAf84LiwDw&q=python+scrapping&oq=python+scrapping&gs_l=img.3...59606.62140..62272...1.0..0.132.1450.16j2......0....1..gws-wiz-img.......0i67j0j0i7i30j35i39j0i30j0i19j0i5i30i19j0i8i30i19j0i10i24j0i24.l21lfWJaV5U&ved=0ahUKEwjgmMrM9YHnAhWBLewKHXwwDvYQ4dUDCAc&uact=5)

Figură Uipath [sursa](https://www.google.com/search?q=uipath&sxsrf=ACYBGNRZQT6MoeZJYkf7IHCF80z_KEsQrw:1578964612313&source=lnms&tbm=isch&sa=X&ved=2ahUKEwj3vtHG9YHnAhXowqYKHVg8CQMQ_AUoAXoECBIQAw&biw=1396&bih=657)

Beautiful Soap este o librarie din python pe care am folosit-o impreuna cu alte cateva librarile de creare a cererilor http si emulare a unui browser. L-am folosit pentru extragerea de text specific din pagini web construite dupa acelasi sablon de pe blog-uri de vacanta ( pentru textul reprezentativ a unei locatii ) , de pe site-uri de recomandari turistice ( pentru lista de obiective ) , de pe site profesionist de tinere a evidentei obiectivelor turistice ( pentru detalii despre obiective folosite in construirea itinerariului unei rute de vizitare ).



**GUI**

Interfata grafica este una simpla , care are scop exemplificativ ( proof of concept ) al procesului si al functionalitatii ideii.

Am folosit PySimpleGui , o librarire python construita pe un gui deja existent pentru a simplifica structura. Floew-ul ferestrelor este unul simplu si bazat pe happy flow-ul initial si mock-up-urile create la inceput.

Elementele principale folosite sunt organizate in layere, predominant sunt elemente de input ( text, path ) si elemente de output precum un tabel sau o imagine.

Toate metodele din .Gui vor apela numai metode din .service pentru conservarea arhitecturii aplicatiei.

O ilustrare a GUI-ului este prezenta la final de report pe partea de scenarii de rulare.

-cu informatiile obtinute testam similaritatea cu query-ul userului..

*a)* *Comparare text cu text*

Unui utilizator i se va recomanda un top al celor mai relevante locatii de vizitat  in functie de similaritatea textuala gasita intre textul introdus de acesta si descrierea locatiilor pastrate in  baza de date. Se propun doua metode prin care textul utilizatorului si cele din baza de date sunt comparate obtinandu-se astfel o similaritate intre acestea.

***i) Abordarea folosind libraria Gensim***

Libraria Gensim, folosita in special asupra task-urilor legate de NLP, cuprinde numeroase topice cum ar fi: document indexing, topic modelling, similarity retrieval with large corpora si multe altele.

**Corpus** – o colectie de documente utilzata in antrenarea modelelor

**Vector Space Model (VSM**) - a mathematically convenient representation of a document. Fiecare document poate fi privit ca un array de features ( format din perechi cheie-valoare). Cheia e reprezentata de id-ul cuvantului si valoarea de numarul sau de aparitii in acel document. Libraria Gensim omite perechiile pentru care valoare e egala cu 0.

**Model** – un algoritm pentru transformarea vectorilor dintr-o reprezentare in alta.

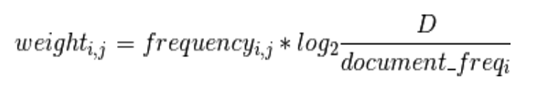
Implementare:

1.      Segmentarea cuvintelor folosind libraria jieba. Se primeste textul utilizatorul si apoi este segmentat si pastrat intr-o lista.

2.      Obtinerea corpus-ului folosind un dictionar:

Corpus: [[(0, 5), (1, 1), (2, 2), (3, 1), (4, 1), (5, 1)], [(0, 2), (2, 3), (6, 2)]]

3.      Utilizarea modelului TF-IDF pentru procesarea corposului. Modelul TF\_IDF este calculat prin multiplicarea frecventei termenului (componenta locala) cu frecventa inversa a documentului (componenta globala).



sau



***ii) Abordarea folosing Parallel Dots API***

Parallel Dots APIs ofera o serie de utilitare in domeniul Text Analysis si Visual Intelligent.  In legatura cu Text Analysis acesta ofera suport pe partea de NLP asupra textului raw si ofera o serie de APIs utile pentru analiza textului. Unele dintre cele mai importante sunt:  extragerea de keywords, sentimentul textului, sarcasmul textului, etc.

Similaritate intre doua texte:

Pentru calcularea similaritatii intre 2 texte, Parallel dots ia in calcul si similaritatea lexicala, dar si cea semantica.

1. Similaritatea lexicala nu ia in calcul sensul cuvintelor sau sensul unei intregi fraze intr-un

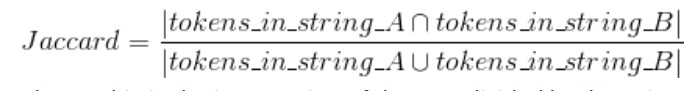
anumit context. Similaritate lexicala poate fi calculata folosind diferite granularitati: la nivel de caracter,

la nivel de cuvant sau la nivel de fraza.



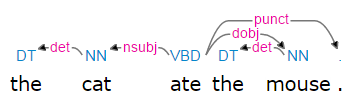
In imagine este evidentiata similaritatea la nivel de cuvant

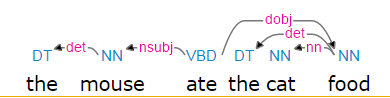
Cateva dintre cele mai importante metrici pentru calcularea similaritatii intre doua bucati de text sunt: similaritatea Jaccard, Dice si Cosine.



1. Similaritatea semantica presupune aflarea unui coeficient care descrie cat sunt de similare doua

propozitii in inteles.





Pentru aflarea similaritatii intre doua texte se face un HTTP request la endpoint-ul:<https://apis.paralleldots.com/v4/similarity> . Ca si parametri, request-ul asteapta cele 2 texte (cel putin cate 2 cuvinte in fiecare) si api-key-ul din cont. API-ul va returna ca raspuns un json cu un scor intre 0 si 1.

Benchmarks:

Se vor considera urmatoarele perechi de texte:

Test #1

1.      I would like to visit a sunny beach in California.

2.      Malibu is one of the warmest beaches from California.

Similaritatea obtinuta de modelul din Gensim: 0.3

Similaritatea obtinuta de Parallel Dots API: 0.71

Test #2

1.      I want to plan a surprise trip for me and my family. My husband is a hiking lover and my children adore long walks in the forest.I want to go in a new place like California to see giant sequoia trees for the first time. There is no budget limit.

2.      Mariposa Woods are a grove of giant sequoias. When you arrive you will feel like you are in a fairy-tale as the trees get taller and taller the further in you go. Some of the trees are between 1900 and 2400 years old with the star of the show being the giant sequoia named ‘Grizzly Giant’.

Similaritatea obtinuta de modelul din Gensim: 0.01

Similaritatea obtinuta de Parallel Dots API: 0.65

Test #3

1.      I wish to go with my family in a warm place where my children can go to the pool and where my husband can play poker. Also I want this place to be in the United States. Somewhere in California should do the trick. We would like to spend 10 thousand dollars and we want to go this summer.

2.      Santa Cruz is one of the surfing mecca’s of the United States. The city itself is only small with a population of approximately 50.000 but the beautiful beaches and bohemian vibe attracts many visitors from across the world. The all year summer feel adds to the feeling of never ending paradise that you will always feel in this city.

