**UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

LUCRARE DE LICENȚĂ

# Analiza enunțurilor matematice in limbaj natural

**propusă de**

***Lăzăroi Darius-Marian***

**Sesiunea:** *iulie, 2019*

**Coordonator științific**
***Lect. dr. Ionuț-Cristian Pistol***

**UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

# Analiza enunțurilor matematice in limbaj natural

***Lăzăroi Darius-Marian***

**Sesiunea:** *iulie, 2019*

**Coordonator științific**
***Lect. dr. Ionuț-Cristian Pistol***

**Avizat,**

**Îndrumător Lucrare**

Titlul, Numele şi prenumele

_____

Data _____ Semnătura

_____

## DECLARAŢIE

## privind originalitatea conţinutului lucrării de licenţă/disertaţie

Subsemntatul(a)

……

domiciliul în ……..

născut(ă) la data de ……..…., identificat prin CNP ……..……..……..., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de ……. specializarea ……, promoţia ……,

declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul:

_____

_____

_elaborată sub îndrumarea _____, pe care urmează să o susţin în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă/disertaţie să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ……  Semnătură student ……

# Cuprins

# Introducere

În această lucrare am studiat metode de procesare a limbajului natural, algoritmi de învățare automată și rețele neuronale reușind să implementez un sistem care este capabil să recunoască unul sau mai multe tipuri de enunțuri de probleme matematice , bazându-ne pe anumite exemple de antrenare. Ideea a fost următoarea: dacă se poate detecta tipul de expresie matematică a unui enunț în această formă problema este foarte ușor de rezolvat pentru un calculator. Dacă avem tipul expresiei, numeralele pot fi extrase ușor și  astfel putem obține rezultatul ecuației.

Problemele matematice sunt foarte ușor de rezolvat pentru un calculator atunci când acestea sunt în format numeric, însă, atunci când ele sunt în formulate în limbaj natural apar probleme la interpretare.

În funcție de complexitatea lor, aceste probleme matematice în limbaj natural, apar cel mai des în domeniu educațional dar este posibilă crearea unui sistem suficient de complex capabil să recunoască o varietate foarte mare de probleme. Acest sistem ar fi capabil să rezolve probleme matematice întâlnite fără a necesita cunoștințe extinse din domeniul respectiv.

Una dintre soluțiile posibile ar fi standardizarea unei structuri pentru fiecare tip de problemă, totuși, acest lucru ar fi foarte apropiat de forma numerica a unei ecuații matematice. Dacă exista o astfel de structura algoritmul extrage numerele doar numerele din această propoziție deoarece acesta cunoaște poziția lor.[10]

O altă soluție constă într-o abordare exact pe dos. Pornind de la o ecuație matematică, se încearcă generarea problemei în limbaj natural, programul cunoscând deja soluția la problemă. Desigur, o asemenea soluție ar avea nevoie de multe șabloane pentru a putea acoperi cât mai multe tipuri de probleme matematice.

Pentru procesarea limbajului natural exista 2 opțiuni principale: rețele neuronale „deep” sau învățarea automată. Rețelele neuronale sunt avantajate când baza de date pentru antrenare este foarte mare iar învățarea automată creează un model matematic precis pentru selecția acțiunilor.[11][12]

Capitolul 1 cuprinde descrierea tehnologiilor folosite și contribuția acestora la algoritmul final. Capitolul 2 explică procesul de dezvoltare a sistemului de clusterizare în detaliu, cu evaluarea performanței acestuia pe parcurs și descrierea soluției finale, chiar dacă unele părți nu au fost folosite în aceasta.

Capitolul 3 constă în antrenarea și optimizarea rețelelor neuronale pentru a obține un clasificator capabil să recunoască cel puțin 90% din propoziții. De asemenea, aici avem descris fluxul aplicației, și un mic ghid de utilizare.

# Capitolul I - Tehnologii folosite

## Limbajul Python:

La dezvoltarea aplicației am avut de ales din mai multe limbaje de programare: C/C++, Java, Python, etc. . Limbajul python a fost ales datorită simplității instalării librăriilor externe și faptul că acest limbaj a fost creat specific pentru a facilita analiza datelor și vizualizarea acestora.[1]

Versiunea aleasă a fost Python 3 datorită familiarității personale.

În momentul de față varianta 3.7 este cea mai nouă, 3.6 a fost aleasă datorită librăriilor keras și tensorflow deoarece am întâmpinat dificultăți la instalarea acestora pe versiunea 3.7. Diferențele între 3.6 și 3.7 sunt foarte minore (câteva librării cu care nu am lucrat au fost îmbunătățite, 2 librării noi adăugate: contextvars și dataclasses de care nu am avut nevoie).

Totuși acest limbaj are un dezavantaj mare , și anume viteză de execuție este mai lentă decât c/c++ și java. Această performanță are loc din mai multe motive:

- Este un limbaj interpretat și nu compilat[9]

În limbajele compilate instrucțiunile pot fi direct transformate în limbaj mașină, în limbajele interpretate instrucțiunea trebuie transformată, în funcție de tipul parametrilor, într-o altă instrucțiune care apoi va fi schimbată în cod de tip limbaj mașină. Acest strat adițional reprezintă o greutate în plus care apare la fiecare instrucțiune.

- Este un limbaj scris dinamic[9]

Într-un limbaj static, tipul fiecărei variabile trebuie specificat la declararea acesteia. În python variabilele nu necesita această declarație, nici la declararea matricelor sau structurilor de tip matrice ( Queue etc. ), putând să conțină orice tip de date la orice poziție.

Ex:
*a=10*
*a="test"*
*index = [0, "zero", ["a","b","c"]]*

Din exact același motiv menționat mai sus, clasa list are o performanță înceată atunci când aceasta conține un număr mare de elemente și încercăm să iterăm de mai multe ori prin ea sau să adăugam un element undeva în mijlocul listei.
Această ultimă problemă a fost rezolvată folosind librăria NumPy.

## Librăria NumPy

NumPy este o librărie python folosită pentru efectuarea rapidă a operațiilor matematice ,de nivel înalt, peste matrici mari și de asemenea păstrarea acestor date și accesarea mai rapidă a lor.

Numpy a fost folosit datorită numărului mare de propoziții și nevoia de a itera de mai multe ori prin ele.

Acesta este un librărie open source care a fost pusă pe piață în anul 2006 (repositoriul git având aproximativ 21000 de comiteri) fiind un succesor al modului Numeric care l-a rândul lui a fost creat în 1996.

Limbajul Python inițial nu a fost optimizat pentru efectuarea operațiilor numerice intense, dar , datorită faptului că acesta a fost creat pentru facilitarea analizei datelor și vizualizarea acestora, el a atras atenția inginerilor ducând la necesitatea creări unei librarii specifice pentru efectuarea operațiilor matematice intense.[3]

NumPy este scris în mare parte în C și împachetat cu python pentru a putea fi folosit foarte ușor.

Funcționalitatea sa principală este metoda de a crea matricele.

Fată de listele din Python care sunt dinamice și pot conține orice tip de variabila pe orice poziție, in matricele din NumPy toate elementele trebuie să aibă același tip. NumPy mai are și o reprezentare eficientă în memorie.

Totuși datorită acestei implementări adăugarea elementelor într-o matrice deja existenta nu este la fel de ușoară ca listele implicite din python (list.append(element)). De fiecare dată când se încercă adăugarea unei noi linii, coloane sau element matricea este de fapt copiată în altă matrice cu modificările respective efectuate.[2]

Transformarea din tipul list în tipul numpy.array este foarte ușoară:

```
x=[0,0,0,0]
Y=np.asarray(x)
```

Reprezentarea a fost folosită pentru transformarea listelor în matrici numpy pentru librăriile scikit-learn, keras,tensorflow

## Librăria SciKit-Learn

Scikit-learn este o librărie open source, de învățare automată având la bază NumPy, SciPy, Matplotlib. Aceasta este ușor de folosit conținând funcții eficiente pentru analiza datelor. A fost lansată în anul 2007.

Funcțiile din librărie sunt rapide deoarece limbajele folosite au fost Cython, C și C++.

Pentru partea de machine learning aveam de ales intre mai multe asemenea librării. Am ales sci-kit learn datorită performantei oferita și faptului că librăria continuă să fie îmbunătățita și actualizată destul de des (în momentul de față ultima actualizare a avut loc pe data de 17 Mai 2019).

| | scikit-learn | mlpy | pybrain | pymvpa | mdp | shogun |
|---|---|---|---|---|---|---|
| Support Vector Classification | **5.2** | 9.47 | 17.5 | 11.52 | 40.48 | 5.63 |
| Lasso (LARS) | **1.17** | 105.3 | - | 37.35 | - | - |
| Elastic Net | **0.52** | 73.7 | - | 1.44 | - | - |
| k-Nearest Neighbors | 0.57 | 1.41 | - | **0.56** | 0.58 | 1.36 |
| PCA (9 components) | **0.18** | - | - | 8.93 | 0.47 | 0.33 |
| k-Means (9 clusters) | 1.34 | 0.79 | ⋆ | - | 35.75 | **0.68** |
| License | BSD | GPL | BSD | BSD | BSD | GPL |

-: Not implemented. ⋆: Does not converge within 1 hour.

**Figura 1:** Comparația performanței unor algoritmi în mai multe librării de tip ML [3]

Din figura 1 se poate observa că bibliotecile Shogun și mlpy au o performanță mai bună pentru k-means, dar, acestea nu oferă și posibilitatea de a vectoriza datele când ele sunt în formă de text. În procesul de creare a aplicației, a existat un moment în care nu am mai avut nevoie de un vectorizator, dar datorită accesibilității, familiarității și diferenței minore de timp, am decis să rămân la scikit-learn. Din această librărie am folosit următoarele:

### Vectorizatorul TF-IDF:

Vectorizatorul TF-IDF (acronim pentru term frequency-inverse document frequency) încearcă să asigneze fiecărui termen o anumită greutate (valoare numerica) pentru a putea vizualiza mai ușor cât de important este acesta pentru întregul text. Acesta se bazează pe tehnica bag of words.[4]

De obicei într-un anumit text vor exista cuvinte care apar de foarte multe ori care însă nu sunt interesante deoarece ele nu aduc informații noi, de exemplu articolele (un, o, etc.).

Vectorizatorul încearcă să elimine aceste cuvinte, prin inversarea valorilor numerice sau prin ignorarea totală, pentru a nu influența negativ greutățile pentru celelalte cuvinte.

De asemenea după cum spune și numele(inverse term frequency) cuvintele cu o frecventa mai mică vor avea o greutate mai mare. Vectorizatorul funcționează cel mai bine pentru documente de mărime medie dar poate fi folosit și pentru propoziții.

Frecvența unui termen este calculată cu formula:

$$tfidf(t,d) = tf(t,d) * idf(t)$$ [4]

Unde idf este calculat astfel:

$$idf(t) = \log\frac{n}{df(n)} + 1$$ l[4]

Rezultatul obținut fiind normalizat cu normal Euclidiană.[4]

Datele de intrare la acest algoritm vor fi toate propozițiile de la datele de antrenare ,combinate sau separate. El returnează o matrice rară uniformă pentru fiecare exemplu de la intrare.

Această matrice va fi folosită direct pentru generarea clusterelor cu algoritmul k-means

## Tehnica bag of words:

Tehnica bag of words este o metodă de a extrage , sau de a transforma, o colecție de propoziții sau documente în valori numerice.[4]

Librăria scikit-learn folosește 3 atribute pentru a genera această valoare:

- Tokenizarea

Fiecărui cuvânt unic îi va fi asignat o valoare numerica. Cuvintele sunt separate în funcție de spațiile libere , semne de punctuație, și de limba aleasă. [4]

- Numărarea

Pentru fiecare token se calculează numărul de apariții în fiecare document , text ,propoziție etc.

- Normalizarea

După efectuarea normalizării se pot scoate caracteristicile textului. O caracteristică reprezintă de fapt numărul de apariții unui token (după modificările specifice făcute de algoritmul respectiv) în textul ales. [4]

## Vectorizatorul Hashing:

Diferența principală dintre Vectorizatorul TF-IDF și Hashing este aceea că Hashing aplică o funcție hash peste frecvența termenilor în fiecare propoziție, iar TF-IDF recalculează această frecvență în fiecare propoziție prin penalizarea termenilor care apar mai des.[4]

Un avantaj pe care Hashing îl are peste TF-IDF este rapiditatea acestuia.

Hashing încearcă să rezolve și problema când vocabularul este foarte mare. În acest caz vectorizatorul tf-idf va fi încetinit și de asemenea va ocupa foarte multă memorie. Prin utilizarea acestei metode de hașurare matricea rezultată o să aibă mereu aceeași dimensiune.

