


Extragerea și sumarizarea informației din pagini web



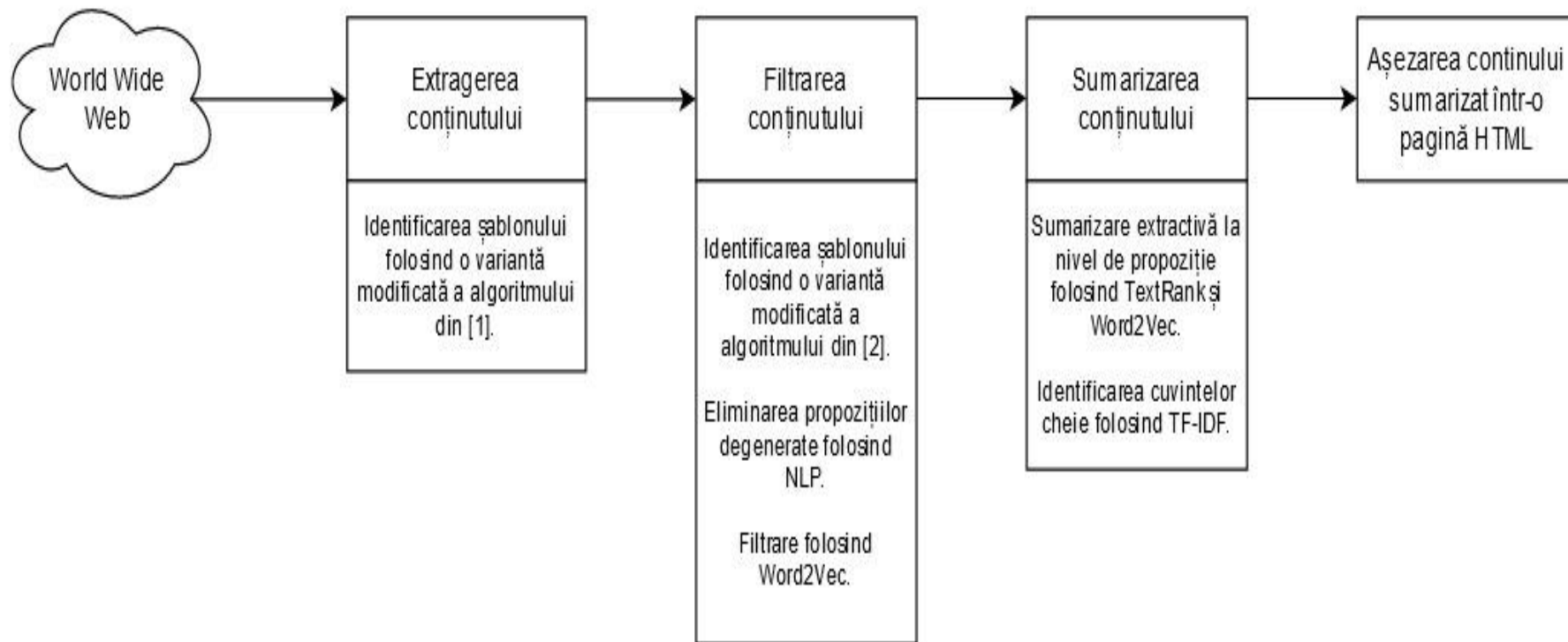
Coordonator științific: Asist.dr. Mihăiță Drăgan

Absolvent: Nicolae Cristian-Cătălin

Cuprins

1. Introducere
2. Concepte teoretice
 - a. Tehnici clasice de procesare a limbajului natural (NLP).
 - b. Term Frequency - Inverse Document Frequency (TF-IDF)
 - c. Word2Vec.
 - d. TextRank
3. Etapa de extragere
4. Etapa de filtrare
5. Etapa de sumarizare
6. Concluzii
7. Demo
8. Bibliografie

1. Introducere



2.a Tehnici clasice de procesare a limbajului natural

- Segmentarea conținutului în propoziții. În OpenNLP se face pe baza algoritmului de segmentare “Maximum Entropy Markov Model” (MEMM).
- Segmentarea propozițiilor în cuvinte. Se începe cu împărțirea după spații, dar apoi sunt evaluate toate cuvintele non-alfanumerice pentru a decide dacă acestea trebuie să fie separate.
- Ultima metodă aplicată este cea care etichetează fiecare cuvânt din propoziție cu o parte de vorbire. Se folosește tot un model bazat pe entropie maximă antrenat pe limba engleză.

