UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Sumarizare Text

propusă de

Mihnea Bîgu

Sesiunea: *iulie, 2019*

Coordonator științific

Lect. Dr. Bogdan Pătruț

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Sumarizare Text

Mihnea Bîgu

Sesiunea: *iulie, 2019*

Coordonator științific

Lect. Dr. Bogdan Pătruț

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele

_____

Data _____ Semnătura _____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemntatul(a)

……

domiciliul în

……..

născut(ă) la data de ……..…., identificat prin CNP ……..…….. ……..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza" din Iași, Facultatea de …… specializarea ……, promoția ……., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul:

_____

_____

_____

__elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice in vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… S e m n ă t u r ă s t u d e n t ……

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Mihnea Bîgu*

_____

(semnătura în original)

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil şi sursă, să aparţină autorului prezentei lucrări, *Mihnea Bîgu.*

Încheierea acestui acord este necesară din următoarele motive:

[Se explică de ce este necesar un acord, se descriu originile resurselor utilizate în realizarea produsului-program (personal, tehnologii, fonduri) şi aportul adus de fiecare resursă.]

Iaşi, *data*

Decan *Adrian Iftene* Absolvent *Mihnea Bîgu*

_____

_____

(semnătura în original) (semnătura în original)

Cuprins

Introducere …… 8

1. Descrierea problemei …… 9

2. Abordări anterioare …… 10

2.1. Metoda extractivă …… 10

2.2. Metoda abstractă …… 10

3. Descrierea soluției …… 11

3.1. Preprocesarea textului …… 11

3.2. Calcularea vectorilor fiecărei propoziții prin TF-IDF …… 12

3.3. Construirea matricii de similaritate dintre propoziții …… 15

3.4. Calcularea scorului final al fiecărei propoziții …… 17

3.5. Construirea rezumatului prin gruparea propozițiilor cu cel mai bun scor …… 19

Concluziile lucrării …… 21

Bibliografie …… 22

Introducere

În cadrul acestui proiect se urmărește găsirea unei soluții pentru sumarizarea automată a unui text. Unul dintre principalele motive pentru alegerea acestei teme îl reprezintă dezbaterile politice ce au loc în cadrul alegerilor prezidențiale din turul al doilea, mai exact, momentul în care cei doi candidați rămași trebuie să își exprime argumentele pentru care ei merită să fie aleși în poziția de președinte. Discursurile acestora pot fi deseori mult prea complicate pentru a fi enunțate în timpul admis, de regulă de trei minute, iar o astfel de aplicație ar putea fi de ajutor pentru comprimarea discursului și păstrarea ideilor principale, fără a deteriora conținutul.

Pentru realizarea aplicației, am utilizat metoda extractivă de sumarizare a unui text, inspirată de algoritmul celor de la Google, numit PageRank. Astfel, soluția este bazată pe reprezentarea textului sub formă de graf, fiind folosită tehnica TextRank pentru a stabili care sunt propozițiile cu cel mai mare interes în alcătuirea rezumatului.

1. Descrierea problemei

În general, ne dorim ca informația să poată fi accesată într-un mod cât mai ușor, însă acest lucru nu este întotdeauna posibil, întrucât nivelul de informație poate fi prea mare pentru a fi digerat în timp util. Astfel, în ultima perioadă s-au alocat resurse pentru căutarea unei soluții a acestei probleme. Au apărut, așadar, diferite metode de construire a unui rezumat al textului inițial, fără a altera conținutul sau mesajul transmis.

În principiu, metodele de abordare sunt cele extractive sau abstracte. Prima categorie are ca beneficiu faptul că rezumatul este alcătuit prin extragerea propozițiilor de interes din textul inițial, fără a modifica conținutul, însă nu este foarte flexibilă în ceea ce privește textele ce nu sunt de tip articol. A doua categorie ne permite construirea unui rezumat al unui roman, spre exemplu, deoarece putem construi fraze de legătură între două sau mai multe propoziții, însă ar trebui să construim un model ce ar putea genera conținut coerent și care să aibă sens în contextul prezentat.