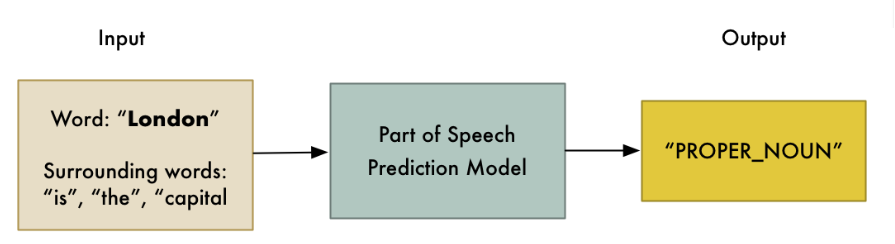
Similaritatea obtinuta de modelul din Gensim: 0.0

Similaritatea obtinuta de Parallel Dots API: 0.71

Extragere feature-uri dintr-un text in limbaj  natural folosind aylien

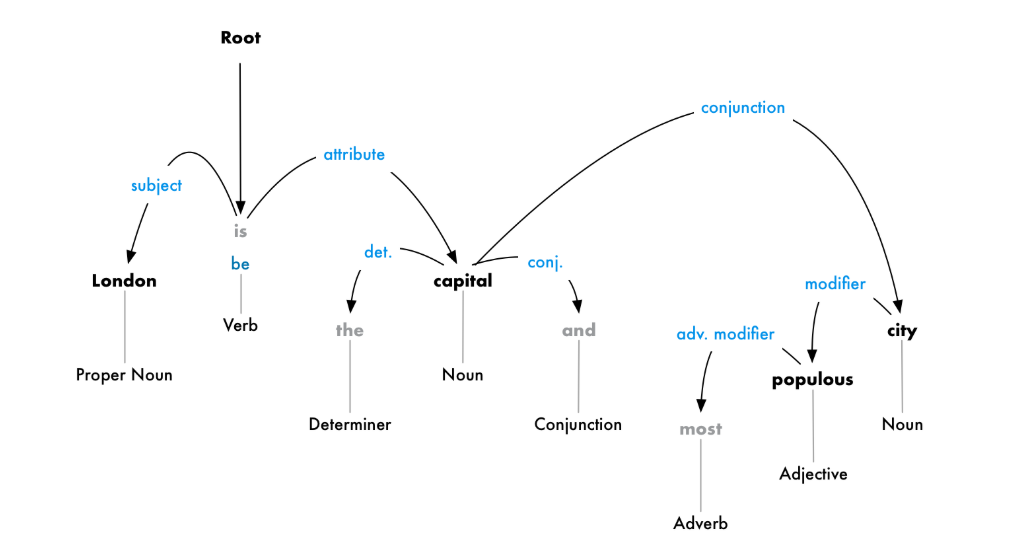
Felul in care functioneaza Aylien in spate consta in cativa pasi pe care ii putem descrie foarte succint.

1. Sentence Segmentation: despartim textul dat ca input in mai multe propozitii
2. Word Tokenization: putem intelege acest pas printr-un exemplu: “London is the capital and most populous city of England and the United Kingdom.” se va transforma in: “London”, “is”, “ the”, “capital”, “and”, “most”, “populous”, “city”, “of”, “England”, “and”, “the”, “United”, “Kingdom”, “.” Acest proces este banal pentru textele in limba engleza, putand face numai split.
3. Predicting Parts of Speech for Each Token: in continuare, vom analiza fiecare simbol și vom încerca să-i ghicim partea de vorbire - indiferent dacă este un substantiv, un verb, un adjectiv și așa mai departe. Cunoașterea rolului fiecărui cuvânt din propoziție ne va ajuta să începem să aflăm despre ce vorbește propoziția. Putem face acest lucru prin alimentarea fiecărui cuvânt (și a unor cuvinte suplimentare în jurul lui pentru context) într-un model de clasificare parțial pre-instruit:

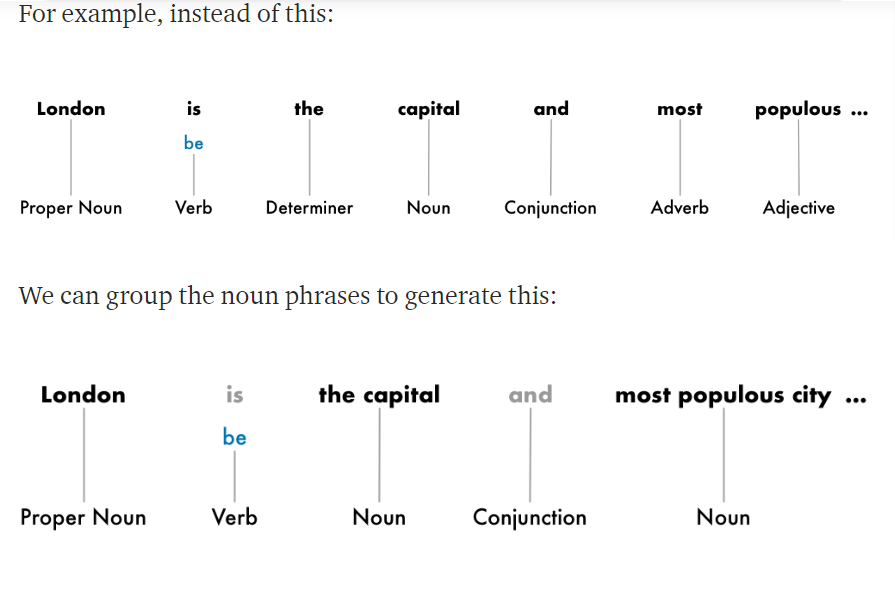


Figură Extragerea substantivelor propria

1. Text Lemmatization: În engleză (și în majoritatea limbilor), cuvintele apar sub diferite forme. Uită-te la aceste două propoziții: Aveam un ponei. Am avut doi ponei.Ambele propoziții vorbesc despre substantivul ponei, dar folosesc diferite inflexiuni. Când lucram cu text într-un computer, este util să cunoaștem forma de bază a fiecărui cuvânt, astfel încât să știți că ambele propoziții vorbesc despre același concept. În caz contrar, șirurile „ponei” și „ponei” arată ca două cuvinte total diferite pentru un computer. În NLP, apelăm la găsirea acestei lematizări a procesului - descoperind forma cea mai de bază sau lema fiecărui cuvânt din propoziție. Același lucru se aplică și verbelor. De asemenea, putem lemmia verbele găsind forma lor rădăcină, neconjugată. Deci „Am avut doi ponei” devine „Eu [am] doi [ponei]”. Lumematizarea se face, de obicei, având un tabel de căutare a formelor lema de cuvinte bazate pe partea lor de vorbire și, eventual, având câteva reguli personalizate pentru a gestiona cuvinte pe care nu le-am văzut niciodată.
2. Identifying Stop Words: Când facem statistici despre text, aceste cuvinte introduc mult zgomot, deoarece apar mult mai frecvent decât alte cuvinte. Unele conducte NLP le vor marca sub formă de cuvinte de oprire - adică cuvinte pe care poate doriți să le filtrați înainte de a face orice analiză statistică.
3. Dependency Parsing: Următorul pas este să ne dăm seama cum se referă toate cuvintele din propoziția noastră. Aceasta se numește analiză de dependență. Scopul este de a construi un arbore care să atribuie un singur cuvânt părinte fiecărui cuvânt din propoziție. Rădăcina arborelui va fi verbul principal din propoziție. Iată cum va arăta începutul arborelui de analiză pentru propoziția noastră:



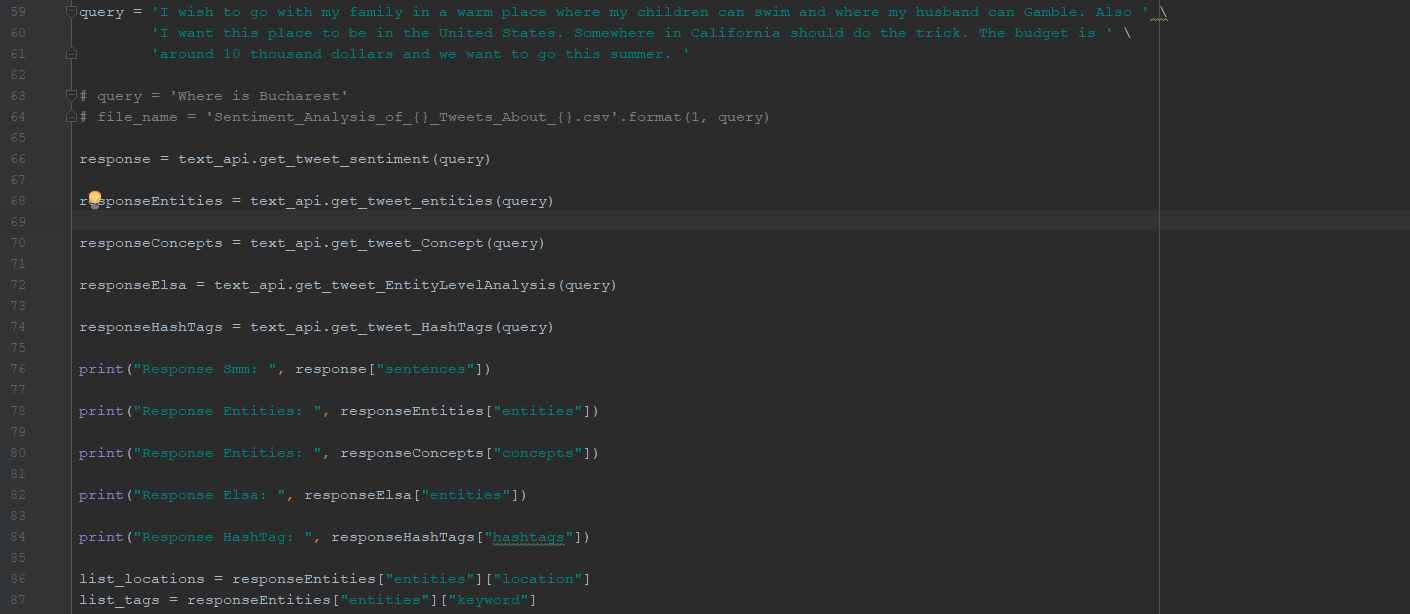
Figură Arbore de analiza pentru o propozitie [sursa](https://medium.com/@5hirish/dependency-parsing-in-nlp-d7ade014186)

1. Finding Noun Phrases: Până acum, am tratat fiecare cuvânt din propoziția noastră ca o entitate separată. Dar, uneori, are mai mult sens să grupăm cuvintele care reprezintă o singură idee sau lucru. Putem folosi informațiile din arborele de analiză al dependenței pentru a grupa automat cuvinte care vorbesc toate despre același lucru. 

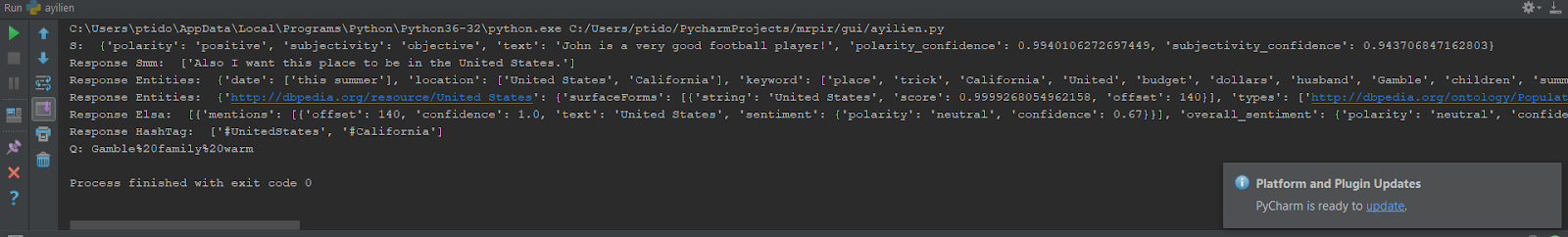
Figură Finding noun phrases

1. Named Entity Recognition (NER): Unele dintre aceste substantive prezintă lucruri reale în lume. De exemplu, „Londra”, „Anglia” și „Regatul Unit” reprezintă locuri fizice pe o hartă. Ar fi frumos să putem detecta asta! Cu aceste informații, am putea extrage automat o listă de locuri din lumea reală menționate într-un document folosind NLP. Scopul recunoașterii entității numite sau NER este de a detecta și eticheta aceste substantive cu conceptele din lumea reală pe care le reprezintă. Iată cum arată fraza noastră după rularea fiecărui simbol prin modelul nostru de etichetare NER.
2. Coreference Resolution: Engleza este plină de pronume - cuvinte ca el, ea și ea. Acestea sunt comenzi rapide pe care le folosim în loc să scriem nume repetate în fiecare propoziție. Oamenii pot urmări ceea ce reprezintă aceste cuvinte pe baza contextului. Dar modelul nostru NLP nu știe ce înseamnă pronumele, deoarece examinează doar o propoziție la un moment dat. Rezolvarea coreferențelor este una dintre cele mai dificile sarcini de implementat. Este chiar mai dificil decât analizarea frazelor. Progresele recente în învățarea profundă au dus la noi abordări mai exacte, dar încă nu este perfectă.

Ca un exemplu de functionare, avem textul urmator pe care aplicam modelele oferite de framework:



Acest cod va avea ca output (keywords, sentiment analysis):



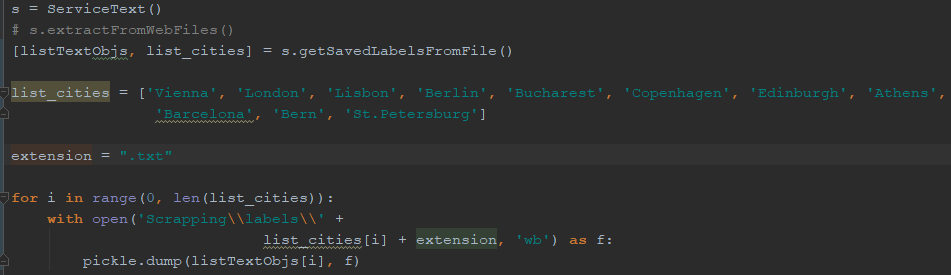
*Cum vom folosi totusi aceste functii?*

**Modul de functionare a recomandarii pe baza de text**

Avand functia de baza…textToLabel care extrage cuvintele cheie dintr-un text de orice dimensiune in totalitate sau pe paragrafe de 3-4 propozitii, putem sa testam cat de similire sunt aceste labeluri cu cele extrase din baza de date de descrieri despre locatii obtinute prin scrapping. Avem datele scrappuite din 12 locatii distincte (orase mari si mici din Europa). Dimensiunea textelor difera de la cateva linii la cateva zeci. In cazul in care textile sunt prea mari, o eventuala spargere in paragrafe este posibila. Din aceste texte, ne intereseaza doar cuvintele cheie a lor intrucat pe asta vom face comparatie.