2b. Term frequency - Inverse Document Frequency

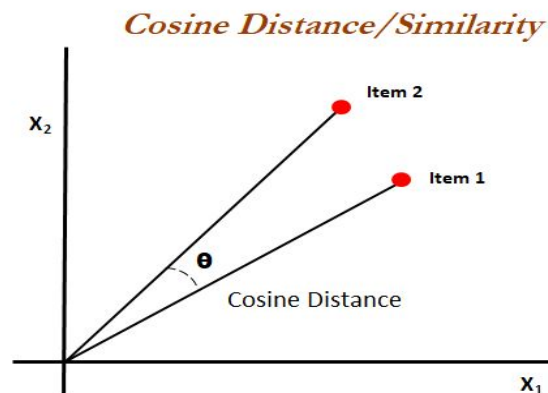
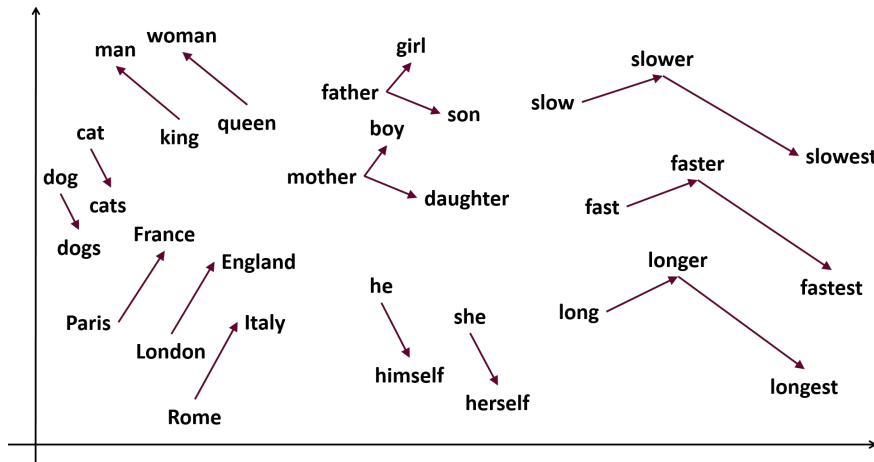
- Factorul “Term frequency” este dat de formula $\frac{Nt}{St}$ unde:
 - Nt este numărul de apariții al unui termen într-un text.
 - St este numărul total de termeni din document.

De obicei această valoare se normalizează prin împărțirea la lungimea documentului.

- Factorul “Inverse Document Frequency” este dat de formula $IDF = \ln\left(\frac{Sd}{N_{dt}}\right)$ unde:
 - Sd este numărul total de documente.
 - N_{dt} este numărul de documente în care apare termenul t .

2c. Word2Vec

- Reprezentare numerică a cuvintelor ca vectori într-un spațiu de dimensiune 300, vectori pe care putem face operații precum adunarea, scăderea sau produsul scalar.
- Exemplul oferit de autori este faptul că operația “ $\text{vec}(\text{Madrid}) - \text{vec}(\text{Spania}) + \text{vec}(\text{Franța})$ ” este mai aproape de vectorul $\text{vec}(\text{Paris})$ decât de orice alt vector învățat.