2. Abordări anterioare

2.1. Metoda extractivă

Una dintre cele mai folosite metode de rezolvare este cea bazată pe modelul unui graf. De regulă, este folosită tehnica TextRank, în care fiecare nod reprezintă propozițiile textului inițial, iar muchiile semnifică relațiile dintre propoziții. Pentru determinarea importanței unei propoziții din text, este folosit un algoritm asemănător celui de la Google, PageRank. Mai sunt folosite, de asemenea, modele bazate pe caracteristicile unei propoziții, cum ar fi poziția acesteia în cadrul textului, fontul folosit, lungimea ei, frecvența termenilor, prezența verbelor și așa mai departe, modele bazate pe tema textului, în care aceasta este folosită în principal pentru punctarea unei propoziții cu un scor mai mic sau mai ridicat, în funcție de legătura pe care o are cu subiectul prezentat, sau chiar modele bazate pe gramatica textului, în care propoziții precum ”în cadrul conflictelor de anvergură din Orientul Mijlociu” devin ”în Orientul Mijlociu”.

2.2. Metoda abstractă

O altă metoda de rezolvare a acestei probleme este prin folosirea modelului de codificare-decodificare. El este folosit deseori atunci când se dorește construirea rezumatului în special al unui roman, spre exemplu. Acest principiu are la bază transformarea propozițiilor în forma unor vectori (codificare), iar apoi decodificatorul generează rezumatul textului folosindu-se de acei vectori. Sunt ridicate câteva probleme, precum: trebuie ca rezumatul să fie cât mai coerent și să aibă sens în contextul prezentat, trebuie folosit un vocabular cât mai larg, cum putem verifica faptul că un cuvânt sau o frază care are o importanță foarte mare dar care apare foarte rar nu este neglijată și așa mai departe.

3. Descrierea soluției

3.1. Preprocesarea textului

Pentru rezolvarea acestei probleme, am abordat o metodă extractivă, folosind modelul grafului bazat pe TextRank. În primul rând, textul inițial a fost împărțit în propoziții, rezultând astfel o listă ce conține toate propozițiile textului.

```
# Separate the text into sentences and return them in an array of sentences

sentences = nltk.tokenize.sent_tokenize(text)
```

Apoi, toate cuvintele din cadrul propozițiilor au fost transformate încât toate literele să fie mici, după care am eliminat toate simbolurile, cum ar fi semnele de punctuație, dar și de ortografie, iar apoi am eliminat cuvintele de legătură ce nu au o semnificație importantă, numite ”stopwords”:

```
# Clean the sentences: make everything lowercase, eliminate any symbol, except the ' symbol and remove the stopwords

cleanSentences = [sentence.lower() for sentence in sentences]

cleanSentences = [''.join(letter for letter in sentence if
                          not letter.isdigit() and letter not in [',', '.', '"', ':', ';', '!', '?', '”', '“', '–',
                                                                  '&']) for sentence in cleanSentences]

cleanSentences = [removeStopwords(sentence.split()) for sentence in cleanSentences]
```

Pasul următor este de a extrage toate cuvintele rămase din propozițiile curățate și să le introducem într-un set pentru a nu apărea dubluri.

```
# Make a set of all the words found in the cleanSentences

words = set()

for sentence in cleanSentences:
    for word in sentence.split():
        words.add(word)
```

3.2. Calcularea vectorilor fiecărei propoziții prin TF-IDF

Coeficientul TF-IDF (Term Frequency - Inverse Document Frequency) este un coeficient ce ne indică cât de important este un cuvânt în cadrul unui document. În cazul nostru, deoarece lucrăm pe propoziții, vom face referire la frecvența unui cuvânt în relație cu acea propoziție, dar și cu întreg contextul.