Această soluție poate duce și la apariția coliziunilor în cazul în care numărul de cuvinte depășește numărul de caracteristici selectate la rularea algoritmului. Coliziunile pot apărea și când avem mai puțin cuvinte dar șansa este mică.

Acesta a fost aplicat pe fiecare propoziție pentru a obține o matrice rară de forma 2^20 cu caracteristicile fiecărei propoziții, matrice care apoi a fost folosită pentru clusterizarea k-means.

Decizia de a folosi Hashing a fost luată după ce am făcut vectorizarea TF-IDF pentru a putea observa dacă ordinea cuvintelor va aduce un rezultat mai bun, sau cel puțin diferit.

### Clusterizarea K-means în libraria Scikit-learn:

Algoritmul k-means a fost construit, în această librărie, în jurul distanței Euclidiene. Acesta va fi folosit pentru efectuarea clusterizării la fiecare pas. El încearcă să minimalizeze inerția cu formula:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} \|x_i - \mu_j\|$$ [5]

Inerția este definită ca o metodă de măsurare a nivelului de coerență a clusterelor rezultate. [5]

In această librărie mai exista și varianta k-means cu minibatch care are o convergența mai rapidă la soluție dar calitatea rezultatului este redusă.

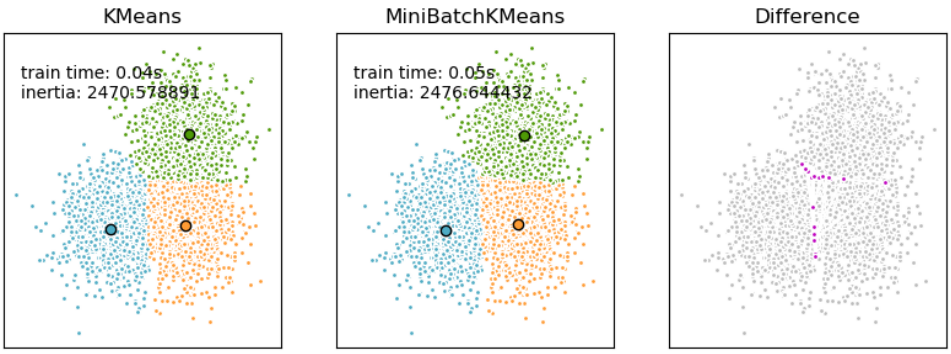



**Figura 2:** Comparația dintre K-means cu si fără minibatch [5]

Diferența rezultatelor poate varia, în *Figura 2* această diferență este foarte mică. Dar, după părerea mea, luând în considerare sensibilitatea rețelelor neuronale la datele de antrenament și diferența de timp între cei doi algoritmi avem de câștigat mai mult dacă nu folosim minibatch.

Această varianta k-means cu minibatch a fost folosită inițial, dar, după observarea diferenței de timp am decis să folosesc doar varianta normală pentru a maximiza calitatea rezultatelor obținute.

## Librăria NLTK

NLTK [1], sau natural language processing tool-kit, este o librărie folosită pentru a procesa limbajul natural. In această lucrare am folosit librăria pentru a extrage părțile de propoziție din datele de antrenare cu funcția NLTK pos_tag. Această decizie a fost luată datorită structurii acestor date.

[1] Manning, Christopher, et al. "The Stanford CoreNLP natural language processing toolkit." *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014. https://www.aclweb.org/anthology/P14-5010

De exemplu pentru următoarele 2 propoziții:
19+73|Draco has nineteen magic potions and seventy-three love potions. How many do we have now?
71+80|Ron has seventy-one cats and eighty old rats. How many do we have now?

In afară de ultima parte “How many do we have now” singurele cuvinte comune sunt “has” și “and” dar aceste propoziții au același semn.

Aici vectorizatorii de mai sus care se bazează pe frecventa cuvintelor nu sunt foarte folositori deoarece, cel mai probabil, cuvintele “has” “now” “how” și “and” vor avea o valoare foarte mică sau foarte mare(mica pentru tf-idf datorată idf). Rezultând ca aceste doua propoziții să se afle în clustere diferite.

Desigur, aceasta este doar o analiza la prima vedere, performanța proprie va fi puțin diferită. Totuși când încercam să aplicăm NLTK tagging pe aceleași propoziții avem următoarele rezultate:
Draco/NNP has/VBZ nineteen/VBN magic/JJ potions/NNS and/CC seventy/NN -/: three/CD love/NN potions/NNS ./.. How/WRB many/JJ do/VBP we/PRP have/VB now/RB ?/.

Ron/NNP has/VBZ seventy/NN -/: one/CD cats/NNS and/CC eighty/NN old/JJ rats/NNS ./..
How/WRB many/JJ do/VBP we/PRP have/VB now/RB ?/.

Se poate observa că acest rezultat este similar dacă vom considera adjectivele (JD) și substantivele (CD) similare (sau identice) .

Există și propoziții care se abat de la această regulă:
74/4|We have seventy-four chicken toppings, but we are told that we have to divide all by four. This being given, what is the left number?
78/37|There is pollo ad astra that has seventy-eight peperonata toppings. Peperonata toppings is divided by thirty-seven. How many toppings are left?

Rezultatul după procesarea acestor propoziții nu este similar:

We/PRP have/VBP seventy/NN -/: four/CD chicken/NNS toppings/NNS ,/, but/CC we/PRP are/VBP told/VBN that/IN we/PRP have/VBP to/TO divide/VB all/DT by/IN four/CD ./.. This/DT being/VBG given/VBN ,/, what/WP is/VBZ the/DT left/VBN number/NN ?/.

There/EX is/VBZ pollo/NN ad/NN astra/NN that/WDT has/VBZ seventy/NN -/: eight/CD peperonata/NNS toppings/NNS ./.. Peperonata/NNP toppings/NNS is/VBZ divided/VBN by/IN thirty/NN -/: seven/CD ./.. How/WRB many/JJ toppings/NNS are/VBP left/VBN ?/.

## Librăria Keras

Keras este un API scris în python care, în această aplicație, folosește tensorflow pentru a putea antrena mai ușor rețelele neuronale.

In mod normal python poate folosi un singur nucleu de pe procesor, tensorflow adaugă opțiunea de antrenare pe placa video, de asemenea acesta poate folosi toate nucleele de pe procesor accelerând enorm antrenarea.

Cu ajutorul acestei librarii am creat rețele neuronale de tip "dens" cu următoarea secvența de cod:

```
model = Sequential()
model.add(keras.layers.Dense(1024, input_dim=512, activation='tanh'))
model.add(keras.layers.Dense(8, activation='softmax'))
```

Aici vom avea o rețea neuronala cu 3 straturi:

- Primul strat va fi considerat stratul de intrare care va avea 512 noduri
- Al doilea strat va avea 1024
- Si ultimul strat 8

Toate cele 512 noduri vor fi conectate cu fiecare nod de pe stratul 2 cu ajutorul unor greutăți. Învățarea se va face pe aceste greutăți. Vor exista 1024 de asemenea valori pentru fiecare nod din primul strat.

Se va efectua un produs scalar apoi pe valoarea rezultată va fi aplicată funcția de activare aleasă , în cazul de fata TanH. Același lucru se aplică și pentru celelalte straturi.

Funcția soft-max este folosită în special pentru crearea clasificatorilor. Aceasta returnează un procentaj ( o valoare intre 1 și 0) care adunata cu celelalte valori de pe stratul respectiv rezultă valoarea 1(100%) .

Apoi compilăm modelul:

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])
```

De obicei optimizatorul ales pentru rețelele neuronale este Stochastic gradient descent, dar, dupa cum se poate observa in *Figura 3*, acesta se apropie greu de soluție comparativ cu ADAM. Acesta a fost construit pentru rețele neuronale de tipul "deep", dar, datorită faptului că fiecare greutate are propria rată de învățare , acesta se apropie de soluție mult mai rapid.

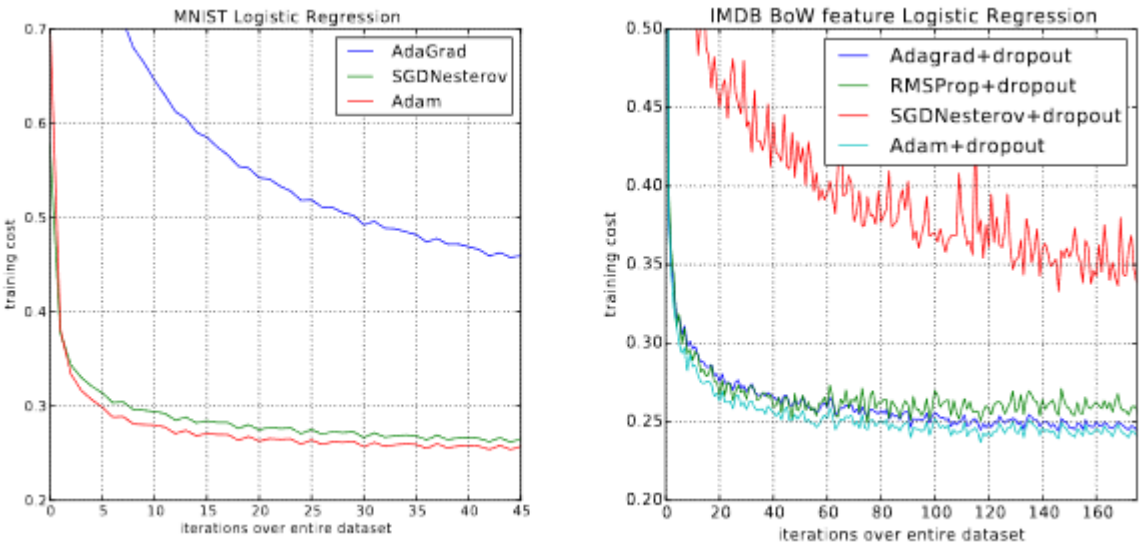



**Figura 3**: Comparația dintre ADAM si alți optimizatori. [6]

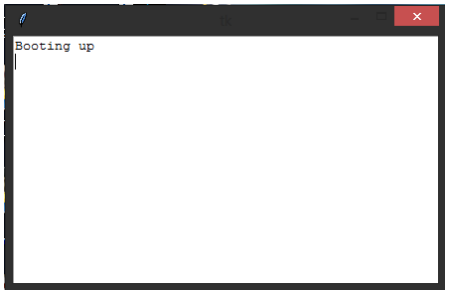
## Librăria Tkinter

Modulul tkinter este librăria standard de interfață grafică din python. Acesta a fost inițial creat pentru python 2 și apoi transferat în python 3.

Cu ajutorul acestei librarii am creat o interfață grafică simplă pentru o interacționare mai ușoară.

Secvența de cod următoare creează o fereastră de tipul Text box:

```
root = tkinter.Tk()
text_box = tkinter.Text(root, width=100, height=40)
text_box.grid(row=1, column=0, columnspan=2)
text_box.insert("end-1c", "Booting up\n")
```




De fiecare dată când se adaugă text , trebuie apelată metoda *root.update()*.

Un dezavantaj al acestei metode este faptul că între apelurile acestei funcții fereastra poate da impresia că s-a oprit din funcționat. Acest lucru se întâmpla din cauza funcției *root.mainloop()*,apelul acestea fiind blocant.

După ce s-a terminat rularea programului se poate naviga prin rezultatele produse cu ajutorul mausului.Am ales această librărie datorită simplității deoarece nu am avut nevoie de o interfață grafică foarte complexă.

# Capiloul II - Structura programului:

Aplicația este alcătuită , în mare parte, din 2 algoritmi. Primul este algoritmul de clusterizare care v-a împărți propozițiile din datele de antrenament în 8 fișiere diferite și o rețea neuronală care folosește aceste date la antrenament. Pentru prima parte a fost folosită librăria scikit-learn (mai târziu și nltk) și pentru partea a 2-a keras și tensorflow.

Împărțirea propozițiilor în 8 fișiere este esențială pentru ca rețeaua neuronală să poată să distingă propozițiile intre ele.

Algoritmul de clusterizare ales a fost k-means. Alegerea a fost făcută din mai multe motive:

- Algoritmul k-means are opțiunea de a alege numărul de centroizi finali și inițiali doriți.
- Un arbore id3 nu a aduce un nivel suficient de informație.
- Este un algoritm ușor de înțeles. De asemenea a fost predat în facultate la învățare automata în anul 3 semestrul 2.
- Raportul intre calitatea rezultatelor obținute și timp este cel puțin acceptabil.

Totuși k-means din sci-kit are și câteva dezavantaje:

- Poate folosi doar distanta Euclidiană. Folosirea unei alte distanțe poate avea efecte negative asupra calității soluției generate.
- Când distantele euclidiene sunt fie prea mari, fie prea mici, propozițiile vor fi puse în același cluster.(Acest lucru va fi exemplificat mai târziu)

Pentru a doua parte am ales să folosesc o rețea neuronală datorită structurii variabile a propozițiilor pentru fiecare cluster, și pentru a evita hard-codarea termenilor specifici (Ex : divide, add etc.). Deoarece, în acest caz, datele de antrenament pot fi schimbate cu altele pentru a obține un alt tip de clasificator.

## Datele de antrenament:

Numărul de propoziții: 5552 (cu un semn de operație) și 6046 ( cu doua semne de operație).

Câteva exemple de propoziții cu un semn de operație:

- 88+84|Dumbledore has eighty-eight magic animal and eighty-four cats. What is the total now?
- 43+62|Cajun spice topping has forty-three pizza toppings and sixty-two olive toppings. How many do we have now?
- 14-77|From fourteen old broomsticks someone decides to remove seventy-seven. This being given, what is the left number?

- 22-12|There is Hagrid that has twenty-two wands, but twelve of them disappear. This being given, what is the left number?
- 51*1|There is Ron that has fifty-one hate potions. If we add one times same amount of hate potions, how many do we have now?
- 83*2|Being given eighty-three sleeping potions, someone decides to multiply this by eighty-three. How many do we have now?

Câteva exemple de propoziții cu 2 semne de operație:

- 17+76+10|Hermione has seventeen weak wands and seventy-six wands. To those weak wands, we add ten more. How many do we have now?
- 69+82+59|Dumbledore has sixty-nine performant broomsticks and eighty-two broomsticks. Fifty-nine of them are added. How many do we have now?
- 52-7-46|From fifty-two parma ham toppings someone decides to remove seven. Somehow, forty-six are gone too. This being given, what is the left number?
- 60-35-30|From sixty pizzas someone decides to remove thirty-five. Somehow, thirty are gone too. This being given, what is the left number?
- 35*33*16|There is chicken topping that has thirty-five mushrooms. Thirty-three times same amount of mushrooms is added. After, we multiply all by sixteen. How many do we have now?
- 33*32*75|Being given thirty-three pine kernel, someone decides to multiply this by thirty-three. If we add seventy-five times the same amount what is the total now?
- 51/44/39|We have fifty-one deep pan bases, but we are told that we have to divide all by forty-four. Finally, we need to divide all this by thirty-nine. How many bases are left?
- 98/33/97|There is soho that has ninety-eight thin and crispy bases. This is divided by thirty-three. It is decided to share all to ninety-seven friends. This being given, what is the left number?

Etichetele de la începutul propozițiilor reprezintă expresia matematică pe care s-a bazat generarea propozițiilor. Ea nu v-a fi folosită la clusterizare sau în rețeaua neuronala, ci la verificarea performanței algoritmilor

Primul pas a fost de a transforma propozițiile în valori numerice. Pentru realizarea acestui lucru am folosit ,prima dată, vectorizatorii tf-idf și hashing.

## Vectorizatorul TF-IDF:

Subprogramul corespunzător:

```
Def tfid_clustering(lines,clusters):

   # Folosim TF-IDF pentru a transforma propozitiile în cuvinte
   vectorizer = TfidfVectorizer(max_df=0.3, max_features=None, min_df=0.05,
stop_words='english', use_idf=True)

   X = vectorizer.fit_transform(lines)
   #alegem algoritmul și parametrii specifici
   alg = MiniBatchKMeans(n_clusters=clusters, init='k-means++', n_init=10, init_size=500,
batch_size=2500)

   #clusterizarea
   alg.fit(X)

   return alg.labels_
```

Vectorizatorul TF-IDF folosește o tehnica bazată pe modelul "bag-of-words".Acesta returnează mai multe matrici rare care conțin trăsăturile propoziției respective.

Modelul bag-of-words este o reprezentare simplificată folosită în procesarea limbajului natural pentru extragerea informațiilor dintr-un text. Prima data textul este transformat doar în cuvinte. Apoi se calculează anumite masuri pentru a caracteriza textul. Cea mai folosită măsura este numărul de apariții.

**Semnificația parametrilor pentru vectorizator:**

- Max_df: Dacă exista cuvinte care au o frecvență mai mare decât max_df ele vor fi ignorate.
- Max_features: numărul maxim de trăsături
- Min_df: Dacă exista cuvinte care au o frecvență mai mică decât min_df ele vor fi ignorate.
- Stop_words: Cuvintele folosite des din vocabularul selectat (engleza pentru exemplul de mai sus (the,in etc.) ) vor fi eliminate primele.
- Use_idf: Rația intre greutatea la clusterizare și numărul de apariții va fi invers proporțională (daca atributul este TRUE)

**Semnificația parametrilor pentru algoritmul k-means cu metoda minibatch:**

- n_clusters: Numărul de centrioli inițiali și finali.
- Init: Metoda de inițializare
- N_init: Numărul de inițializări făcute la întâmplare la început. Doar inițializarea cea mai bună va fi folosită în continuare
- Init_size: Un subset de date ,alese la întâmplare, folosit pentru a grăbi inițializarea
- Batch_size: Dimensiunea unui minibatch

**Evaluarea antrenării Pentru TF-IDF CU MINIBATCH:**

Timp total de execuție: 11.45 + 12.82 + 33.96=58.23 secunde.

Timp total execuție cu minibatch: 0.98 secunde.

**Clusterizarea pe 4 clustere cu propoziții cu 1 semn de operație:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| Propoziții cu + | 269 | 134 | 437 | 137 |
| Propoziții cu - | 0 | 395 | 1118 | 0 |
| Propoziții cu * | 0 | 382 | 1131 | 0 |
| Propoziții cu / | 858 | 0 | 127 | 528 |

**Tabelul 1**

In *tabelul 1* avem doar doua clustere în care exista o majoritate ( 1, 4). Ambele conțin aceleași tipuri de propoziții și același semn majoritar.

Exemple de propoziții care aparțin clusterului 1:

- There is four seasons that has forty-eight pizza toppings. Pizza toppings is divided by eighty-two. This being given, what is the left number?
- Siciliana has fifty-three pine kernel and thirty rocket toppings. How many do we have now?
- We have sixty-six olive toppings, but we are told that we have to divide all by ninety-two. This being given, what is the left number?

In exemplele de mai sus cuvintele there , is ,that , has , but the, of vor fi ignorate deoarece au ponderea mai mare decât procentajul maxim (30%). Numeralele și substantivele nominale vor fi ignorate deoarece au ponderea sub 5%. Pe de alta parte câteva substantive comune și câteva verbe nu vor fi ignorate deoarece se încadrează intre 5% și 30%. în clusterul 1 avem cuvântul toppings care

aparține fiecărei propoziții din acest cluster. Cuvântul toppings apare în aproximativ 1500 de propoziții din totalul de 5552, procentajul de apariții fiind sub 30%.

In concluzie clusterizarea în clusterul 1 a fost făcută în jurul acestui substantiv. Același lucru s-a întâmplat și în cazurile celelalte. Pentru clusterul 4 cuvântul a fost bases , pentru 2 potions , pentru 3 wands.

Clusterul 2 este singurul care conține propoziții fără substantivul cheie ( potions). Aici au fost asignate și exemplele care nu conțin nici un cuvânt specific ca mai sus.

**Clusterizarea pe 4 clustere cu propoziții cu 2 semne de operație:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| Propoziții cu ++ | 0 | 1472 | 51 | 0 |
| Propoziții cu -- | 1503 | 7 | 0 | 0 |
| Propoziții cu ** | 0 | 0 | 1510 | 0 |
| Propoziții cu // | 1 | 6 | 2 | 1504 |

**Tabelul 2**

Același fenomen apare și aici (*tabelul 2*) dar de data aceasta fiecare tip de propoziție are unul sau mai mulți termeni specific:

- ++ -> add ,added
- -- -> remove
- ** -> multiply (sau add+times)
- // -> divide,divided

Numărul de apariții al oricărui dintre aceștia se încadrează intre 30% și 5%.

Aici toate tipurile de propoziție au două forme:

- 17+76+10|Hermione has seventeen weak wands and seventy-six wands. To those weak wands, we add ten more. How many do we have now?
- 67+44+27|Hermione has sixty-seven phoenix wands and forty-four dragon wands. Twenty-seven of them are added. What is the total now?
- 35*33*16|There is chicken topping that has thirty-five mushrooms. Thirty-three times same amount of mushrooms is added. After, we multiply all by sixteen. How many do we have now?
- 33*32*75|Being given thirty-three pine kernel, someone decides to multiply this by thirty-three. If we add seventy-five times the same amount what is the total now?

- 49/99/48|We have forty-nine deep pan bases, but we are told that we have to divide all by ninety-nine. It is decided to share all to forty-eight friends. This being given, what is the left number?
- 6/66/12|There is hot green pepper topping that has six herb spice toppings. This is divided by sixty-six. Finally, we need to divide all this by twelve. How many ingredients are left?
- 52-7-46|From fifty-two parma ham toppings someone decides to remove seven. Somehow, forty-six are gone too. This being given, what is the left number?
- 31-57-80|There is rosa that has thirty-one meat toppings, but fifty-seven of them disappear. From that amount, eighty are removed too. How many toppings are left?

Clusterizarea este aproape perfectă datorită structurii și termenilor specificați mai sus.
In clusterul 3 apar și 50 de exemple cu ++ datorită termenului add.

**Clusterizarea pe 8 clustere cu ambele tipuri de propoziție:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 | Cluster 7 | Cluster 8 |
|---|---|---|---|---|---|---|---|---|
| Propoziții cu + | 0 | 0 | 0 | 168 | 134 | 134 | 141 | 404 |
| Propoziții cu - | 550 | 0 | 0 | 555 | 191 | 188 | 0 | 28 |
| Propoziții cu * | 0 | 734 | 0 | 2 | 192 | 195 | 0 | 390 |
| Propoziții cu / | 0 | 0 | 598 | 528 | 0 | 0 | 382 | 5 |
| Propoziții cu ++ | 0 | 0 | 0 | 10 | 306 | 335 | 41 | 820 |
| Propoziții cu -- | 650 | 0 | 0 | 609 | 0 | 0 | 250 | 1 |
| Propoziții cu ** | 0 | 1008 | 0 | 0 | 0 | 0 | 272 | 230 |
| Propoziții cu // | 0 | 0 | 1029 | 314 | 0 | 0 | 167 | 2 |

**Tabelul 3**

Performanța este similară cu și fără inverse document frequency.

Din nou se întâmpla același fenomen ca mai sus. Clusterizarea în clustere 1 ,2 ,3 s-a făcută după termenii remove,multiply și divided. Deoarece acești termeni erau în limita acceptată mai sus , când combinăm toate propozițiile și creștem numărul total de propoziție de la 5552 și respectiv 6046 la 11598 ele rămân în această limita.

In clusterul 5 termenul ales a fost wands. Totuși, exista 85 de propoziții în clusterul 1 care conțin acest termen, dacă parametrul use_idf este setat ca False atunci cuvântul cu numărul de apariții mai mare îi va corespunde un număr mai mare.

Cuvintele wands și remove au un număr de apariții similar:2231 pentru remove și 2188 pentru wands. Deci valoarea asignată lui remove este puțin mai mare decât wands și de aceea propozițiile au fost puse în clusterul 1. Singurele propoziție care conțin cuvântul remove și apar în clusterul 5 sunt cele care conțin și termenul wands de doua ori.

Exemple din clusterul 5(Vezi tabelul 3):

- From seventy-seven phoenix wands someone decides to remove eighteen. How many wands are left?
- From fifty-one wands someone decides to remove fifteen. How many wands are left?

Exemple din clusterul 1(Vezi tabelul 3):

- From sixty-two dragon wands someone decides to remove ninety-eight. This being given, what is the left number?
- From thirty-two wands someone decides to remove forty-eight. This being given, what is the left number?

Clusterul 6(Vezi tabelul 3): conține toate propozițiile în care , cuvântul potions apare de 2 ori. Același lucru se întâmplă și în clusterul 7(Vezi tabelul 3): cuvântul de data asta fiind bases. Clusterele 8 și 4 nu au nici un termen specific.

**Parametri folosiți**: max_df=0.3, min_df=0.05, stop_words=English, use_idf=False, n_init=30, max_iter=2000.

Am decis ca frecvența maximă să fie de 30% pentru elimina o parte mai mare din cuvintele care nu ajută la clusterizare.

**Concluzie:**

Vectorizatorul TF-IDF se comportă cel mai bine la detectarea domeniilor din care fac parte o mulțime de documente datorită modelului "bag-of-words".

## Vectorizatorul HASHING:

Vectorizatorul HASHING transformă o colecție de documente/propoziții într-o matrice rară care conține caracteristicile fiecărei intrări.

Vectorizatorul HASHING are mai multe avantaje peste TF-IDF:

- Folosește mai putină memorie.
- Ordinea cuvintelor contează.

Singurul dezavantaj care afectează programul este faptul că acesta nu poate inversa ponderea dintre numărul de apariții și valoarea reprezentativă (Inverse Document-Frequency la tf-idf).
Funcția de hașurare folosită este Murmurhash3(versiunea pe 32 de biți).

Subprogramul corespunzător:

```
def hashing_clustering(lines, clusters):

    # Using vectorizer to turn strings into numerical values
    vectorizer = HashingVectorizer(stop_words='english,norm='l1')

    X = vectorizer.fit_transform(lines)

    # k-means almgorithm
    alg = KMeans(n_clusters=clusters, init='k-means++', n_init=20, max_iter=1000,
random_state=1)

    # actual clustering
    alg.fit(X)

    return alg.labels_
```

**Semnificația parametrilor pentru vectorizator:**

- Norm: algoritmul de normalizare , poate fi l1 ,l2 sau none
- N_features: în hashing matricele se populează pe loc , n_feature reprezentând numărul de coloane din matricea rară. Un număr mic poate duce la crearea coliziunilor.

Încă o diferență intre HASHING și TF-IDF ar fi că acesta cere numărul de coloane din matricea rară generată. Acest lucru ajută foarte mult la controlarea coliziunilor deoarece un număr mare de coloane rezultă un număr mai mic de coliziuni. Totuși timpul de rulare poate crește. Am ales n_features ca 2^20 datorită numărului mare de propoziție.

Din ce se poate observa mai jos Clusterizarea pe 4 clustere cu minibatch care folosește normalizarea l1 are rezultate mai slabe decât l2. Dar clusterizarea pe 8 clustere este neschimbată. Timpul de rulare a crescut pentru ca HASHING este puțin mai lent decât TF-IDF. Una dintre probleme întâlnite la algoritmul TF-IDF cu parametrii folosiți a fost faptul că rezultatele erau puțin diferite de la o rulare la alta și datorită acestei inițializări, am decis să măresc numărul acest număr la 10. Bineînțeles, acest lucru a avut un impact negativ la timpul de rulare, mai ales pentru k-means fără minibatch.

Pentru a vizualiza mai ușor comportamentul algoritmului prima dată am încercat clusterizarea pe 4 clustere cu cele 2 tipuri de propoziții separate. Majoritatea deciziilor de optimizare sau rafinare au avut ca prioritate clusterizarea pe 4. Dacă nu exista o diferență clara intre propoziții cu același număr de semne de operație atunci va fi mult mai greu să diferențiem propozițiile când au număr diferit de semne de operație.

**Evaluarea antrenării Pentru HASHING cu MINIBATCH:**

Timp total de execuție: 34.76 + 38.93 + 90.61=164.3 secunde

Timp total de execuție cu minibatch: 2.79 + 2.47 + 4.52=9.78 secunde

**Clusterizarea pe 4 clustere cu propoziții cu 1 semn de operație și L2:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| Propoziții cu + | 573 | 169 | 0 | 272 |
| Propoziții cu - | 358 | 0 | 1154 | 523 |
| Propoziții cu * | 780 | 0 | 733 | 114 |
| Propoziții cu / | 69 | 523 | 0 | 807 |

**Tabelul 4**

**Clusterizarea pe 4 clustere cu propoziții cu 1 semn de operație și L1:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| Propoziții cu + | 166 | 0 | 275 | 573 |
| Propoziții cu - | 0 | 1157 | 0 | 355 |
| Propoziții cu * | 0 | 734 | 0 | 779 |
| Propoziții cu / | 523 | 114 | 854 | 22 |

**Tabelul 5**

Clusterizarea cu l1 (*tabelul 5*) rezultată este puțin mai bună deoarece exista majoritate în 2 clustere fată de clusterizarea cu l2(*tabelul 4*) în care exista doar un singur cluster cu majoritate. Procentajul de omogenitate cu l1 din clusterul 1 este de ~70% iar cel cu l2 din clusterul 2 este tot de 70%.

Datorita faptului că algoritmul creează o configurație de început la întâmplare , rezultatul nu este exact la fel de fiecare dată.

Exemple din clusterul 1(*tabelul 5*):

- We have seventy-seven thin and crispy bases, but we are told that we have to divide all by eighty-four. How many bases are left?
- We have forty-eight pizza bases, but we are told that we have to divide all by sixty-nine. This being given, what is the left number?
- Giardiniera has sixty-one nut toppings and forty pizza bases. What is the total now?

Exemple din clusterul 3(*tabelul 5*):

- There is pollo ad astra that has twenty-nine peperonata toppings. Peperonata toppings is divided by seventy-three. How many ingredients are left?
- We have seventy-four chicken toppings, but we are told that we have to divide all by four. This being given, what is the left number?
- Fiorentina has eighty-six pizza toppings and fifty chicken toppings. How many do we have now?

Datorită structurii similare a propozițiilor și faptul cp ambii algoritmi lucrează cu fervența termenilor avem o concluzie similară ca și la tf-idf. Toate exemplele din clusterul 1 conțin cuvântul bases cel puțin de 2 ori în fiecare propoziție, și pentru clusterul 3 95% dintre acestea conțin termenul toppings ,din nou, cel puțin de 2 ori. De asemenea termenul ales pentru clusterul 2 a fost decides.

Pentru 1 și 3 algoritmul a preferat să aleagă toppings și bases datorită numărul de apariții și faptul ca apar de doua ori într-o singura propoziție. Toppings apare de 1863 de ori, bases apare de 1353 de ori fata de divided 763 ,divide 740,multiply 734,remove 772,disappear 729,add 769 și decides 1503.În clusterul 4 se află toate propozițiile care nu respectă nici o regulă de mai sus.

Nu există o soluție  datorită faptului că nu putem modifica propozițiile și nu folosim IDF. Dar după cum s-a observat mai sus chiar dacă am folosi IDF algoritmul ar găsi alți termeni.

Am folosit normalizarea l2 în continuare.

**Clusterizarea pe 4 clustere propoziții cu 2 semne de operație:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| Propoziții cu + | 0 | 1404 | 0 | 109 |
| Propoziții cu - | 1508 | 2 | 0 | 0 |
| Propoziții cu * | 0 | 1 | 0 | 1509 |
| Propoziții cu / | 1 | 7 | 1503 | 2 |

**Tabelul 6**

Pentru propozitiile cu 2 (*tabelul 6*) semne de operatie multiply apare de 1488 ori,add 1562, added 1452,disappear 752,removed 710,remove 749,mai apare și termenul gone 794 ,share 752,si restul termenilor menționați mai sus.

In clusterul 1(*tabelul 6*) exemplele conțin remove și removed sau remove și gone, acești termeni având o valoare similară, în acelasi timp:

- From ninety-seven sweet pepper toppings someone decides to remove seventy-seven. From that amount, ninety-eight are removed too. This being given, what is the left number?
- From twenty-three cheese toppings someone decides to remove forty. From that amount, fifty-four are removed too. How many toppings are left?
- From seventy-six herb spice toppings someone decides to remove eighty-six. Somehow, eighty-seven are gone too. How many ingredients are left?

La prima vedere există iluzia că algoritmul a ignorat cuvintele care au fost folosite pentru clusterizarea anterioară ( toppings, bases) chiar dacă aceștia apar de 3393 respectiv 2317 din totalul de 6046 de propoziții fată de 5552. Datorită faptului că de data aceasta propozițiile sunt mai lungi termenii caracteristici (remove,removed,add etc) pot apărea de două ori. Acest lucru ajută foarte mult la egalizarea valorilor.

Pentru exemplele următoare din clusterul 4(*tabelul 6*):

- Being given ninety-eight peperoni sausage toppings, someone decides to multiply this by ninety-eight. If we add twenty-eight times the same amount what is the total now?

Simpla apariție a cuvântului toppings nu este suficientă la selectarea clusterului final.
Pentru clusterul 2(*tabelul 6*) avem potion(potion apare doar de 843 de ori în total), add, added:

- Hermione has thirteen healing potions and forty-seven potions. Forty-three of them are added. How many do we have now?
- Wizard has fourteen weak wands and sixty-six phoenix wands. To those weak wands, we add fifty-three more. What is the total now?

Și pentru clusterul 3 avem divide(care poate apărea de mai multe ori într-o propoziție):

- We have ninety-nine tomato toppings, but we are told that we have to divide all by seventy-six. Finally, we need to divide all this by eighty-one. How many toppings are left?

**Clusterizarea pe 8 clustere cu ambele tipuri de propoziții:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 | Cluster 7 | Cluster 8 |
|---|---|---|---|---|---|---|---|---|
| Propoziții cu + | 0 | 134 | 305 | 134 | 0 | 272 | 169 | 0 |
| Propoziții cu - | 0 | 191 | 80 | 189 | 0 | 0 | 0 | 1052 |
| Propoziții cu * | 0 | 192 | 392 | 195 | 734 | 0 | 0 | 0 |
| Propoziții cu / | 491 | 0 | 12 | 0 | 0 | 564 | 385 | 61 |
| Propoziții cu ++ | 0 | 306 | 761 | 335 | 0 | 71 | 39 | 0 |
| Propoziții cu -- | 0 | 0 | 2 | 0 | 0 | 441 | 233 | 833 |
| Propoziții cu ** | 0 | 0 | 125 | 0 | 766 | 347 | 272 | 0 |
| Propoziții cu // | 870 | 0 | 10 | 0 | 0 | 355 | 202 | 75 |

**Tabelul 7**

Pentru clusterul 1(*tabelul 7*) avem divide(1504 + 2257) și toppings(3393+1863). Cuvântul toppings aici apare într-o măsură mai mică sau egală cu divide:

- We have seventy-nine deep pan bases, but we are told that we have to divide all by twenty. This being given, what is the left number?
- There is ice cream that has sixty-one rosemary toppings. This is divided by fifty. Finally, we need to divide all this by twenty-three. How many ingredients are left?

Pentru clusterul 2(*tabelul 7*) există doar cuvântul wands de cel puțin 2 ori în fiecare propoziție:

- Draco has twenty-eight powerful wands and eighty-one dragon wands. Sixty-eight of them are added. How many do we have now?
- There is Hermione that has fifty-two weak wands, but sixty-five of them disappear. How many wands are left?

Același lucru și în clusterul 4(*tabelul 7*) de data asta peste cuvântul potions:

- There is Hermione that has three love potions. If we add fifty-eight times same amount of love potions, how many do we have now?

Clusterul 5(*tabelul 7*) are termenii multiply, someone, decides:

- Being given sixteen white owls, someone decides to multiply this by sixteen. What is the total now?
- Being given three asparagus toppings, someone decides to multiply this by three. After, we multiply all by eighty-five. How many do we have now?

Clusterul 6(*tabelul 7*) are toppings și remove. Aici apar doar exemplele în care numărul de apariții al cuvântului toppings este mai mare decât celorlalți termeni specifici:

- Fiorentina has ninety-two nut toppings and thirteen sliced tomato toppings. What is the total now?
- There is american hot that has twenty red onion toppings. Red onion toppings is divided by forty-one. This being given, what is the left number

Clusterul 7(*tabelul 7*) are propoziții cu termenul bases cu 2-3 apariții:

- There is cajun that has eighty-one pizza bases. Pizza bases is divided by eight. How many bases are left?
- Veneziana has forty-two deep pan bases and eighteen thin and crispy bases. How many do we have now?

Clusterul 3 conține propozițiile care se abat de la orice regulă din celelalte clustere.

**Concluziile de până acum:**

- HASHING are o performanță mult mai bună clusterizarea pe 4 clustere cu propoziții cu 1 semn de operație (Exista o majoritate de cel puțin 60% în fiecare cluster la TF-IDF nu exista nici un cluster cu o asemenea majoritate)
- Nu există nici o majoritate în ultima clusterizare.
- Propozițiile cu 1 semn de operație nu sunt suficient de distincte fată de cele cu 2 semne de operație pentru că acești vectorizatori să le poate distinge intre ele.
- Ambii vectorizatori folosesc doar cuvintele din propoziții creând o matrice rară pe baza acestora.

**Alte mențiuni:**

- Am încercat să folosesc și affinity clustering dar toți vectorizatorii returnează o matrice rară și affinity clustering are nevoie de o matrice densă. De asemenea affinity clustering nu lucrează pe un număr de centroizi prestabiliți.