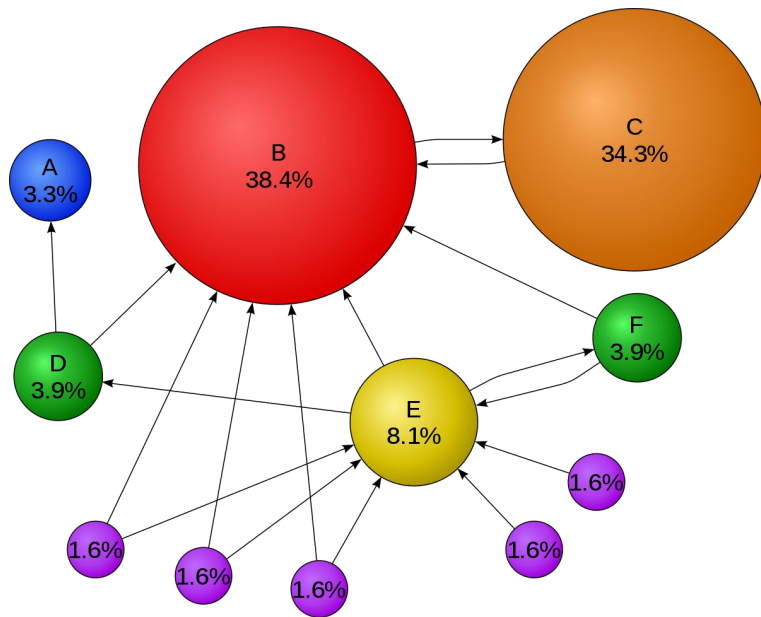
1. https://miro.medium.com/max/3902/1*hELIVp9hmZbDZVFstS61pg.png
2. <https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/assets/2b4a7a82-ad4c-4b2a-b808-e423a334de6f.png>

2d. TextRank

- Bazat pe PageRank, algoritmul de la Google oferă un mod de a asigna un indice de importanță unor pagini. Algoritmul primește ca intrare un graf orientat în care fiecare pagină este un nod, iar muchiile vor avea ponderi ce vor reprezenta costul de la un nod altul.
- Formula de calcul a indexul la nivel de nod:

$$PR(A) = (1 - d) + d \left(\frac{PR(B)}{C(B)} + \frac{PR(C)}{C(C)} + \frac{PR(D)}{C(D)} + \dots \right)$$

- Intuiția: persoana ce navighează aleatoriu.