Vom trece prin fiecare file si vom apela functia textToLabel care ne va returna un **TextObject**, un obiect propriu definit, format dintr-o lista de **EntityProb** (obiect care reprezinta perechea (cuvant cheie : String, factorul de incredre : numar real din (0,1)), date despre locatie – date spatiale, date temporale, date legate de buget. Acest **TextObject** il serializam si il salvam intr-un file care are numele locatiei cu ajutorul lui *pickle* din python.

Figură Modul de functionare text



Pentru a obtine similaritea sunt folosite 2 metode: prima cu **parallel dots**, a doua folosind **distanta Levenshtein** intre doua cuvinte, frameworkul **nltk wordnet** pentru identificarea de sinonime dintre cuvinte.

Distanta Levenshtein dintre doua cuvinte este un algoritm simplu care ne asigura o metrica pentru similaritate. Acest algoritm va returna numarul de caractere care mai trebuie adaugate, sterse sau inlocuite pentru ca cele doua cuvinte date ca input sa devina identice. De exemplu, pentru *sun* si *sunny,* distanta va fi 2, iar pentru *sun* si *arc*, distanta va fi 3 intrucat trebuie toate 3 caracterele inlocuite.

**Nltk WordNet** este folosit pentru indica daca doua cuvinte cheie sunt substantive intrucat entitatile extrase din textul utilizatorului pot sa fie complet diferite de cele pe care le avem in baza de date dar intelesul lor sa fie acelasi, caz in care vrem sa reflectam acest fapt in metrica similaritatii.

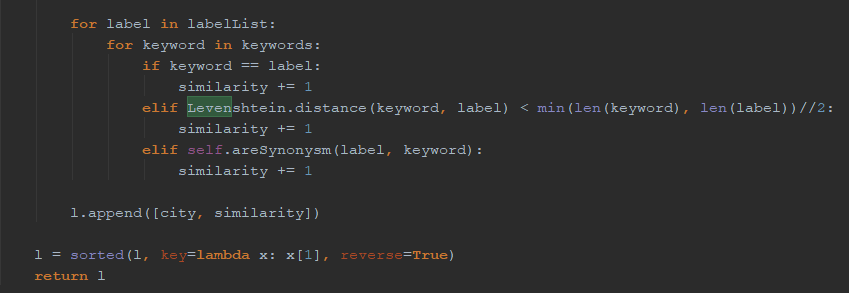
*Cum functioneaza nltk wordNet?*

WordNet este o bază de date lexicală mare a limbii engleze. Substantivele, verbele, adjectivele și adverbe sunt grupate în multimi de sinonime cognitive (sinteze), fiecare exprimând un concept distinct. Ansamblurile sunt interconectate prin intermediul relațiilor conceptual-semantice și lexicale.

Putem spune ca **WordNet** seamănă superficial cu un tezaur, prin faptul că grupează cuvinte în funcție de semnificațiile lor. Cu toate acestea, există câteva distincții importante:

* În primul rând, WordNet nu se leagă doar de formele de cuvinte - șiruri de litere - ci simțuri specifice ale cuvintelor. Drept urmare, cuvintele care se găsesc în imediata apropiere între ele în rețea sunt deambiguate semantic.
* În al doilea rând, WordNet etichetează relațiile semantice între cuvinte, în timp ce grupările de cuvinte dintr-un tezaur nu urmează niciun tipar explicit, în afară de semnificație.

Astfel, pentru fiecare label de la fiecare locatie pe care o avem in baza de date si pentru fiecare cuvant cheie din textul utilizatorului, testam similaritatea cuvintelor intre ele si la final sortam aceste locatii corespunzator.



**Modul de functionare a recomandarii pe baza de imagini**

Utilizatorul va incarca o imagine si va primi o locatie cat mai apropiata de caracteristicile ce prezinta acea imagine. Pentru asta, trebuie sa extragem labeluri din imagine. Realizarea acestui task s-a indeplinit cu ajutorul frameworkului **Watson Image Recognition** oferit de IBM.

Ceea ce se intampla in spate, practice acest framework contine foarte multe modele preantrenate de detectie a obiectelor/conceptelor dintr-o imagine. Pentru fiecare obiect, exista un model convolutional antrenat pe o baza de date de imagini cu acel obiect si nu numai. Daca un specific obiect de care avem nevoie nu are un model in sine, putem antrena noi custom o astfel de retea pentru a-l identifica. Este posibil sa existe si legaturi semantice intre labelurile extrase in sensul in care cuvantul pe care il cautam sa nu exista la lista de output dar un sinonim al acestuia/cuvant apropiat sa fie present. Varianta default a frameworkului a fost suficienta pentru realizarea functionalitatii.

Labelurile crude obtinute, astfel, trec intai printr-un filtru setat de noi de prelucrare. Aceste filtre au datoria de a elimina labeluri ca sky, color, building care vor aparea in aproape orice imagine desi nu vor fi relevante. Tipul de culoare din imagine, faptul ca avem cer sau dirt sau o constructie nu ne este de folos. Ne intereseaza tipul de landscape, de exemplu plaja/desert si tipul de constructie: zgarie-nori, cabana. Astfel, eliminam o mare parte din junk, de asemenea, vom filtra rezultatele si in functie de factorul de incredere (0,1) de care este insotit fiecare label.

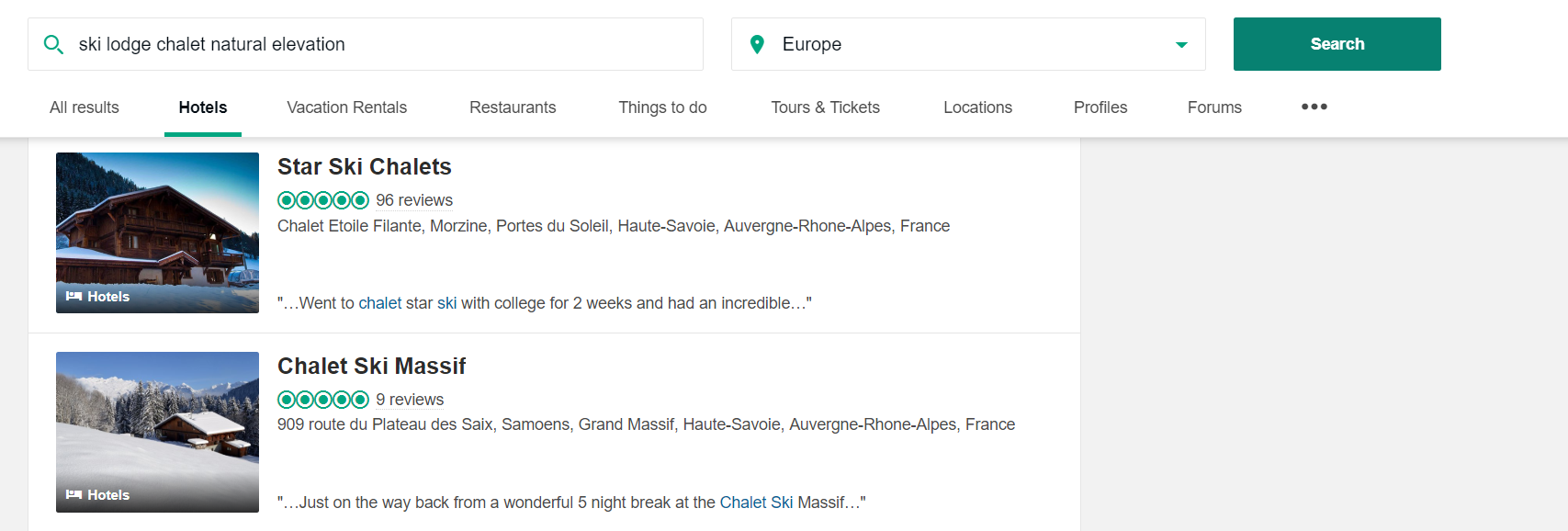
*De exemplu, ca un caz de testare, pentru imaginea:*



Figură Imagine cabana [sura](https://www.google.com/search?q=chalet&sxsrf=ACYBGNT6DGmBN8tPvHReDmN4bFfop0tcNQ:1578964981366&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjcv8729oHnAhUP_KQKHetyCxAQ_AUoAXoECA4QAw&biw=1396&bih=657)

Vom obtine ca labeluri filtrate cu o relevanta setata la >0.6: ski lodge, natural elevation, chalet (cabana).