TF sau Term Frequency (Frecvența termenului) măsoară numărul de apariții ale unui cuvânt în cadrul unei propoziții împărțit la numărul de cuvinte din acea propoziție. Spre exemplu, dacă avem urmatoarele documente:

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Putem observa că termenul ”this” apare în primul document o singură dată, iar în al doilea la fel, o singură dată. Astfel, calculul TF pentru cuvântul ”this” în raport cu cele doua documente este:

$$\mathrm{tf}(''\mathsf{this}'', d_1) = \frac{1}{5} = 0.2$$
$$\mathrm{tf}(''\mathsf{this}'', d_2) = \frac{1}{7} \approx 0.14$$

unde 5 reprezintă numărul de cuvinte din primul document (1+1+2+1), iar 7 numărul de cuvinte din al doilea document (1+1+2+3).

IDF sau Inverse Document Frequency (Frecvența inversă al unui termen în cadrul documentului) reprezintă o măsură prin care putem vedea câtă informație aduce acel termen în cadrul acelui text. Astfel, un termen precum ”the” ce apare foarte des în cadrul propozițiilor ne-ar putea induce în eroare dacă am calcula doar coeficientul TF, întrucât ar rezulta faptul că acel cuvânt este foarte important. Calculul acestui coeficient IDF este realizat prin împărțirea numărului de documente (sau, în cazul nostru, numărul propozițiilor) la numărul de documente în care apare acel cuvânt, iar apoi logaritmarea acestui rezultat în baza 10. Dacă facem referire la exemplul anterior, atunci pentru cuvântul ”this” formula de calcul este următoarea:

$$\text{idf}(''\textsf{this}'', D) = \log\left(\frac{2}{2}\right) = 0$$

Termenul ”this” apare în ambele documente, deci coeficientul IDF pentru el este egal cu 0.

În final, pentru calculul TF-IDF al unui cuvânt în relație cu o anumită propoziție, trebuie să înmulțim cele doua rezultate, mai exact TF * IDF:

$$\text{tfidf}(''\textsf{this}'', d_1, D) = 0.2 \times 0 = 0$$
$$\text{tfidf}(''\textsf{this}'', d_2, D) = 0.14 \times 0 = 0$$

Se observă, așadar, că termenul ”this” din acest exemplu nu este foarte important. Dacă am lua ca exemplu termenul ”example”, vom avea:

$$\text{tf}(''\textsf{example}'', d_1) = \frac{0}{5} = 0$$
$$\text{tf}(''\textsf{example}'', d_2) = \frac{3}{7} \approx 0.429$$
$$\text{idf}(''\textsf{example}'', D) = \log\left(\frac{2}{1}\right) = 0.301$$

Iar în cele din urma:

$$\text{tfidf}(''\text{example}'', d_1, D) = \text{tf}(''\text{example}'', d_1) \times \text{idf}(''\text{example}'', D) = 0 \times 0.301 = 0$$
$$\text{tfidf}(''\text{example}'', d_2, D) = \text{tf}(''\text{example}'', d_2) \times \text{idf}(''\text{example}'', D) = 0.429 \times 0.301 \approx 0.129$$

Astfel, pentru fiecare propoziție din textul nostru, vom construi câte o listă (vector) ce va conține coeficienții TF-IDF al fiecărui cuvânt din text în relație cu acea propoziție:

```
#TF-IDF

def tf(word, sen):
    if len(sen.split()) == 0:
        length = 0.01
    else:
        length = len(sen.split())
    return sen.split().count(word)/length

def idf(word, cleanSentences):
    appears = 0
    for sentence in cleanSentences:
        if word in sentence.split():
            appears += 1
            continue
    return math.log10(len(cleanSentences)/appears)
```