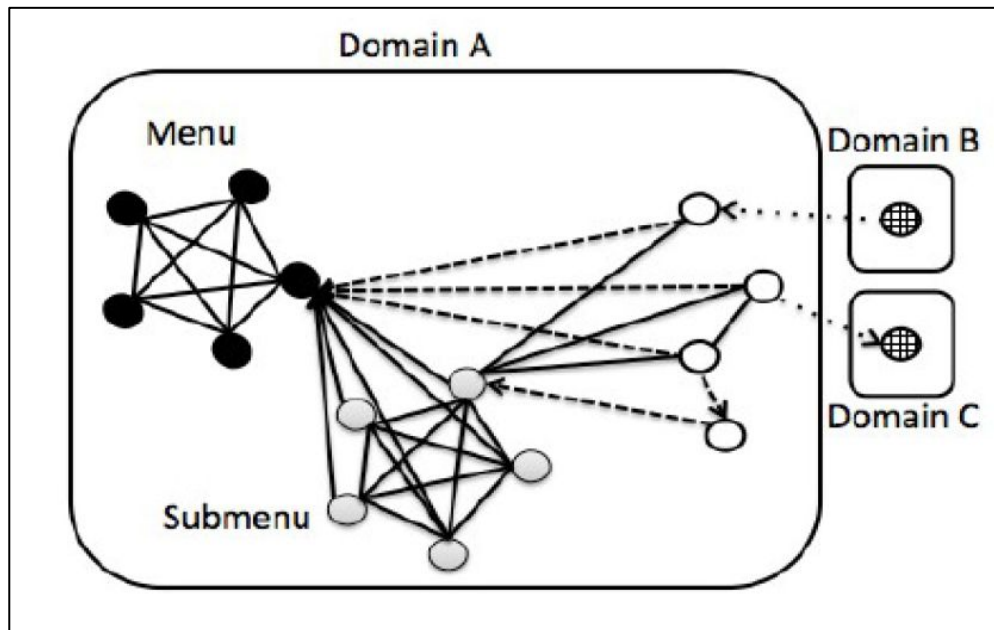


2d. TextRank

- TextRank folosește algoritmul PageRank cu propoziții în loc de pagini, oferind cele mai relevante propoziții din text.
- Ponderile de pe muchiile grafului sunt lăsate libere la implementare. În cazul de față, vom folosi similaritatea propozițiilor pe bază de cosinus. Pentru a reduce o propoziție la un singur vector, vom suma vectorii cuvintelor.
- La final, vom alege propoziții din mulțimea sortată de TextRank până când depășim numărul de caractere dorit în sumar.

3. Etapa de extragere

- Algoritmul de extragere are la bază observația că toate paginile de pe un anumit website sunt legate printr-un meniu, iar acesta formează un subgraf complet.



4. Etapa de filtrare

- Variantă modificată a metodei din lucrarea [2], metodă ce calculează anumiți factori la nivel de nod.
- Luăm în considerare doar etichetele ce au conținut text în mod uzual: <div>, <table>, , <p>, <td>.
- Primul factor: densitatea textului.

$$\text{Numărul de linii} > 1: \frac{T'(b)}{(L(b)-1)*maxLen} \quad \text{Numarul de linii} == 1: \frac{T(b)}{maxLen}$$

Unde

- $T'(b)$ reprezintă numărul de caractere fără ultima linie.
- $L(b)$ este numărul de linii din nodul curent.
- $maxLen$ reprezintă numărul maxim de caractere întâlnite într-o linie din pagină.

4. Etapa de filtrare

- Al doilea factor: densitatea de URL-uri. $\frac{Ta(b)}{T(b)}$
- $Ta(b)$ este numărul de caractere prezente în nodul curent ce fac parte dintr-o etichetă de tip $\langle a \rangle$, folosită pentru URL-uri. $T(b)$ este numărul total de caractere din nodul b .
- Într-un final, Se elimină noduri pe baza unor valori de prag pentru densitatea textului și cea de URL-uri. Autorii au folosit valori de prag pentru cei doi factori.
- Valorile de prag fixe au fost înlocuite, acestea calculându-se la rulare drept media aritmetică a tuturor densităților de text, respectiv URL, din mulțimea de noduri.

4. Etapa de filtrare

- Textul de la pasul curent se împarte în propoziții și apoi în cuvinte, făcând de asemenea și analiza părților de vorbire. Putem elimina toate propozițiile mult prea scurte sau toate propozițiile ce nu conțin verbe.
- Ultimul nivel de filtrare este realizat cu ajutorul modelului Word2Vec ce conține peste 100 de miliarde de cuvinte. Vom elimina toate grupurile de trei cuvinte consecutive ce nu returnează un vector și vom repeta acest proces până la convergență.
- După ce filtrarea este complet realizată, putem extrage cuvintele cheie prin aplicarea TF-IDF asupra conținutului filtrat. Acestea vor fi folosite la căutarea după cuvinte cheie.

5. Etapa de filtrare

- Vom folosi distanța Levenshtein pentru a testa partea de extragere și filtrare:

	Conținut extras manual	Conținut extras algoritmic	Distanța Levenshtein	Distanța împărțită la lungime
<u>Articol HackerNews 1</u>	5600	5792	924	0.165
<u>Articol HackerNews 2</u>	3272	3639	663	0.202
<u>Articol HackerNews 3</u>	3861	4283	608	0.157
<u>Articol The Guardian Article 1</u>	6107	6902	1586	0.259
<u>Articol The Guardian Article 2</u>	5410	5940	850	0.155

5. Etapa de sumarizare

- Etapa de sumarizare este realizată prin execuția algoritmului TextRank folosind similaritățile pe bază de cosinus. Vom suma vectorii cuvintelor din fiecare propoziție și vom folosi produsul scalar pentru a obține cosinusul unghiului dintre cei doi vectori.
- Una dintre probleme tehnice pe care le ridică modelul Word2Vec este faptul că nu este fezabil să încarcăm în memoria RAM a procesului întregul model. În general, nu este necesar să încarcăm mai mult de câteva zeci de mii de cuvinte.