Cu acestea, putem forma repede un query pentru trip-advisor in cautarea unui hotel/locatii pentru a compara imaginea pe care o returneaza ei (a zonei):



Figură Rezultat TripAdvisor pentru query [sursa](https://www.tripadvisor.com/)

Rezultatele sunt destul de bune, asadar, in ceea ce priveste extractia. Cum vom folosi aceste rezultate totusi, care sunt de tipul ImgObject - obiect format din label: String si factor de incredere: real (0,1).

La fel ca la text, avand baza de date cu locatiile targetate, vom folosi aceeasi functie ca la text **labelTolabelComparison** care primeste o lista de cuvinte cheie (de la text sau de la imagine) si continua in procesarea similaritatii.

1. Related work

Studiind paper-ul despre Cyberguide a lui Stefan Daniel Dumitrescu s-au identificat urmatori pasi in indexarea unui document. Indexarea unui document, algoritm folosit des in NLP, implica extragerea de keywords, knowledge retrieval si text mining. Indexarea, de asemenea, poate fi definita ca procesul de transformare a unui document intr-o lista de cuvinte. Pasii folositi in Cyberguide, care sunt folositi intr-o oarecare masura si de Api-ul Paralled Dots, sunt urmatorii :

1. Eliminarea de cuvinte care au functie gramaticala, dar nu constituie ideea documentului. Cateva exemple ar fi: conjunctiile, prepozitiile, verbele auxiliare, articolele si pronumele.
2. Pentru cuvintele ramase in document se efectueaza word stemming. Acest lucru inseamna ca se extrage radacina cuvantului. CyberGuide foloseste Porter stemmer pentru limba engleza.
3. Se construieste o retea neuronala care filtreaza cuvintele astfel incat sa ramana cuvintele legate contextului. Nodurile de input ale retelei neuronale sunt reprezentate de urmatoarele features care sunt grupate in statistical features (TF, IDF, ITF) si sintaxa gramaticala (subiect, obiect, predicat SPO).

In comparatie cu Cyberguide, aplicatia noastra Holiday-Planner foloseste o granularite diferita.

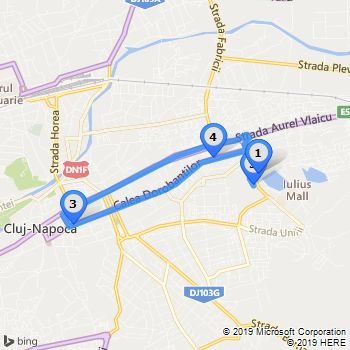
Mai exact Holiday-Planner aplica pasii de mai sus pe paragrafe de text si nu pe intregul fisier, deoarece datale noastre provin din scrapping de pe diferite site-uri, iar unele paragrafe pot coincide.

Planificarea Rutei

Mijloc de transport pana la destinatie, in cazul transportului cu masina sau pe jos, planificam rutai

Problema găsirii unui itinerariu satisfăcător s-a rezolvat consumând 2 API-uri: un API al Microsoft (Bing Maps) și un API al Facebook. Alegerile au fost făcute în urma cercetării beneficiilor oferite de mai multe API-uri, luând în calcul diferite aspecte, precum numărul de request-uri gratuite și precizia informațiilor oferite.

Figură Cod Planificarea Rute

Bing Maps API oferă un serviciu pe care nu l-am găsit la nivelul altor mari competitori (precum Google Maps) sau nu am considerat că este la fel de complex: optimizarea unui itinerariu luând în calcul un set cel puțin suficient de criterii (prioritatea de vizitare a unei locații, orele de început și sfârșit ale călătoriei, timpul preconizat de ședere, programul de vizită a locației, constrângerea vizitării unei locații specifice înaintea alteia etc.). Pe de altă parte, dezavantajele acestui API sunt limitarea privind modalitățile de realizare a unui anumit itinerariu (este permis numai cu mașina sau pe jos) și limitarea aplicării unui anumit mod pe întregul itinerariu. De asemenea, tot cu ajutorul API-ului Microsoft, am creat și o hartă ce reprezintă vizualizarea traseului planificat

Figură Ruta punct plecare, destinatie

Experimental, cel puțin pentru zona continentului european (în special a României), am descoperit și dezavatajul Bing Maps în ceea ce privește căutarea de locații pe baza unei interogări textuale, precum și lipsa unor informații despre locații. Aceasta este justificarea pentru care am cercetat ce oferă alte API-uri pentru a rezolva această problemă, ajungând să aleg serviciile oferite de Facebook, având din nou ca principale aspecte luate în considerare și de această numărul request-urilor gratuite și acuratețea informațiilor oferite. Astfel, am utilizat acest API al Facebook pentru a suplini lipsurile API-ului de la Bing Maps. Ceea ce oferă Bing Maps pe această parte funcționează bine pentru zona continentului american, dar tot este mai „sărac” în ceea ce privește anumite aspecte precum orarul locațiilor.

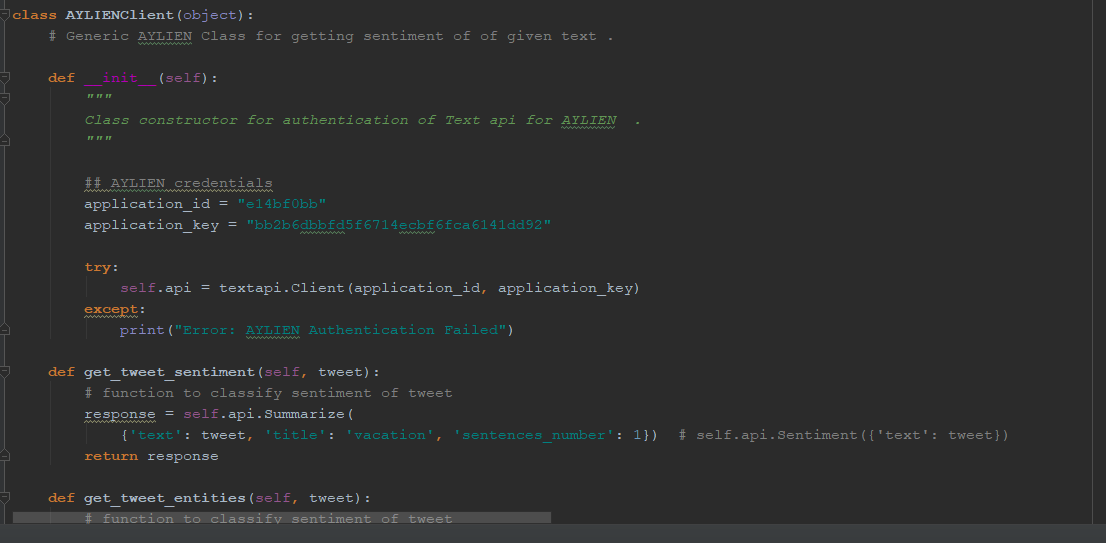
Utilizarea a 2 API-uri diferite nu prezintă niciun impediment, deoarece la nivel de backend al aplicației am realizat un API intern scris în Python ce constituie un wrapper peste cele 2 API-uri, abstractizând total tot ce înseamnă request-uri specifice și îmbinând serviciile pentru a crea funcționalitățile dorite. Întrucât API-ul de optimizare al itinerariului oferit de Microsoft este destul de flexibil, API-ul de Python construit fixează un comportament stabil. Astfel, privind API-ul din perspectiva lui de wrapper, tot codul construit mai departe va fi mult mai simplu, utilizând numai și numai cod Python, după cum se poate observa și în imaginea de mai sus ce conține un exemplu de utilizare al API-ului Python creat.