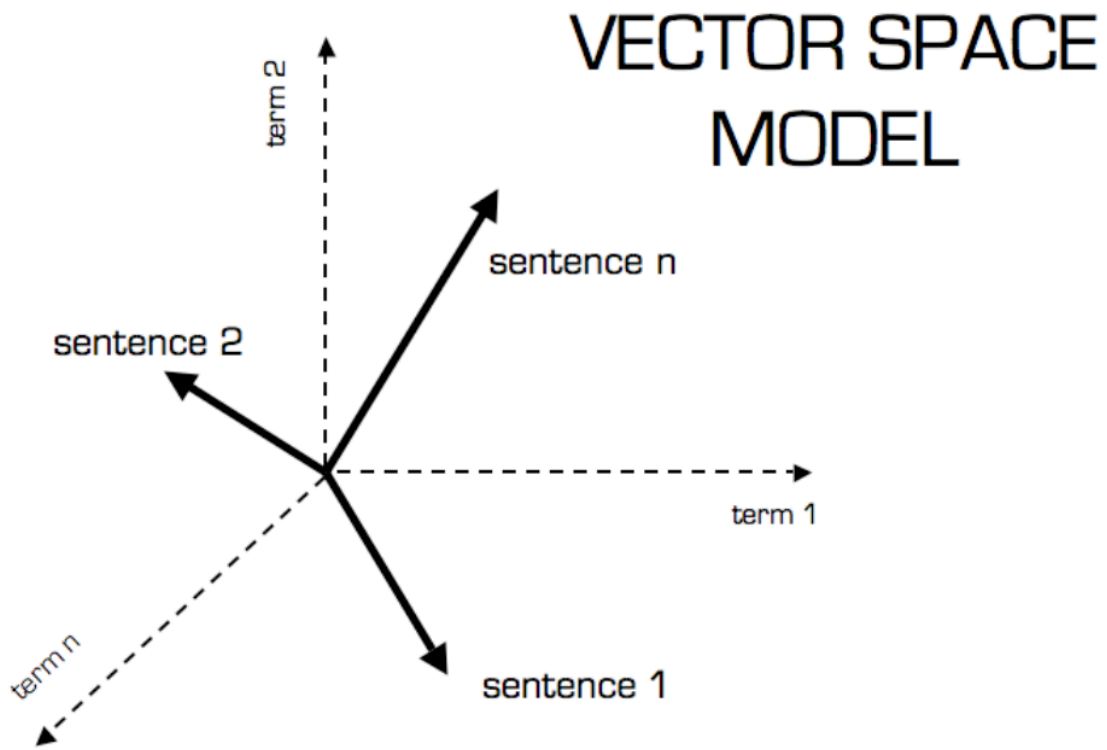
(funcții ajutătoare pentru calculul TF și IDF)

```
# Construct the sentenceVectors, i.e. calculate the TF-IDF for each word in relation to each sentence
# The final array will be of size equal to the number of sentences

sentenceVectors = []

for sentence in cleanSentences:
    sentenceVector = []
    for word in words:
        sentenceVector.append(tf(word, sentence) * idf(word, cleanSentences))
    sentenceVectors.append(sentenceVector)
```

(construirea vectorilor propozițiilor pe baza coeficienților TF-IDF)

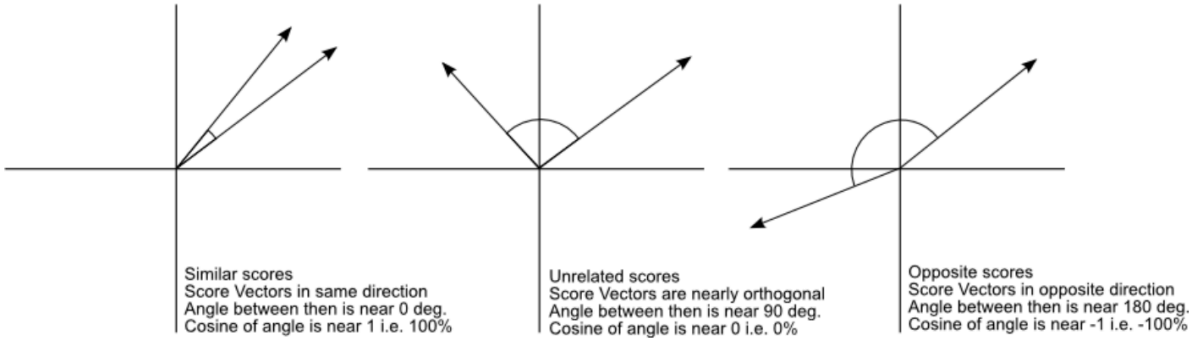
3.3. Construirea matricii de similaritate dintre propoziții