- SCI-KIT mai oferă un vectorizator numit COUNT-VECTORIZER , dar acesta returnează o matrice de token_counts , lucru care nu poate fi folosit la clusterizare.

**Cum ar trebui să arate algoritmul:**

- Trebuie să existe o diferență clară între propozițiile cu 2 semne de operație și cele cu un singur semn de operație
- Verbele care au o semnificație similară (remove/disappear) trebuie să aibă o greutate similară chiar dacă sunt diferite.
- Poziția numeralelor are o importanță relativ mare.
- Semnificația celorlalte cuvinte nu este foarte importantă (De ex: o propoziție poate folosi numele Hagrid iar cealaltă să folosească pizza, ambele sunt substantive).În schimb poziția lor în propoziție contează.

## Algoritmul de clusterziare final:

Pentru a rezolva una dintre problemele de mai sus putem să folosim partea de vorbire în loc de cuvintele în sine. Astfel nu va conta forma substantivului deoarece toate substantivele vor avea o singura etichetă. Putem folosi librăria NLTK pentru acest lucru.

O problemă care a apărut a fost faptul că numeralele care nu sunt în formă numerica nu vor fi etichetate ca numerale în NLTK (Ex: sixty-two va fi etichetat ca adjective).Deci trebuie să pre-procesăm propozițiile și să înlocuim aceste numerale cu forma lor numerică (sixty-two devine 62).

Apoi trebuie să asignăm fiecărei părți de propoziție o valoare numerică specifică pentru a putea folosi datele la clusterizare. Dicționarul folosit are următoarea formă:

dict = {"CC": 0.01, "CD": 0.02, "DT": 0.03, "EX": 0.04, "FW": 0.05, "IN": 0.06, "JJ": 0.13, "JJR": 0.13, "JJS": 0.13, "LS": 0.10, "MD": 0.11, "NN": 0.12, "NNS": 0.12, "NNP": 0.12, "NNPS": 0.12, "PDT": 0.16, "POS": 0.17, "PRP": 0.18, "PRP$": 0.19, "RB": 0.20, "RBR": 0.21, "RBS": 0.22, "RP": 0.23,"SYM": 0.24, "TO": 0.25, "UH": 0.26, "VB": 0.27, "VBD": 0.27, "VBG": 0.27, "VBN": 0.27, ",": 0, "VBZ": 0.27, ".": 0, "WDT": 0.27, "WRB": 0.27, "WP": 0.27, "VBP": 0.50}

NN , NNS ,NNP ,NNPS au aceeași valoare pentru că toate reprezintă substantive și nu este relevant dacă acesta este nominal etc. VB,VBD,VBG,VBN,VBZ,WDT,WRB,WP,VBP reprezintă verbele , JJ reprezintă adjectivele și CD reprezintă numeralele. Am ales ca adjectivele și substantivele să aibă valori similare datorită grupurilor substantiv-adjectiv sau adjectiv-substantiv sau chiar adjectiv-adjectiv-substantiv (câteodată și substantiv-substantiv, substantiv-substantiv-adjectiv). Simbolul „ . ” face referință la semnele de punctuație (! ? . ,).

Aceste grupuri reprezintă subiectul propoziției sau obiectul care este folosit în propoziție:

- 6*2|Being given six performant broomsticks, someone decides to multiply this by six. How many do we have now?
- 81/43|There is unclosed pizza that has eighty-one cajun spice toppings. Cajun spice toppings is divided by forty-three. This being given, what is the left number?
- 8/97|We have eight peperoni sausage toppings, but we are told that we have to divide all by ninety-seven. This being given, what is the left number?

Dacă aplicăm acest model pe propoziția următoare:

69-49|There is Hagrid that has sixty-nine old broomsticks, but forty-nine of them disappear. How many sticks are left?

După transformarea numeralelor vom avea:

There is Hagrid that has 69 old broomsticks, but 49 of them disappear. How many sticks are left?

După tokenizarea NLTK:

EX VBZ NNP WDT VBZ CD JJ NNS . CC CD în PRP VBP . WRB JJ NNS VBP VBN .

În lista finală înainte de valorile fiecărei părți de propoziție am decis să fie inclusă și dimensiunea cuvântului pentru distingerea a 2 cuvinte diferite.

Lista finală:

0.04 5 0.27 2 .012 6 0.27 4 0.27 3 0.02 2 0.13 3 0.12 11 0 1.0 0.10 3 0.2 2 0.6 2 0.18 4 0.50 9 ….

Știm ca algoritmul k-means (implementat în librăria SCI-KIT learn) folosește distanta euclidiana deci poziția elementelor în vector este foarte importantă. Acesta este un avantaj pe care îl putem folosi și reprezintă încă un motiv pentru care am ales valorile de mai sus pentru părțile de propoziție. Nu putem folosi o altă formula pentru distanta vectoriala (Manhattan, Minkowski etc.) datorită implementării algoritmului .

Folosim algoritmul k-means cu următorii parametrii:

*km = KMeans(n_clusters=true_k, init='k-means++', n_init=30, max_iter=2000, random_state=1)*

Dacă random_state nu este setat la 1 algoritmul nu va fi determinist datorită inițierii aleatorie a centroizilor inițiali. După cum o să se vadă mai jos de data aceasta nu putem folosi 10 stadii inițiale cum am folosit la HASHING datorită timpului crescut de rulare. Nu se folosește mini-batch pentru că se dorește maximizarea performației algoritmului. Un timp de rulare în jur de 60 de secunde nu ar fi o problemă majora. De obicei algoritmul termină clusterizarea până la iterația 20.

**Analiza Performantei:**

Timp total de execuție: 15.05+20.26+40.83=76.14 secunde

Un timp mai bun decât HASHING și în același timp determinist.

**Clusterizarea pe 4 clustere cu propoziții cu 1 semn de operație:**

Timp de execuție: 15.05583 secunde

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| propoziții cu + | 26 | 2 | 983 | 3 |
| propoziții cu - | 0 | 551 | 837 | 124 |
| propoziții cu * | 609 | 576 | 132 | 196 |
| propoziții cu / | 34 | 189 | 16 | 1274 |

**Tabelul 8**

Clusterul 1(*tabelul 8*):

- Being given sixty-five dragon wands, someone decides to multiply this by sixty-five. What is the total now?
- Being given ninety-three performant broomsticks, someone decides to multiply this by ninety-three. How many do we have now?

Faptul că 609 din 670 de exemple din clusterul 1 încep cu "Being given" și de asemenea faptul că exact aceleași propoziții au semnul înmulțire se datorează generării neuniforme. Aici 90% din propoziții au între 18 și 22 de cuvinte.

Pentru clusterul 2(*tabelul 8*):

- There is Harry Potter that has twenty-eight wands. If we add twenty-three times same amount of wands, what is the total now?
- There is Hermione that has sixty-six love potions, but sixty-nine of them disappear. This being given, what is the left number?

Există 1315 exemple care încep cu "There is" din 1318. Faptul că propozițiile de acest tip au o lungime similara se datorează din nou generării. Aici 90% din propoziții au intre 18 și 23 de cuvinte.

Pentru clusterul 3(*tabelul 8*):

- Ron has thirty-six broom sticks and forty-two performant broomsticks. How many do we have now?
- Being given ten wands, someone decides to multiply this by ten. How many do we have now?

In acest cluster exista și propoziții mai scurte (15-16 cuvinte),numărul de cuvinte în majoritatea exemplelor este intre 17 și 20.

Pentru C4(*tabelul 8*):

- We have eight peperoni sausage toppings, but we are told that we have to divide all by ninety-seven. This being given, what is the left number?

- There is rosemary topping that has forty-seven fish toppings. Fish toppings is divided by eighty-nine. This being given, what is the left number?

Se poate observa că datorită valorilor mici din dicționar și valorilor mari (in comparație) a lungimii unui cuvânt (0.11 3) clusterizarea s-a făcut după lungimea cuvintelor.

**Clusterizarea pe 4 clustere propoziții cu 2 semne de operație:**

Timp de execuție: 20.26006 secunde

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| propoziții cu ++ | 11 | 1133 | 1 | 367 |
| propoziții cu -- | 728 | 369 | 15 | 108 |
| propoziții cu ** | 714 | 372 | 18 | 406 |
| propoziții cu // | 686 | 2 | 825 | 0 |

**Tabelul 9**

Exact acelaşi lucru s-a întâmplat și aici (*tabelul 9*). Faptul ca în clusterul 3 exista în mare parte doar propoziții cu doua diviziuni se datorează structurii similare:

- We have seventy-two thin and crispy bases, but we are told that we have to divide all by fifty-eight. Finally, we need to divide all this by twenty-seven. This being given, what is the left number?

769 dintre aceste incepand cu “We have” , acesta fiind singurul cluster în care acest tip apare.

**Clusterizarea pe 8 clustere cu ambele tipuri de propoziții:**

Timp de execuție: 40.833265 secunde

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 | Cluster 7 | Cluster8 |
|---|---|---|---|---|---|---|---|---|
| propoziții cu + | 0 | 11 | 0 | 1 | 0 | 2 | 1 | 999 |
| propoziții cu - | 85 | 0 | 0 | 50 | 0 | 0 | 552 | 826 |
| propoziții cu * | 100 | 609 | 0 | 107 | 0 | 0 | 589 | 108 |
| propoziții cu / | 201 | 75 | 52 | 962 | 0 | 67 | 114 | 12 |
| propoziții cu ++ | 944 | 268 | 3 | 272 | 1 | 2 | 4 | 18 |
| propoziții cu -- | 199 | 2 | 313 | 514 | 0 | 394 | 80 | 8 |
| propoziții cu ** | 221 | 400 | 464 | 155 | 18 | 252 | 0 | 0 |
| propoziții cu // | 0 | 0 | 322 | 2 | 809 | 378 | 0 | 1 |

**Tabelul 10**

La fel ca și la clusterizarea 4 clustere propoziții cu 2 semne de operație propozițiile cu două împărțiri au o structură unică având un cluster separat.

Rezultatul cel mai bun a fost obținut după ce valorile dicționarului au fost înmulțite cu 1000. Pentru valori mai mari (ex:10000) distanța este prea mare ducând la clusterizarea tuturor propozițiilor într-un singur cluster. Pentru valori mai mici diferență nu este suficientă.

Încercăm să clusterizăm din nou:

**Clusterizarea pe 4 clustere cu 1 semn de operație:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| propoziții cu + | 97 | 4 | 912 | 0 |
| propoziții cu - | 1 | 731 | 781 | 0 |
| propoziții cu * | 738 | 770 | 5 | 0 |
| propoziții cu / | 9 | 776 | 9 | 719 |

**Tabelul 11**

Avem același număr de clustere în care exista o majoritate(2), pe exact aceleași semne. De data asta avem o puritate mai mare pentru o singura împărțire.

**Clusterizarea pe 4 clustere cu 2 semne de operație:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| propoziții cu ++ | 490 | 1004 | 19 | 0 |
| propoziții cu -- | 398 | 426 | 686 | 0 |
| propoziții cu ** | 739 | 7 | 764 | 0 |
| propoziții cu // | 5 | 2 | 743 | 763 |

**Tabelul 12**

Cea mai mare problemă vizibila este că propozițiile cu plus nu au un verb specific. Pentru clusterizarea pe 4 clustere cu 2 semne de operație(*tabelul 12*) avem din nou o creștere a omogenității pentru clusterul plus+plus și div+div.

Totuși încă nu s-a reușit departajarea concretă a propozițiilor.

In continuare vom adauga numărul de verbe și numărul de numerale la finalul listei generate. Acest număr va fi din nou înmulți cu 1000(100 nu departajează suficient propozițiile cu 2 semne de propoziție cu 1 semn iar 10000 este prea mult și afectează centroizii). Performanța optimă are loc atunci când numărul de amplificare la verbe este egal cu cel de la numerale. Aceste 2 valori trebuie adăugate la finalul sau la începutul listei pentru a afecta distanța euclidiană într-un mod favorabil.

**Analiza performantei după modificările de mai sus:**
**Clusterizarea pe 4 clustere cu 1 semn de operație:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| propoziții cu + | 4 | 0 | 966 | 44 |
| propoziții cu - | 1135 | 1 | 4 | 372 |
| propoziții cu * | 813 | 0 | 10 | 690 |
| propoziții cu / | 245 | 1257 | 7 | 4 |

**Tabelul 13**

Față de rezultatele de mai sus (*tabelul 8*) propozițiile care încep cu “Being given” nu mai apar doar intr-un singur cluster. Tipurile de propozitii care au majoritate sunt împartirea și adunarea fată de înmulțire și adunare.

În primul cluster avem urmatorul exemplu:

- There is Dumbledore that has sixty-four old rats, but sixty-three of them disappear. This being given, what is the left number?

Nu exista nici o propoziție în clusterul 4(*tabelul 13*) care să se termine cu "left number".