În cadrul aplicației, pentru principalele obiective turistice propuse pentru vizitare către un user, sunt utilizate date provenite din scrapping. Astfel, aceste locații sunt instanțiate prin coordonate, fără a folosi API-ul Facebook pentru regăsirea informațiilor în felul explicat mai sus. Un asemenea exemplu de creare a unei locații se poate observa în comentariul din imaginea ce prezintă lucrul cu API dezvoltat.

Așadar, această funcționalitate este construită pentru a planifica un itinerariu de vizitare a unor locații turistice, luând în calcul o serie de criterii pentru a optimiza și automatiza acest proces, astfel încât un turist să nu fie nevoit să își consume timp în vacanță pentru a planifica vizitarea mai multor obiective.

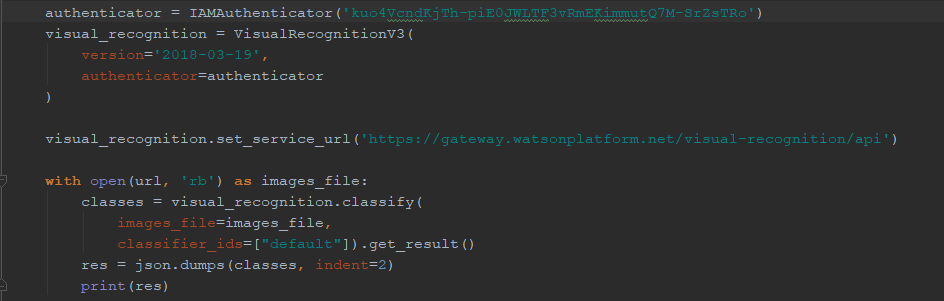
**Cum instalam software-ul?**

In cazul **Aylien**, pentru a putea folosi framework-ul trebuie sa obtinem credentiale de acces. Acestea se pot obtine la adresa [urmatoare](https://blog.aylien.com/how-to-get-started-with-aylien-text-analysis-api/). Avand credentiale, trebuie doar sa instalam **aylienapiclient**, si sa importam textapi. Intrucat modul de functionare este prin comunicare client-server, este recomandat sa cosntruima ceasta clasa generica a noastra de client:



Pentru celelalte librarii, la fel avem nevoie de credentiale pentru acces la API care se obtin de pe site-ul companiei ce detine software-ul. Instalarea ulterioara a frameworkului este evidenta.

**Cum instalam IBM API Watson NLP and Image Recognition?**

La adresa <https://cloud.ibm.com/>, trebuie facut cont si cerere pentru credentiale pentru cele doua pachete de NLP si Image Recognition.

Credentialele sunt reprezentate printr-un URL si un key specific. Acestea vor fi folosite in cod in realizarea clientului, outpurile le obtinem prin cereri HTTP la server.

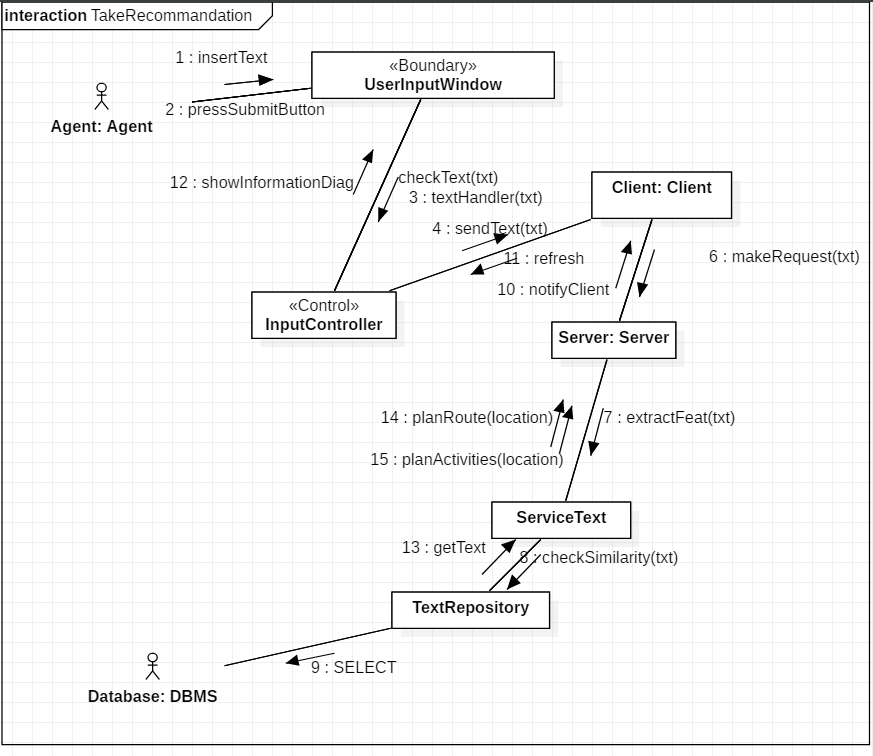
**Cum instalam Parallel Dots software?**

La adresa <https://www.paralleldots.com/>, trebuie facut cont si cerere pentru credentiale, acestea se vor primi imediat.

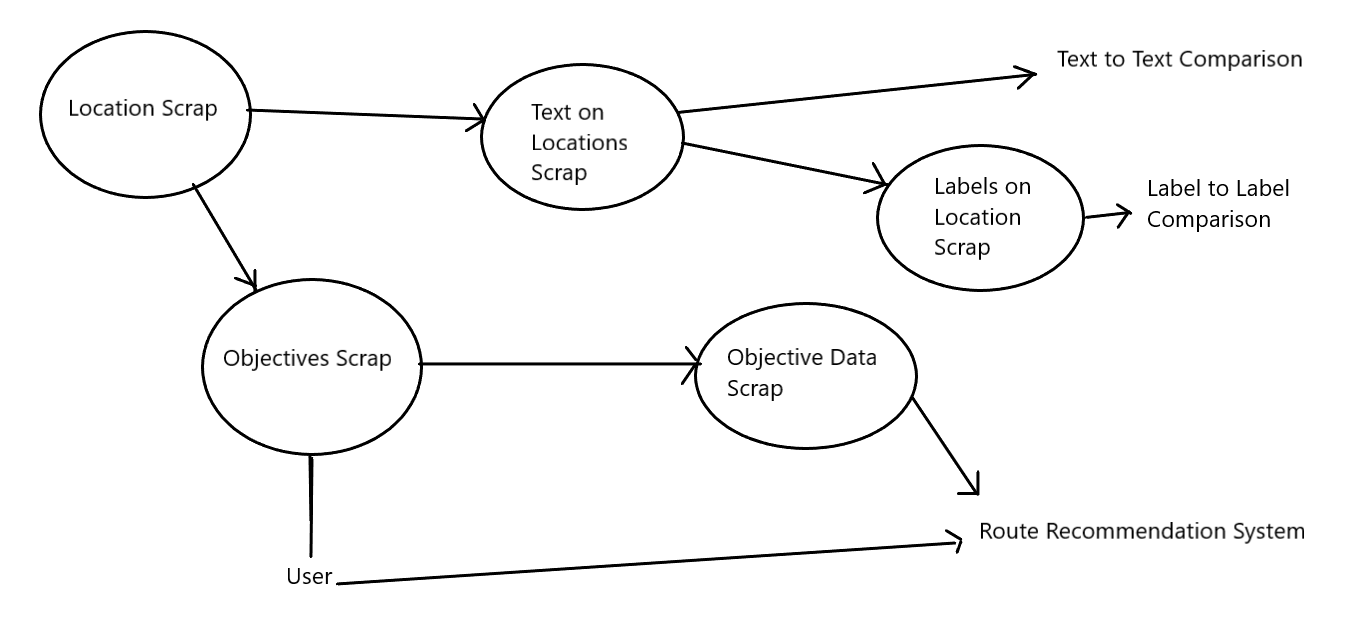
Credentialele sunt reprezentate printr-un URL si un key specific. Acestea vor fi folosite in cod in realizarea clientului, outpurile le obtinem prin cereri HTTP la server.