Odată ce am construit pentru fiecare propoziție câte un vector ce conține coeficienții TF-IDF al fiecărui cuvânt din text, acești vectori ar putea fi reprezentați într-un spațiu cartezian:



Vector Space Model

Astfel, pentru determinarea similarității dintre două propoziții este suficient să calculăm cosinusul dintre vectorii acelor două propoziții. Pentru a realiza acest lucru, vom calcula mai întâi produsul scalar al celor doi vectori, apoi vom calcula produsul dintre modulele lor, iar în final vom împărți primul rezultat la cel de-al doilea.



The Cosine Similarity values for different documents, 1 (same direction), 0 (90 deg.), -1 (opposite directions).

Așadar, deoarece funcția cosinus are valori în intervalul [-1, 1], rezultatul calculului de mai sus ne indică cu exactitate daca două propoziții sunt similare (rezultatul este în intervalul (0, 1]) sau dacă două propoziții nu au vreo legătură una cu cealaltă (rezultatul este egal cu 0). Cum coeficientul TF-IDF nu poate fi negativ, atunci nici cosinusul dintre cei doi vectori nu poate fi negativ. Astfel, doua propoziții sunt foarte similare dacă rezultatul cosinusului dintre vectorii lor este apropiat de valoarea 1.

```
#Function to determine the cosine similarity between two sentence vectors

def cosineSimilarity(sen1, sen2):
    product = 0
    for i in range(len(sen1)):
        product += sen1[i] * sen2[i]
    return product/(np.linalg.norm(sen1) * np.linalg.norm(sen2))
```

(funcție ajutătoare pentru calcularea cosinusului dintre vectorii propozițiilor)

```
# Construct the similarityMatrix between the sentences using the cosineSimilarity function

similarityMatrix = np.zeros([len(cleanSentences), len(cleanSentences)])

for i in range(len(cleanSentences)):
    for j in range(len(cleanSentences)):
        if i != j:
            similarityMatrix[i][j] = cosineSimilarity(sentenceVectors[i], sentenceVectors[j])
```