De asemenea clusterul 1(*tabelul 13*) conține exemplele cu multiply care au urmatoare formă:

- Being given sixty-one potions, someone decides to multiply this by sixty-one. How many do we have now?

Și clusterul 4(*tabelul 13*) :

- Being given thirty-one wands, someone decides to multiply this by thirty-one. What is the total now?

Toate exemplele cu multiply care se află în clusterul 1 se termină cu "how many do we have now" iar cele din clusterul 4 (*tabelul 13*) cu "what is the total now". Nu există exemplu care să se abată de la această regulă.

Același lucru și pentru "add":

- There is Ron that has forty-one potions. If we add sixty times same amount of potions, how many do we have now? (C1)
- There is Hermione that has eighty-eight dragon wands. If we add one times same amount of dragon wands, what is the total now? (C4)

În clusterul 3(*tabelul 13*) se află exemplele care încep cu un substantiv:

- Quattro formaggi has one pizza base and seventeen deep pan bases. What is the total now?
- Draco has ninety-five new broomsticks and seventy-four old broomsticks. How many do we have now?

Există foarte puține exemple (sub 5%) în celelalte clustere care să respecte structura de mai sus.

**Clusterizarea pe 4 clustere cu 2 semne de operație:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster |
|---|---|---|---|---|
| propoziții cu ++ | 0 | 1129 | 383 | 0 |
| propoziții cu -- | 1051 | 4 | 361 | 94 |
| propoziții cu ** | 458 | 9 | 1036 | 7 |
| propoziții cu // | 263 | 8 | 3 | 1239 |

**Tabelul 14**

De data aceasta există o majoritate de cel puțin 60% în fiecare cluster pentru a 2-clusterizare. Clusterul plus-plus și div-div având o omogenitate aproape perfectă.

Clusterul 2*(tabelul 14)* conține propozițiile care încep cu substantive:

- Wizard has twenty-eight love potions and fifteen sleeping potions. To those love potions, we add eighty-two more. What is the total now?

Toate exemplele generate pentru plus-plus au această formă.

Se poate observa faptul că fiecare tip de propoziție are ca majoritate o structura unică dar există anumite exemple care au fost generate cu structura de la un alt semn.

Exemplu pentru doua înmulțiri:

- There is cajun that has sixty-two giardinieras. Eight times same amount of giardinieras is added. After, we multiply all by fifty-nine. How many do we have now?
- Being given seventeen hot spiced beef toppings, someone decides to multiply this by seventeen. If we add seventy-three times the same amount how many do we have now?

Majoritatea propozițiilor care au doua înmulțiri și respecta structura exemplului 2 se află în clusterul 3 (756 astfel de exemple și doar 9 în clusterul 1) .

Al doilea exemplu este specific propozițiilor cu două scăderi:

- There is mushroom that has forty-seven red onion toppings, but sixty-five of them disappear. From that amount, forty-six are removed too. This being given, what is the left number?

De asemenea propozițiile cu două împărțiri:

- We have ten petit pois toppings, but we are told that we have to divide all by ninety. Finally, we need to divide all this by sixteen. This being given, what is the left number?
- We have sixty-six tobasco pepper sauces, but we are told that we have to divide all by twenty-two. It is decided to share all to thirty-one friends. This being given, what is the left number?

**Clusterizarea pe 8 clustere cu ambele tipuri de propoziții:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 | Cluster 7 | Cluster 8 |
|---|---|---|---|---|---|---|---|---|
| propoziții cu + | 4 | 0 | 43 | 0 | 966 | 0 | 1 | 0 |
| propoziții cu - | 725 | 0 | 372 | 0 | 4 | 1 | 0 | 410 |
| propoziții cu * | 741 | 0 | 690 | 0 | 10 | 0 | 0 | 72 |
| propoziții cu / | 2 | 29 | 4 | 29 | 7 | 1199 | 10 | 233 |
| propoziții cu ++ | 267 | 0 | 744 | 1 | 383 | 0 | 116 | 1 |
| propoziții cu -- | 2 | 94 | 4 | 1051 | 0 | 0 | 359 | 0 |
| propoziții cu ** | 0 | 7 | 5 | 91 | 4 | 367 | 813 | 223 |
| propoziții cu // | 1 | 1239 | 7 | 262 | 0 | 1 | 2 | 0 |

**Tabelul 14**

La clusterizarea finală avem o majoritate de 60% sau mai mult în 5 clustere. și avem 3 clustere în care nu există o majoritate, și 3 tipuri de propoziții care nu pot fi distinse între ele.

Pentru C1(*tabelul 14*) avem următoarele exemple:

- Being given ten wands, someone decides to multiply this by ten. How many do we have now?
- There is Draco that has sixty-five magic animal, but two of them disappear. How many animals are left?

Pentru C3(*tabelul 14*):

- Being given thirteen old broomsticks, someone decides to multiply this by thirteen. What is the total now?
- From twenty-eight dragon wands someone decides to remove seventy-five. How many wands are left?
- There is Harry Potter that has twenty-two performant broomsticks. If we add ninety-one times same amount of performant broomsticks, what is the total now?

Pentru C8(*tabelul 14*):

- There is frutti di mare that has fifty-two hot green pepper toppings. Seventy-five times same amount of hot green pepper toppings is added. If we add sixty-two times the same amount what is the total now?
- There is food that has ninety-seven foods. Ninety-two times same amount of foods is added. After, we multiply all by thirty-one. What is the total now?
- There is wizard that has sixty-seven broom sticks, but fifty-three of them disappear. This being given, what is the left number?

Datorită faptului că avem doar 8 clustere totale, peste 8 tipuri de propoziții și reprezentarea aleasă conține un număr insuficient de date (de exemplu propozițiile cu “disappear” sunt echivalent cu “gone too” dar verbele nu au un parametru care să exprime semnificația lor, de aceea aceste doua tipuri pot exista în clustere diferite chiar dacă ele au o semnificație similară) câteva exemple sunt confundate.

In continuare putem aplica încă o clusterizare de 8 peste clusterele care nu au o majoritate și să încercam să segmentăm propozițiile rezultate. După ce facem clusterizarea din nou vom încerca să extragem propozițiile din clusterele care au o majoritate și să le punem în cele formate inițial.

Dacă există clustere care nu au o majoritate confirmată după a doua clusterizare se poate încerca o clusterizare din nou sau să nu folosim aceste exemple în rezultatul final deoarece sunt prea similare cu alte propoziții.

Clusterizarea a 2-a v-a fi rulata pe 8 clustere. Am ales 8 pentru că, din documentația de la sci-kit, reiese faptul că acest algoritm lucrează cel mai bine cu 8 centroizi. Dacă am avea de exemplu 2 clustere am risca să nu avem majoritatea în nici un cluster, în același timp nu vrem să avem prea multe deoarece performanța k-means poate fi afectată negativ.

Procentajul de majoritate minim pe care l-am ales a fost de 60%

**După reclusterizarea clusterului 1:**

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 | Cluster 7 | Cluster 8 |
|---|---|---|---|---|---|---|---|---|
| propoziții cu + | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 |
| propoziții cu - | 0 | 0 | 263 | 305 | 102 | 0 | 55 | 0 |
| propoziții cu * | 357 | 321 | 0 | 1 | 0 | 0 | 0 | 62 |
| propoziții cu / | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| propoziții cu ++ | 0 | 2 | 0 | 96 | 0 | 0 | 168 | 1 |
| propoziții cu -- | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| propoziții cu ** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| propoziții cu // | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Tabelul 15**

Clusterizarea a avut efectul dorit.

Toate propozițiile din C1(*tabelul 15*) au următoarea formă:

- Being given sixty-one magic animal, someone decides to multiply this by sixty-one. How many do we have now?

Pentru restul clusterelor avem același lucru:

- There is Dumbledore that has eighty-eight old broomsticks. If we add seven times same amount of old broomsticks, how many do we have now?(C2 )
- From forty-three weak wands someone decides to remove nine. This being given, what is the left number? (C3)
- There is Hermione that has seventy-five magic potions, but forty of them disappear. How many potions are left? (C4)
- From thirty-nine cats someone decides to remove eighty. This being given, what is the left number?(C5)
- Hagrid has twelve magic animals and twenty-five old rats. Thirty-three of them are added. How many do we have now?(C7)
- There is Harry Potter that has fifty-four old rats. If we add twenty-one times same amount of old rats, how many do we have now?(C8)

Rezultatele din C1,C2,C8 vor fi adăugate clusterului final care conține majoritatea propozițiilor înmulțit. Rezultatele din C3,C4,C5,C7 vor fi adăugate clusterului final care conține majoritatea propozițiilor minus.

**După reclusterizarea clusterului 3 și 8:**

| C3 | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 | Cluster 7 | Cluster 8 |
|---|---|---|---|---|---|---|---|---|
| propoziții cu + | 0 | 0 | 3 | 0 | 38 | 2 | 0 | 0 |
| propoziții cu - | 371 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| propoziții cu * | 0 | 0 | 292 | 247 | 46 | 0 | 105 | 0 |
| propoziții cu / | 0 | 0 | 1 | 0 | 2 | 0 | 1 | 0 |
| propoziții cu ++ | 0 | 154 | 43 | 2 | 210 | 90 | 6 | 239 |
| propoziții cu -- | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| propoziții cu ** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 |
| propoziții cu // | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 4 |

**Tabelul 16**

| C8 | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 | Cluster 7 | Cluster 8 |
|---|---|---|---|---|---|---|---|---|
| propoziții cu + | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| propoziții cu - | 75 | 276 | 0 | 59 | 0 | 0 | 0 | 0 |
| propoziții cu * | 0 | 0 | 0 | 0 | 0 | 0 | 32 | 40 |
| propoziții cu / | 72 | 0 | 0 | 84 | 0 | 0 | 74 | 3 |
| propoziții cu ++ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| propoziții cu -- | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| propoziții cu ** | 0 | 0 | 0 | 0 | 114 | 107 | 2 | 0 |
| propoziții cu // | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Tabelul 17**

În *tabelul 17* rezultatele din clusterul 1 nu pot fi folosite deoarece nu exista majoritate.

**Numărul de propoziții care nu pot fi folosite**:291

**Rezultate finale:**

| | Fișier 1 | Fișier 2 | Fișier 3 | Fișier 4 | Fișier 5 | Fișier 6 | Fișier 7 | Fișier 8 |
|---|---|---|---|---|---|---|---|---|
| propoziții cu + | 965 | 3 | 0 | 4 | 40 | 0 | 0 | 1 |
| propoziții cu - | 4 | 1317 | 1 | 0 | 56 | 0 | 0 | 0 |
| propoziții cu * | 10 | 1 | 32 | 6 | 2 | 29 | 29 | 10 |
| propoziții cu / | 7 | 1 | 1273 | 1424 | 46 | 0 | 0 | 0 |
| propoziții cu ++ | 383 | 96 | 1 | 54 | 861 | 1 | 0 | 116 |
| propoziții cu -- | 0 | 1 | 0 | 0 | 5 | 1051 | 94 | 359 |
| propoziții cu ** | 4 | 0 | 1 | 3 | 5 | 262 | 1239 | 2 |
| propoziții cu // | 0 | 0 | 369 | 0 | 5 | 91 | 7 | 1034 |

**Tabelul 18**

Procentajul de majoritate poate fi mărit pentru o puritate mai mare.

# Capitolul III - Rețeaua neuronala:

Rezultatele obține de la clusterizare au fost folosite pentru învățarea rețelei neuronale. Toate propozițiile dintr-un fișier reprezintă o anumită etichetă.

Pentru a construi rețelele neuronale am folosit librăriile keras și tensorflow.

Prima dată am încercat crearea a 8 rețele neuronale (numite și modele), fiecare fiind specializată în clasificarea unui singur tip de propoziție.

Modelul are ca noduri de intrare exact aceeași structura care a fost folosită și la clusterizare .