Flow – Diagrame de Secventa/Interactiune

-initiala:



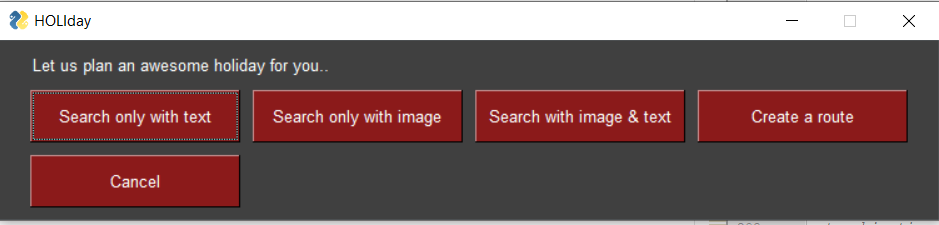
**-**finala:

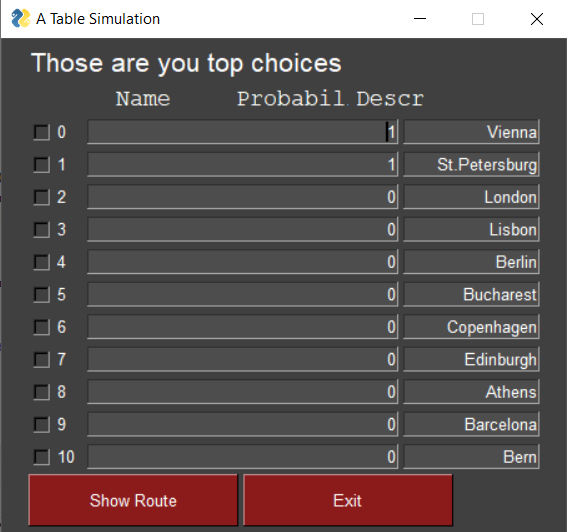


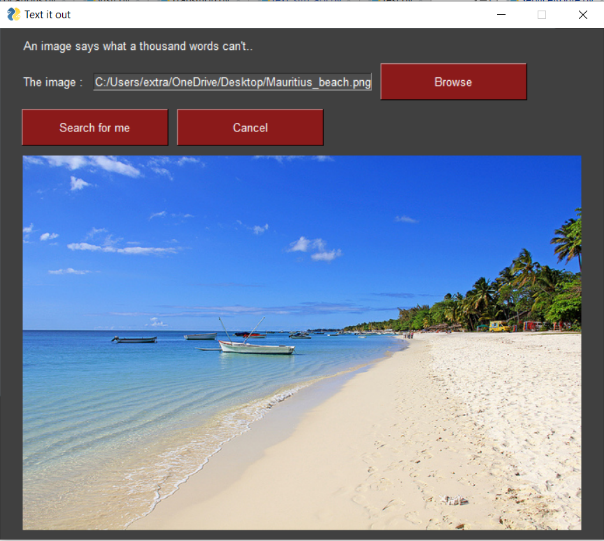
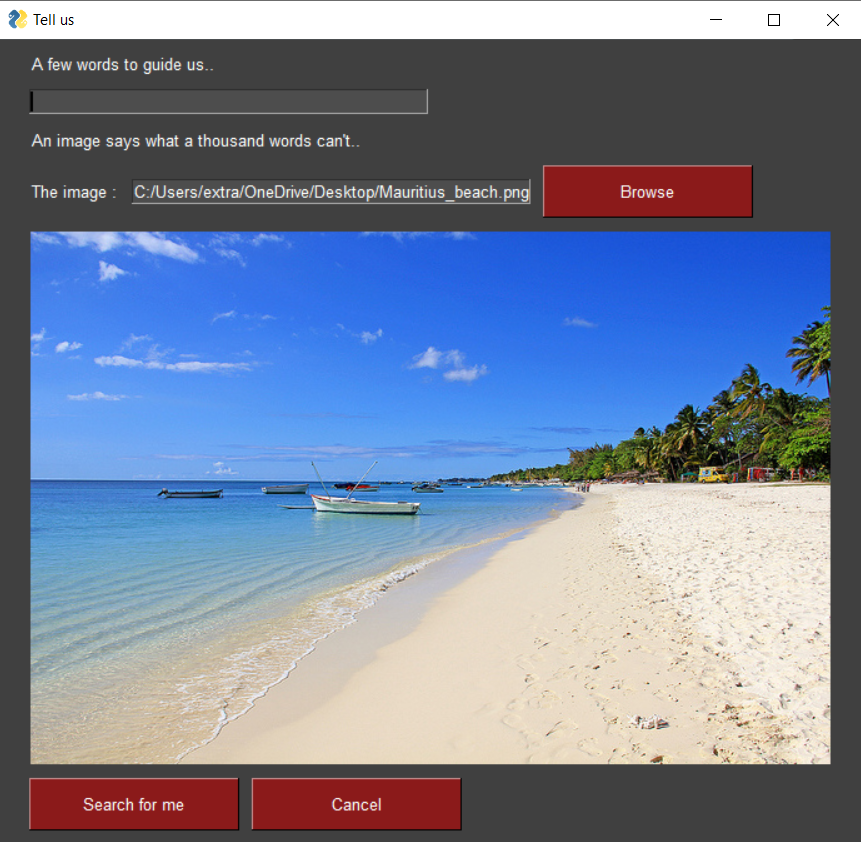
1. Scenarii de testare

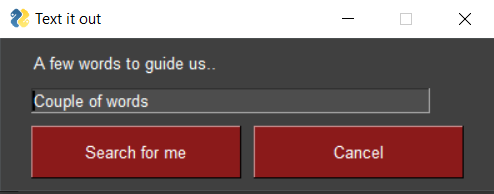
Flow-ul aplicatiei este unul simplu. Fereastra principala contine un simplu meniu care va redirectiona catre fiecare metoda in parte. Toate metodele de recomandare a locatiei se incheie cu o lista de recomandare care poate redirectiona din nou catre functionalitatea de recomandare a traselui ( deoarece foloseste ca input , output-ul acestora ).