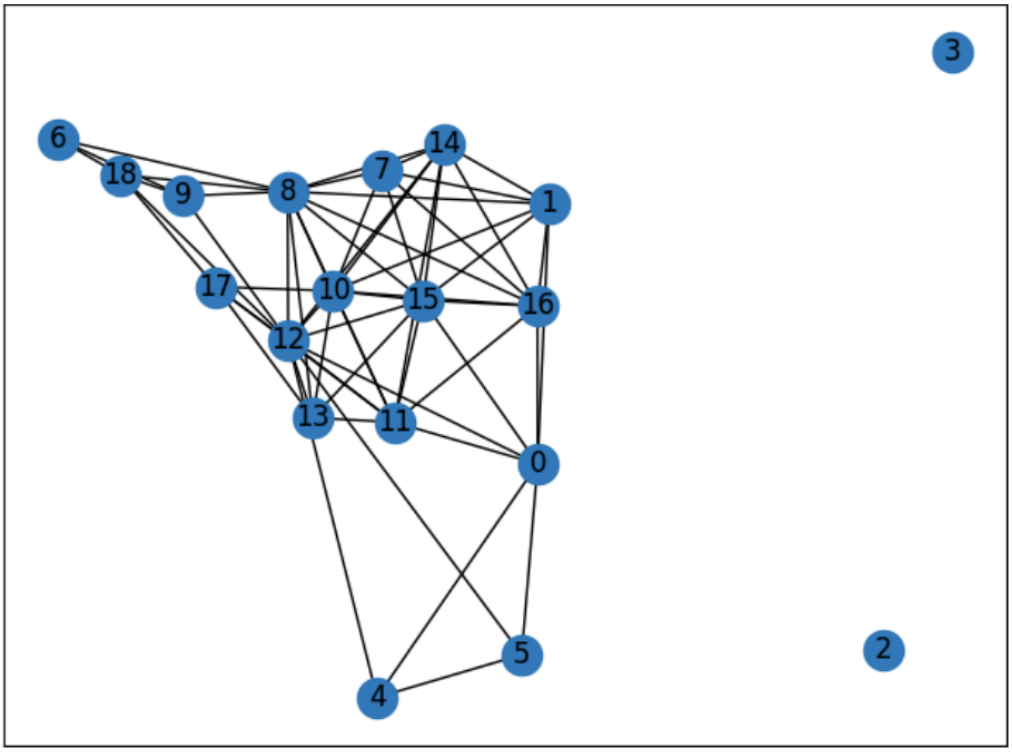
(construierea matricii de similaritate cu ajutorul funcției cosinus de similaritate)

3.4. Calcularea scorului final al fiecărei propoziții

În cele din urmă, pentru calcularea scorului propozițiilor, ne vom folosi de matricea de similaritate calculată anterior, după care vom aduna scorurile fiecărei propoziții (suma muchiilor ce trec printr-o anumită propoziție) și le vom salva într-un dicționar, unde cheia o reprezintă numărul propoziției din cadrul textului, iar valoarea este scorul propoziției respective. Vom aduce aceste scoruri la o stare în care suma lor să fie egală cu 1, încât ele vor reflecta propabilitatea ca acele fraze să fie selectate pentru construirea rezumatului final.

```
# Construct the finalScores
# For each sentence, we calculate the sum of the incoming edges score

finalScores = {}
for j in range(len(cleanSentences)):
    score = 0
    for i in range(len(cleanSentences)):
        if i != j:
            score += similarityMatrix[i][j]
    finalScores[j] = score

# After we calculate the finalScores, we normalize the final vector so that the sum of all the sentence scores equals to 1

finalScoresNormalize = sum(list(finalScores.values()))
for key in finalScores.keys():
    finalScores[key] = finalScores[key] / finalScoresNormalize
```

Spre exemplu, graful asociat unui text, în urma construirii rezumatului acestuia, în care nodurile reprezintă propozițiile, iar muchiile sunt legaturile dintre ele, ar putea arăta în felul următor:



După cum se poate observa, propozițiile cu indicele 2 și 3 nu sunt importante în cadrul textului nostru, deoarece nicio propoziție nu face referință la ele, iar nucleul important se află în jurul nodurilor 10, 11, 12, 13, etc.

3.5. Construirea rezumatului prin gruparea propozițiilor cu cel mai bun scor

Tot ce rămâne de făcut este să selectăm propozițiile cu cel mai bun scor pentru construirea rezumatului. Acest lucru îl putem face cu ușurință prin ordonarea descrescatoare a scorurilor propozițiilor și afișarea unui număr prestabilit de propoziții din top.