```
model_1.add(keras.layers.Dense(360, input_dim=100, activation='sigmoid'))
model_1.add(keras.layers.Dense(1, activation='sigmoid'))

model_1.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
model_1.fit(X, Y, epochs=100, batch_size=5000)
```

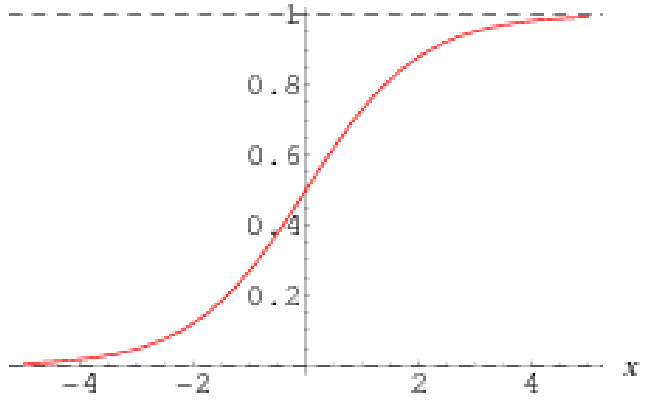



**Figura 4**:Graficul funcției sigmoid[7]

După cum se poate observa din *figura 4* valorile în această funcție lucrează cel mai bine când valorile sunt intre 4 si -4.

Modelele au un singur nod de ieșire care va avea o valoare intre 0 și 1 (datorita formulei sigmoidului: $\frac{1}{1+e^{-x}}$ ). Desi la antrenare aceste modele ajung la o acuratețe de aproximativ 99%, când sunt folosite pentru efectuarea unei predicții ele au o performanta slabă.

Pentru următoarele 4 propoziții, cu un singur semn de operație, alese din datele de antrenare :

- Draco has nineteen magic potions and seventy-three love potions. How many do we have now?
- There is Hermione that has seventy-four potions, but sixteen of them disappear. This being given, what is the left number?
- Being given ninety-three potions, someone decides to multiply this by ninety-three. What is the total now?
- There is napoletana that has eighty-three four cheeses toppings. Four cheeses toppings is divided by twenty-one. This being given, what is the left number?

Au avut următoarele predicții:

| | Model + | Model - | Model * | Model / | Model ++ | Model -- | Model ** | Model // |
|---|---|---|---|---|---|---|---|---|
| 1 | 1.*e-7 | 1. | 0.07 | 1.8*e-5 | 2.2*e-8 | 0.99 | 0.99 | 0.03 |
| 2 | 3.6*e-5 | 2.8*e-5 | 0.99 | 0.99 | 3.3*e-8 | 1.4*e-7 | 3.3*e-5 | 0.14 |
| 3 | 4.1*e-7 | 0.05 | 0.99 | 6.*e-11 | 4.4*e-8 | 4.1*e-5 | 0.99 | 0.47 |
| 4 | 0.99 | 0.63 | 0.99 | 1. | 1.0*e-7 | 2.0*e-8 | 1.05*e-9 | 0.72 |

**Tabelul 19**

Si pentru următoarele 4 cu 2 semne de operație:

- Hermione has sixty-seven phoenix wands and forty-four dragon wands. Twenty-seven of them are added. What is the total now?
- There is caprina that has twenty-four herb spice toppings, but ten of them disappear. From that amount, twenty-four are removed too. This being given, what is the left number
- There is named pizza that has ninety-one pizzas. Fifty-one times same amount of pizzas is added. If we add seventy-five times the same amount how many do we have now?
- There is rosa that has seventy-three olive toppings. This is divided by thirty-four. It is decided to share all to seventy-four friends. How many ingredients are left?

| | Model + | Model - | Model * | Model / | Model ++ | Model -- | Model ** | Model // |
|---|---|---|---|---|---|---|---|---|
| 1 | 1.41*e-7 | 0.01 | 0.99 | 0.89 | 1.91*e-8 | 0.0002 | 0.996 | 0.237 |
| 2 | 0.99 | 0.99 | 0.99 | 0.99 | 0.0001 | 4.1*e-8 | 6.5*e-9 | 0.001 |
| 3 | 0.02 | 0.99 | 0.99 | 1. | 4.41*e-8 | 5.6*e-8 | 6.2*e-10 | 0.010 |
| 4 | 6.5*e-6 | 0.0001 | 0.99 | 1. | 2.41*e-5 | 1.3*e-7 | 6.8*e-8 | 0.0001 |

**Tabelul 20**

Apoi am creat un singur model cu aceleași date de intrare și 8 noduri de ieșire. Am folosit funcția softmax pentru ca fiecare dintre aceste 8 noduri să reprezinte o probabilitate.

*model.add(keras.layers.Dense(1000, input_dim=100, activation='sigmoid'))*
*model.add(keras.layers.Dense(8, activation='softmax'))*

*model.compile(loss= 'categorical_crossentropy', optimizer='adam',*
*metrics=['categorical_accuracy'])*
*model.fit(X, Y, epochs=50, batch_size=2000)*

De data aceasta acuratețea maxima este de 97%(atinsa după 45 de epoci). Modelul este capabil să recunoască toate propozițiile în afara de minus-minus. Această performanță a ambelor rețele neuronale poate fi atribuită metodei de transformare a datelor din text în numere.
Pentru propozițiile de mai sus acest model a avut următoarele procentaje:

- Pentru propoziția 1: 96.919090% pentru o singura adunare
- Pentru propoziția 2: 46.991912% pentru o singura înmulțire
  46.548575% pentru o singura împărțire

- Pentru Propoziția 3: 27.318770% pentru o singura înmulțire
  48.224747% pentru doua adunări
  18.118219% pentru o singura adunare
- Pentru propoziția 4: 21.384722% pentru o singura înmulțire
  65.849906% pentru o singura împărțire
- Pentru propoziția 5: 94.737440% pentru doua adunări
- Pentru propoziția 6: 78.163338% pentru doua scăderi
  15.950073% pentru doua împărțiri
- Pentru propoziția 7: 70.514196% pentru doua scăderi
  13.048778%pentru o singura împărțire
- Pentru propoziția 8: 86.9897902% pentru doua scăderi
  11.5389444% pentru doua împărțiri

În continuare am folosit Universal Sentence Encoder pentru a efectua această transformare. De aici a rezultat următorul model:

*model.add(keras.layers.Dense(1024, input_dim=512, activation='tanh'))*

*model.add(keras.layers.Dense(8, activation='softmax'))*

*model.compile(loss='categorical_crossentropy', optimizer='adam',*

*metrics=['categorical_accuracy'])*

*model.fit(X, Y, epochs=50, batch_size=2000)*

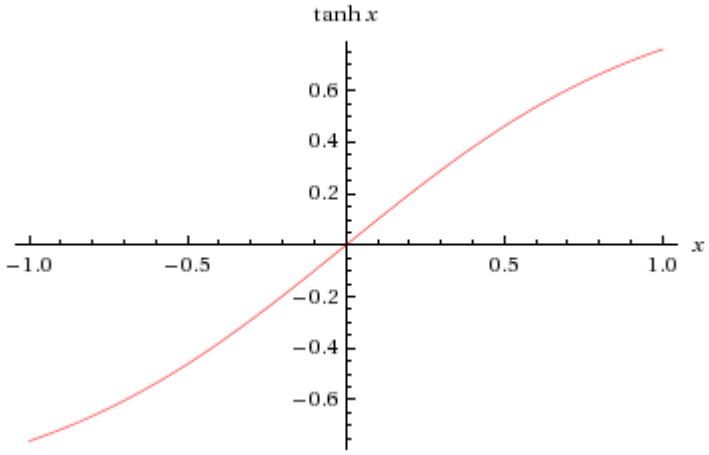



**Figura 5**:Graficul funcției TanH[7]

Universal Sentence Encoder returnează un vector de 512 cu valori intre -1 şi 1. Singurul dezavantaj este faptul ca procesarea a aproximativ 11000 de exemple durează mult (aproximativ 7 min.).

Totuși, acesta trebuie apelat doar când se folosesc alte date pentru clusterizare ( fișierele numite “message_embeddings” și “nn_weights” deja generate trebuie șterse în acest caz ) sau aceasta se efectuează pentru prima data la generarea datelor de antrenare la rețeaua neuronala.

După cum se poate observa in *figura 5* funcția TanH poate avea valori în intervalul (-1,1) iar sigmoidul (0,1).Datorită faptului că TanH poate avea și valori negative ea poate “observa” detalii mai fine.

Acuratețea maxima este de 97.5% și pierderea minima 0.0992 .

Pentru propozițiile de mai sus acest model a avut următoarele procentaje:

- Pentru propoziția 1: 91.89538955688477% pentru o singura adunare
- Pentru propoziția 2: 99.22131896018982%pentru o singura scădere
- Pentru Propoziția 3: 92.10307598114014% pentru o singura înmulțire
- Pentru propoziția 4: 97.09333777427673%pentru o singura împărțire
- Pentru propoziția 5: 86.54012084007263% pentru doua adunări
- Pentru propoziția 6: 87.10753321647644%pentru doua scăderi
- Pentru propoziția 7: 84.76188778877258% pentru o împărțire
- Pentru propoziția 8: 97.27210402488708% pentru doua scăderi

6/8 din aceste predicții sunt corecte.

Alegem 2000 de exemple la întâmplare și încercam să le clasificăm cu rețeaua neuronala ,pentru a putea observa mai ușor performanța acesteia, obținând următorul tabel:

| | + | - | * | / | ++ | -- | ** | // |
|---|---|---|---|---|---|---|---|---|
| Ex. Corecte | 167 | 262 | 242 | 273 | 171 | 179 | 200 | 203 |
| Ex. Greșite | 6 | 10 | 9 | 8 | 95 | 66 | 64 | 45 |

**Tabelul 21**

# Creșterea procentajului de majoritate minim

Datorită sensibilității rețelelor neuronale am mărit procentajul de majoritate la 80% pentru a îmbunătății performanta acestora. De asemenea numărul de sub-clusterizari a fost și el mărit la 2 (daca un cluster nu avea o majoritate acesta era clusterizat din nou, prin faptul că am mărit această valoare la 2 se face posibila o reclusterizare din nou).

**Rezultate obținute sunt următoarele (timpul de rulare este similar):**

| | File + | File - | File / | File * | File ++ | File -- | File // | File ** |
|---|---|---|---|---|---|---|---|---|
| + | **983** | 0 | 0 | 4 | 17 | 0 | 0 | 0 |
| - | 3 | **1503** | 1 | 0 | 5 | 0 | 0 | 0 |
| / | 7 | 1 | **1402** | 6 | 2 | 23 | 29 | 25 |
| * | 12 | 0 | 0 | **1495** | 0 | 0 | 0 | 0 |
| ++ | 2 | 0 | 0 | 55 | **1419** | 2 | 0 | 3 |
| -- | 0 | 0 | 0 | 0 | 6 | **1354** | 96 | 1 |
| // | 0 | 0 | 1 | 3 | 5 | 8 | **1473** | 1 |
| ** | 0 | 0 | 1 | 0 | 9 | 56 | 7 | **1386** |

**Tabelul 22**

Folosim aceleași propoziții ca și mai sus.

**Primul model a avut următoarele predicții:**

| | Model + | Model - | Model * | Model / | Model ++ | Model -- | Model ** | Model // |
|---|---|---|---|---|---|---|---|---|
| 1 | 3.6*e-7 | 1. | 0.03 | 8.9*e-9 | 6.8*e-10 | 0.99 | 0.001 | 0.0198 |
| 2 | 0.623 | 5.9*e-8 | 0.99 | 0.99 | 9.2*e-10 | 1.0*e-7 | 0.004 | 0.171 |
| 3 | 0.99 | 2.8*e-6 | 0.99 | 1. | 5.3*e-9 | 4.5*e-9 | 4.5*e-6 | 0.96 |
| 4 | 7.1*e-7 | 7.0*e-6 | 0.99 | 0.88 | 6.9*e-10 | 0.23 | 0.86 | 0.33 |

**Tabelul 23**

**Pentru al doilea model:**

| | Model + | Model - | Model * | Model / | Model ++ | Model --- | Model ** | Model // |
|---|---|---|---|---|---|---|---|---|
| 1 | 7.1*e-7 | 7.1*e-6 | 0.99 | 0.88 | 6.9*e-10 | 2.33 | 0.869 | 0.33 |
| 2 | 0.99 | 0.99 | 0.47 | 1. | 3.2*e-5 | 1.9*e-8 | 2.0*e-7 | 0.001 |
| 3 | 0.997 | 1.8*e-5 | 0.99 | 1. | 4.67*e-9 | 1.5*e-8 | 0.65*e-9 | 0.048 |
| 4 | 0.05 | 0.23 | 0.99 | 1. | 1.5*e-8 | 1.7*e-7 | 3.7*e-5 | 2.9*e-5 |

**Tabelul 24**

Pentru al treilea model:
Pentru propozițiile de mai sus acest model a avut următoarele procentaje:

- Pentru propoziția 1: 90.31792879104614% pentru o singura adunare
- Pentru propoziția 2: 99.07373785972595% pentru o singura scădere
- Pentru Propoziția 3: 99.37493801116943% pentru o singura înmulțire
- Pentru propoziția 4: 93.61878037452698% pentru o singura împărțire
- Pentru propoziția 5: 99.09936785697937% pentru doua adunări

- Pentru propoziția 6: 70.80990076065063% pentru doua scăderi și
  20.82643210887909% pentru doua împărțiri
- Pentru propoziția 7: 97.69148826599121% pentru doua înmulțiri
- Pentru propoziția 8: 96.80597186088562% pentru doua împărțiri

Toate aceste predicții sunt corecte.

Tabelul pentru cele 2000 de exemple:

| | + | - | * | / | ++ | -- | ** | // |
|---|---|---|---|---|---|---|---|---|
| Ex. Corecte | 172 | 279 | 244 | 271 | 251 | 233 | 264 | 243 |
| Ex. Greșite | 2 | 3 | 4 | 8 | 3 | 17 | 4 | 2 |

**Tabelul 25**

O propoziție care a fost clasificată greșit dar are procentajul foarte apropiat:

- There is petit pois topping that has twenty-two named pizzas, but sixty-nine of them disappear. Somehow, ninety-two are gone too. This being given, what is the left number?

Predicția pentru doua scăderi : 42.094653844833374%

Predicția pentru doua împărțiri :57.660818099975586%

In continuare avem câteva propoziții care nu folosesc regulile pe care s-a efectuat antrenarea:

- Anna has five apples. Ben has the same amount of apples. How many apples do they have în total?

Predicția pentru o adunare : 72.19850420951843%

Predicția pentru doua adunări : 22.019772231578827%

Aici avem un rezultat corect deoarece structura este similară. Singura diferență fiind faptul că al doilea numeral a fost înlocuit de cuvântul “same”.

- The local hospital has 105 patients, while the central hospital has two times more. How many patients are in the central hospital?

Predicția pentru o adunare: 80.19673228263855%

Propoziția nu folosește forma specifică finală pentru propozițiile de antrenament “how many are left” “what is the number now” și nu exista termenul “add” specific pentru exemplele cu înmulțire. Deși cuvântul „times” există propoziția a fost clasificată ca plus datorită lipsei termenului specific.

- Mary has a bouquet with 63 tulips, five rose and one orchid. How many she has got?

Predicția pentru o adunare :29.396802186965942%

Același caz ca si mai sus, propoziția nu are termen specific și nu este suficient de lungă pentru două adunări.

Predicția pentru o împărțire : 32.53012001514435%

Predicția pentru doua împărțiri: 53.16900610923767%

**Concluzie:**

Datorita purității mare a clusterelor și structura de reprezentarea datelor, creată de tensorflow Universal-Language-Encoder, rețeaua neuronala are o acuratețe foarte mare. Totuși, ea nu poate recunoaște exemplele de mai sus datorită faptului ca datele de antrenament nu conțin asemenea propoziții și nu sunt similare cu acestea.

## Fluxul Programului:

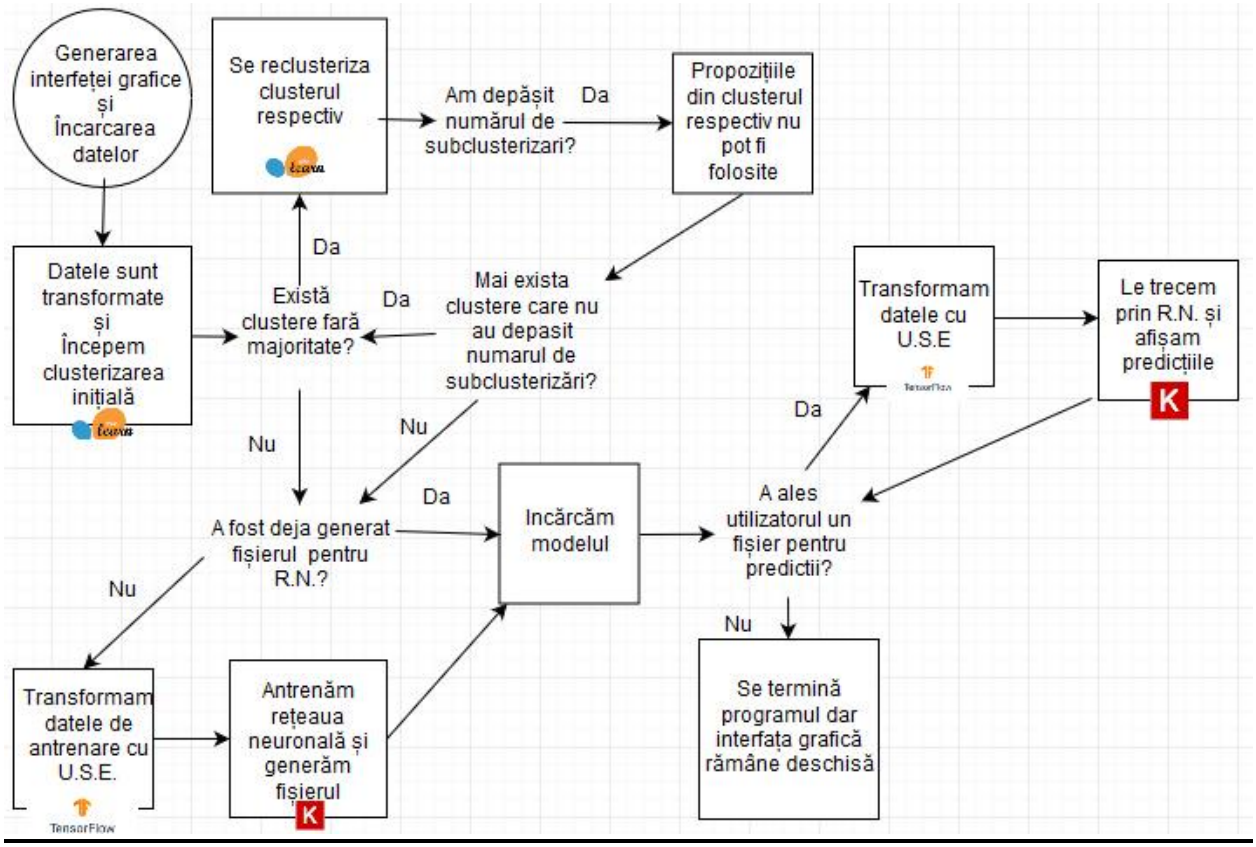



**Figura 6**

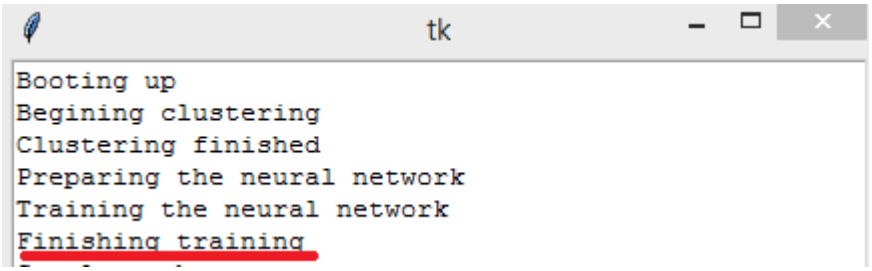
U.S.E- Universal Sentence Encoder
R.N.- Rețea Neuronală

## Interfața grafică:

Am creat o interfață grafică relativ simplă pentru a crește calitatea interacțiunilor dintre user și aplicație.

**Mod de utilizare:**

1. Se pornește aplicația
2. Se așteaptă apariția mesajului "Finished training" ca mai jos:




3. După apariția acestui mesaj pe ecran v-a apărea o noua fereastra care va aștepta selecția fișierului care conține propozițiile pe care se face predicțiile

Fișierul trebuie să aibă extensia .txt și să aibă următorul format:

Anna has five apples. Ben has the same amount of apples. How many apples do they have în total?
From sixty-six hectares thirteen are sold. How many are left?
John has 416 leaves. Mary has fifteen times less. How many leaves Mary has?

O singură propoziție pe fiecare linie, numeralele peste 100 să fie scrise în formă de cifre iar cele sub 100 pot fi scrise și în cuvinte dar să aibă o cratimă între ele (twenty, one ,sixty-seven).Dacă ele nu au o cratimă intre ele algoritmul le va considera ca fiind 2 numere diferite. De asemenea limba , care trebuie folosită obligatoriu datorită modulului nltk , este engleză. De asemenea, nu trebuie să existe caractere din alte formate ( caractere chinezești, caractere chirilice).

4. După selecția fișierului, în funcție de număr de propoziții conținut de acesta și de specificațiile calculatorului, interfața grafică poate da impresia că nu mai funcționează. Acesta nu este cazul. De vină este modului Universal Sentence Encoder din tensorflow care trebuie să acceseze serverul.
5. După terminarea rulării fereastra de la punctul 3 apare din nou. Aceasta se poate închide pentru a vizualiza rezultatele sau se poate alege un alt fișier pentru alte predicții. Rezultatele deja scrise pe ecran nu vor fi șterse.

Aceste rezultate au următorul format:

```
Sample number:
2
Prediction for one substraction :
99.82188344001778
```

Se vor afișa doar predicțiile care au peste 20%.

# Provocări întâmpinate și direcții de viitor

## Provocări întâmpinate:

Cea mai mare provocare întâmpinată a fost procesarea limbajului natural pentru a putea obține o clusterizare concretă , care să ofere , cel puțin, îmbunătățiri majore față de o clusterizare la întâmplare datorită structurii propozițiilor.

Aducerea rețelelor neuronale într-un stadiu în care să poată recunoaște toate tipurile de propoziție cu o acuratețe mare. Aici am avut probleme cu ajustarea parametrilor rețelelor neuronale din keras și alegerea funcțiilor de activare și de optimizare potrivite.

## Direcții de viitor:

Există mai multe aspecte care pot fi îmbunătățite sau modificate:

- Datorită metodei de procesare a datelor aleasă (NLTK tagging) propozițiile trebuie să aibă o structura similară pentru a putea fi clusterizate. Desigur, posibilitatea de a avea 10.000 de propoziții care să aibă o structura diferita , să se refere la un singur tip de expresie matematica și să avem cel puțin 10% astfel de exemple din datele totale de antrenament este foarte mică. Într-un caz ca acesta clusterizarea v-a avea o performanta foarte slaba.
- Se poate încerca procesarea a mai multor tipuri de expresii (mai mult de 8) în același timp. Acest lucru nu este posibil datorită numărului de clustere ales în k-means.
- Crearea si păstrarea unei baze de date mare și diversa pentru a putea recunoaște un număr mai mare de tipuri de propoziții. Cel mai probabil structura rețelei neuronale trebuie schimbată pentru a putea acomoda un număr mai mare de caracteristici.
- Recunoașterea constantelor (de exemplu If the current month is November, how many months are left until the new year?)

# Bibliografie:


- **[1]** Van Rossum, Guido, and Fred L. Drake. "Python language reference manual." (2003).
http://knuth.luther.edu/~bmiller/CS151/Spring05/pythonref.pdf

- **[2]** Oliphant, Travis E. *A guide to NumPy*. Vol. 1. USA: Trelgol Publishing, 2006.
https://oez.es/Guide%20to%20NumPy.pdf

- **[3]** Pedregosa, Fabian, et al. "Scikit-learn: Machine learning în Python. " *Journal of machine learning research* 12.Oct (2011): 2825-2830:
http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

- **[4]**Scikit-learn: User guide
https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
(Accesat ultima data iunie 2019)

- **[5]**Clustering în Scikit:
https://scikit-learn.org/stable/modules/clustering.html#k-means
(Accesat ultima data iunie 2019)

- **[6]** Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).
https://arxiv.org/pdf/1412.6980.pdf

- **[7]** Sigmoid function:
http://mathworld.wolfram.com/SigmoidFunction.html
(Accesat ultima data iunie 2019)

- **[8]** Hyperoblic Tangent function:
http://mathworld.wolfram.com/HyperbolicTangent.html
(Accesat ultima data iunie 2019)

- **[9]** Python executive summary:
https://www.python.org/doc/essays/blurb/
(Accesat ultima data iunie 2019)

- **[10]** Singh, Rohit, Sumit Gulwani, and Sriram Rajamani. "Automatically generating algebra problems." *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.
https://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/viewFile/5133/5310

- **[11]** Lemon, Olivier, and Olivier Pietquin. "Machine learning for spoken dialogue systems." *European Conference on Speech Communication and Technologies (Interspeech'07)*. 2007.
  https://hal-supelec.archives-ouvertes.fr/file/index/docid/216035/filename/Supelec270.pdf

- **[12]** Collobert, Ronan, and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.
  http://www.thespermwhale.com/jaseweston/papers/unified_nlp.pdf