Urmatoarele actiuni pot reprezenta actiuni care un utlizator le poate face in aplicatie:

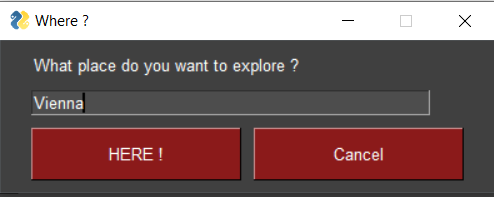
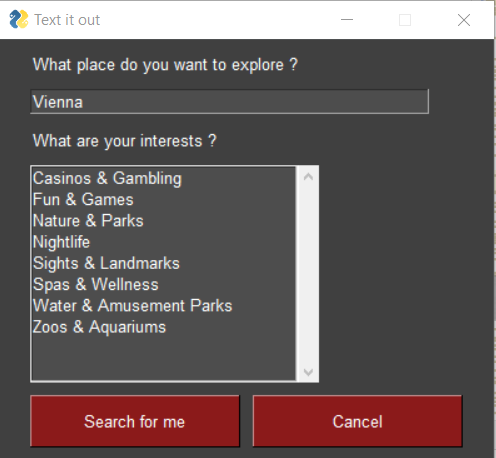


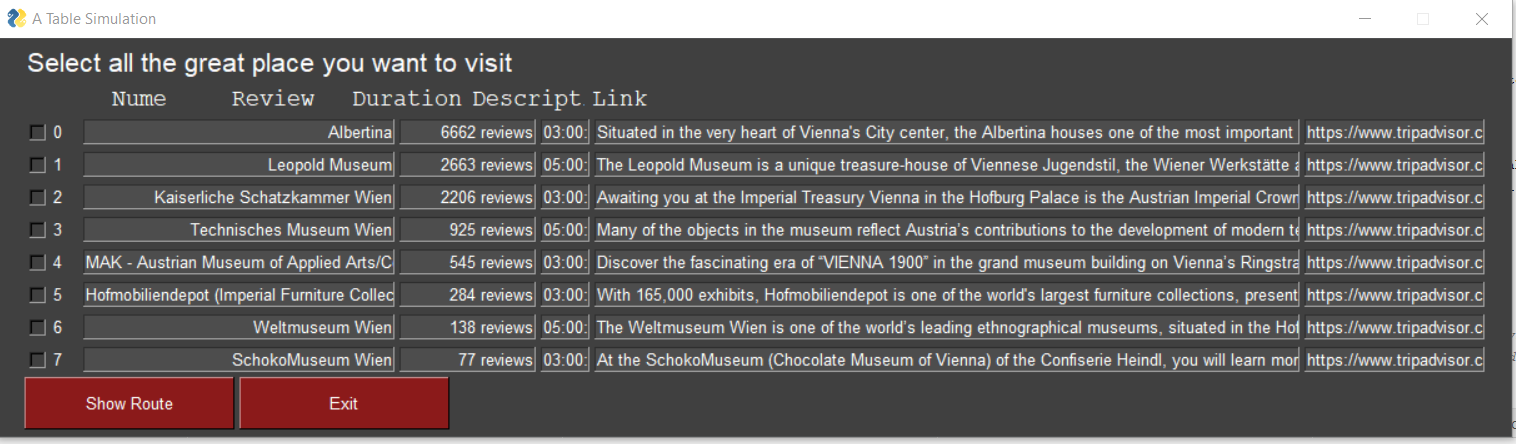
Functia de recomandare prin text, imagine si ambele :

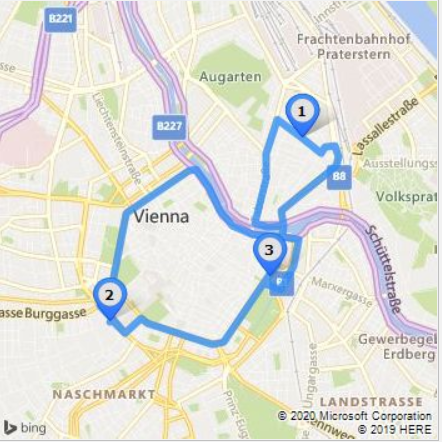
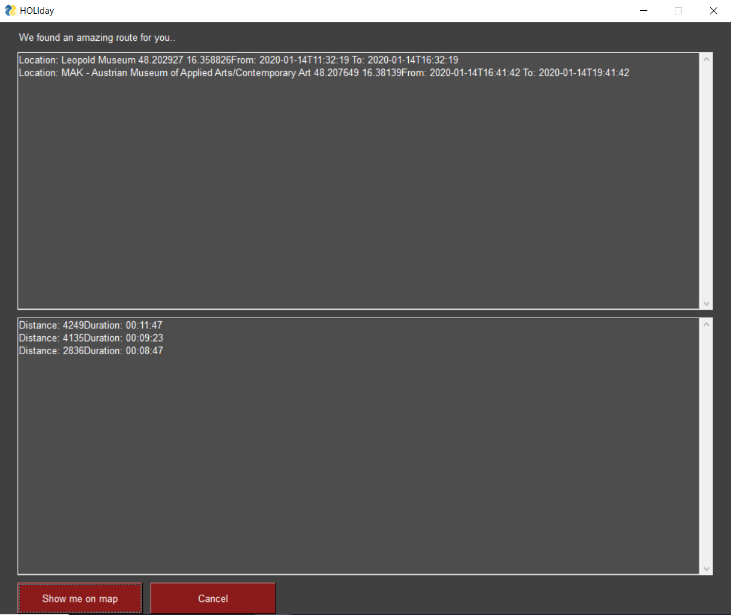




Ferestrele de recomandare a rutei:







Fereastra intermediara de asteptare:

1. Future Work

Cea mai mare bataie de cap a fost in indicarea cuvintelor cheie cu legatura semantica cu o alta multime de cuvinte, de asemenea, baza noastra de date a textelor obtinute din scrapping de pe internet este foarte limitata in momentul de fata.

O extindere la cel putin 70 de orase din lume, pe mai multe continente, ar duce la match-uri mult mai bune a cuvintelor cheie si a ideilor pe care le transmit utilizatorii atunci cand cauta o vacanta.

In cazul imaginilor, labelurile obtinute au o sansa foarte mica sa fie in acelasi camp semantic cu putinele texte analizate de pe internet pentru numarul nostru limitat de orase.

De asemenea, o imbunatarile a graficii este deosebit de importanta pentru utlizatorul de rand, accentul trebuie pus pe cum aratam ruta de la pornire la destinatie, multe imbanatatiri exista aici.

De asemenea, o analiza mai detaliata ce priveste functionalitatea de recomandare de obiective o data ce am gasit o destinatie poate fi mult mai ascutita, printr-un scrapping mult mai complex si o procesare mult mai detaliata a scopului fiecarui obiectiv candidat.

O alta idee de imbunatatire a aplicatiei ar fi urmatoarea: sa ne imaginam urmatorul scenariu: un turist doreste sa viziteze muzeele din Viena si Cluj. Pentru a ajunge dintr-o locatie in alta, turistul trebuie sa treaca prin Budapesta, unde sunt muzee de asemenea. Aplicatia ca sistem de recomandare, ar putea sa faca o sugestie despre muzeele din Budapesta.

Link to repo:

<https://github.com/robertkarol/planificator-vacanta-mirpr>

Bibliografie:

1. <https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS12/paper/viewFile/4421/4801>

1. <https://ieeexplore.ieee.org/document/1644735?tp=&arnumber=1644735&url=http:%2F%2Fieeexplore.ieee.org%2Fiel5%2F69%2F34468%2F01644735.pdf%3Farnumber%3D1644735>

1. <https://cogcomp.seas.upenn.edu/papers/DRSTV09.pdf>
2. <https://docs.microsoft.com/en-us/bingmaps/rest-services/routes/optimized-itinerary>
3. <https://blogs.bing.com/maps/2018-07/bing-maps-routing-api-knows-the-shortest-route-that-visits-all-waypoints>
4. <https://docs.microsoft.com/en-us/bingmaps/rest-services/imagery/get-a-static-map>

1. <https://developers.facebook.com/docs/graph-api/reference/location/>
2. <https://www.researchgate.net/publication/262213788_Natural_Language_Processing_applied_in_itinerary_recommender_systems>