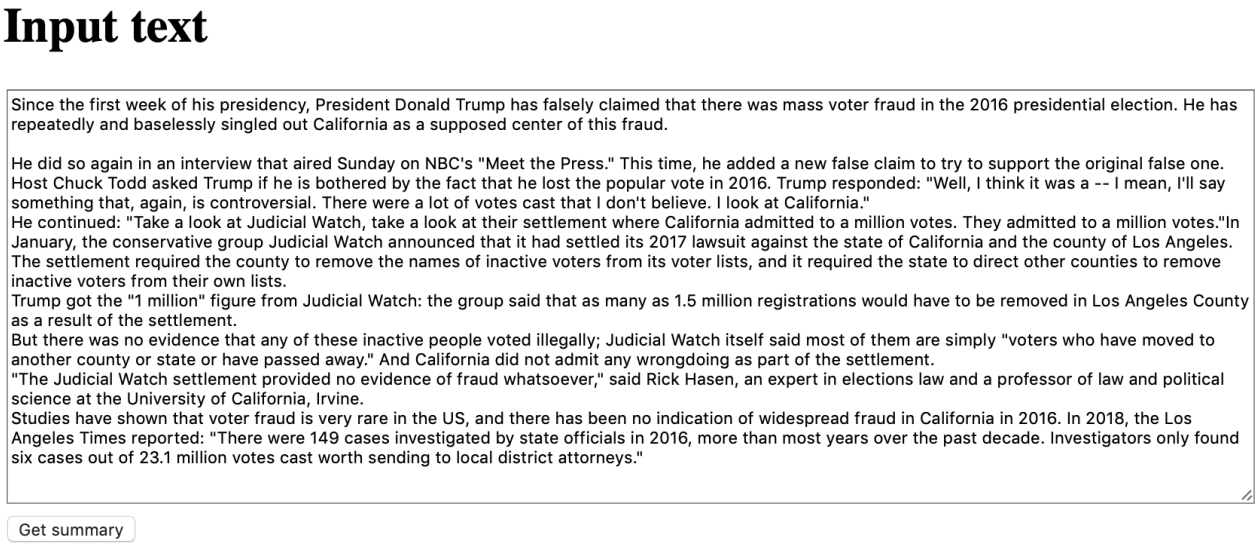
```
#The number of sentences to be printed out for the summary

TOP_RANKED = 5
```

(numărul de propoziții ce vor fi selectate pentru alcătuirea rezumatului)

```
# We construct the final summary using the first TOP_RANKED sentences

summary = ""
ranked = sorted(finalScores.items(), key=operator.itemgetter(1), reverse=True)

for i in range(TOP_RANKED):
    summary += sentences[ranked[TOP_RANKED - i - 1][0]] + '\n'

return summary
```

(construirea rezumatului final)

Pentru exemplificare, am folosit un articol din publicația CNN cu ajutorul căruia am construit un astfel de rezumat:



Output text

Investigators only found six cases out of 23.1 million votes cast worth sending to local district attorneys."
But there was no evidence that any of these inactive people voted illegally; Judicial Watch itself said most of them are simply "voters who have moved to another county or state or have passed away."
"In January, the conservative group Judicial Watch announced that it had settled its 2017 lawsuit against the state of California and the county of Los Angeles.
I look at California."
They admitted to a million votes.
Trump got the "1 million" figure from Judicial Watch: the group said that as many as 1.5 million registrations would have to be removed in Los Angeles County as a result of the settlement.
He continued: "Take a look at Judicial Watch, take a look at their settlement where California admitted to a million votes.

Concluziile lucrării

Această lucrare propune o soluție eficientă de construire a unui rezumat pe baza unui text. Consider că scopul principal a fost atins, acela fiind de a ușura colectarea de informație ale articolelor și nu numai, fără a pierde mesajul acestora.

Ca aspecte de viitor, aș încerca înlocuirea metodei de selectare a cuvintelor cheie prin metoda TF-IDF cu una ce se bazează pe o rețea de cuvinte (word embeddings) precum GloVe. De asemenea, aș implementa și o metodă de sumarizare abstractă, pentru a facilita și construirea de rezumate pentru texte ce fac parte din spectrul literaturii.

Bibliografie

- https://gist.github.com/umer7/f61d6cc4ed35a70c7664deefe3822540
- https://en.wikipedia.org/wiki/Tf–idf
- http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/
- https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/