6. Concluzii

- Schimbarea metodei de obținere a șablonului scapă de rigiditatea inițială, dar pierde informația legată de structura de arbore a documentului HTML => algoritmul are mari probleme pe pagini complexe. S-ar putea concepe o abordare hibridă.
- Cele două niveluri adiționale de filtrare (NLP și Word2Vec) dau rezultate bune, validate de distanța Levenshtein.
- Sumarizarea este mai greu de evaluat științific, dar oferă rezultate bune. Posibile optimizări în lucrul cu Word2Vec: încărcarea întregului model și utilizarea acestuia pe mai multe thread-uri.

Multumesc!

Demo

Website simplu: <https://thehackernews.com/>

Website complex: <https://www.theguardian.com/international>

3. Etapa de extragere

Fie date următoarele 4 pagini:

1. [(<p>, 5), (<a>, 2), (<s>, 2), (
, 2), (<script>, 1)].
2. [(<p>, 4), (<a>, 2), (<s>, 1), (
, 2), (<script>, 0)].
3. [(<p>, 4), (<a>, 3), (<s>, 2), (
, 2), (<script>, 0)].
4. [(<p>, 5), (<a>, 2), (<s>, 2), (
, 2), (<script>, 1)].

Vom avea vectorii de frecvență:

- <p> - [5, 4, 4, 5] – deviația standard 0.57.
- <a> - [2, 2, 3, 2] – deviația standard 0.50.
- <s> - [2, 1, 2, 2] – deviația standard 0.50.
-
 - [2, 2, 2, 2] – deviația standard 0.00.
- <script> - [1, 0, 0, 1] – deviația standard 0.57.

Media aritmetică a deviațiilor standard este 0.428. Dacă această valoare este corespunzătoare vom avea:

- <p> - [4, 4, 5, 5] – mediana 4.5, dar vom folosi partea întreagă 4.
- <a> - [2, 2, 2, 3] – mediana 2.
- <s> - [1, 2, 2, 2] – mediana 2.
-
 - [2, 2, 2, 2] – mediana 2.
- <script> - [0, 0, 1, 1] – mediana 0.5, dar vom folosi partea întreagă 0.

Șablonul considerat de noi va fi vectorul de frecvență:

[(<p>, 4), (<a>, 2), (<s>, 2), (
, 2), (<script>, 0)]

7. Bibliografie

1. Julian Alarte, David Insa, Josep Silva, Salvador Tamarit: Web Template Extraction Based on Hyperlink Analysis. S. Escobar (Ed.): XIV Jornadas sobre Programación Y Lenguajes, PROLE 2014, Revised Selected Papers EPTCS 173, 2015, pp. 16–26.
2. Qingtang Liu, Mingbo Shao, Linjing Wu, Gang Zhao, and Guilin Fan: Main Content Extraction from Web Pages Based on Node Characteristics. Journal of Computing Science and Engineering, Vol. 11, No. 2, Iunie 2017, pp. 39-48.
3. https://miro.medium.com/max/3902/1*hELIVp9hmZbDZVFstS61pg.png
4. <https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/assets/2b4a7a82-ad4c-4b2a-b808-e423a334de6f.png>
5. <https://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/PageRanks-Example.svg/1200px-PageRanks-Example.svg.png>

Github: <https://github.com/CristianNCC/NLPWB-PresentationBuild>

<https://github.com/CristianNCC/NLP-WebScraper>