# Tema 3 IA - Sumarizarea documentelor

Mihai Trăscău

2020

Versiunea 1.0

## 1 Descrierea problemei

Sumarizarea unui text presupune generarea unei forme restrânse a acestuia păstrând însă informațiile cheie și înțelesul textului inițial, sub forma unui abstract. Procedura este adesea utilizată în aplicații de clasificare a textelor, în sistemele automate de regăsire a răspunsurilor la întrebări, în generarea de abstracte sau titluri pentru știri, etc. Identificăm două moduri în care putem realiza sumarizarea:

- *extractiv* - propozițiile cheie din text sunt identificate și apoi selectate (copiate) pentru a face parte din abstract
- *abstractiv* - textul este interpretat iar abstractul este generat prin metode de prelucrare a limbajului natural astfel încât să descrie în mod fluent și coerent informația importantă

În temă veți implementa soluții de **sumarizare extractivă de știri** în limba engleză. În plus, veți implementa și un **clasificator de știri** care va încadra o știre dată într-una dintre clasele $c_k \in C = \{business, entertainment, politics, sport, tech\}$.

## 2 Naive Bayes

Atât clasificarea cât și sumarizarea vor fi efectuate folosind **Naive Bayes**. Aceasta este o metodă statistică inductivă care se bazează pe Teorema lui Bayes, exprimată ca o relație între probabilitate *a priori* și cea *posterioară* a unei ipoteze. Astfel, pentru clasificare avem:

$$P(C = c_k|\mathbf{x}) = \frac{P(\mathbf{x}|C = c_k) \cdot P(C = c_k)}{P(\mathbf{x})} \tag{1}$$

unde:

- $P(C = c_k)$ reprezintă probabilitatea *apriori* a clasei $c_K$

- $P(C = c_k|\mathbf{x})$ reprezintă probabilitatea *a posteriori* a clasei $c_K$ după ce $\mathbf{x}$ este observat
- $P(\mathbf{x}|C = c_k)$ reprezintă probabilitatea ca $\mathbf{x}$ să facă parte din clasa $c_K$ (*verosimilitate*, eng. *likelihood*)
- $P(\mathbf{x})$ reprezintă probabilitatea observațiilor (eng. *evidence*)

Clasificatoarele Naive Bayes se bazează pe conceptul MAP (eng. *Maximum A Posteriori*), alegând clasa cu probabilitatea maximă:

$$c_{MAP} = \arg\max_{c \in \mathcal{C}} P(\mathbf{x}|c) \cdot P(c) \tag{2}$$

Știm că intrările $\mathbf{x}$ pentru care trebuie să găsim clasa sunt texte formate din $N$ cuvinte (atribute), adică $\mathbf{x} = \{x_1, x_2, ..., x_N\}$. În algoritmul Naive Bayes facem presupunerea simplificatoare prin care considerăm toate atributele $x_i$ ca fiind *condițional independente* când clasa $c$ este observată. Astfel, putem scrie că:

$$P(\mathbf{x}|c) = \prod_{i=1}^{N} P(x_i|c) \tag{3}$$

Astfel, din ecuațiile (2) și (3) avem că:

$$c_{MAP} = \arg\max_{c \in \mathcal{C}} P(c) \cdot \prod_{i=1}^{N} P(x_i|c) \tag{4}$$

iar pentru a evita lucrul cu valori foarte mici care pot duce la erori de calcul (eng. *underflow*), vom logaritma expresia:

$$c_{MAP} = \arg\max_{c \in \mathcal{C}} \log(P(c)) \cdot \sum_{i=1}^{N} \log(P(x_i|c)) \tag{5}$$

# 3 Clasificarea documentelor

Având la dispoziție un set de date de antrenare, vom folosi textele din acesta pentru a determina care sunt parametrii modelului nostru. Pentru clasificare, va fi nevoie să estimăm care sunt valorile probabilităților descrise în secțiunea 2:

$$P(C = c_k) = \frac{\text{număr documente în clasa } c_k}{\text{număr total de documente}} \tag{6}$$

$$P(x_i|C = c_k) = \frac{\text{număr de apariții ale lui } x_i \text{ în documente din clasa } c_k}{\text{număr total de cuvinte în documentele din clasa } c_k} \tag{7}$$

Din ecuația (7) reiese că pentru cuvinte rare, care apar în textele testate dar nu și în cele cu care antrenăm modelul, probabilitatea de verosimilitate va fi 0,

ceea ce va determina $c_{MAP} = 0$ din înmulțirea de termeni (în timp ce $log(0)$ nu este definit). Pentru a evita acest lucru este de dorit ca aceste cuvinte să aibă o valoare diferită de zero, oricât de mică ar fi ea. Prin urmare, vom folosi conceptul de *netezire Laplace* (eng. *Laplace smoothing*):

$$P(x_i|C = c_k) = \frac{\text{număr de apariții ale lui } x_i \text{ în documente din clasa } c_k + \alpha}{\text{număr total de cuvinte în documentele din clasa } c_k + |Voc| + \alpha} \tag{8}$$

unde $\alpha$ este parametrul de netezire (deseori în practică găsim $\alpha = 1$) iar $|Voc|$ este dimensiunea vocabularului din setul de date.

Evaluarea performanțelor modelului se poate face calculând *precizia* și *regăsirea* (eng. *recall*) acestuia [1, 2]. Precizia reprezintă numărul de predicții corecte ale modelului din numărul total de predicții făcute. Valoarea de regăsire a informației se calculează ca fiind numărul de predicții corecte făcute de model pentru o anumită clasă din numărul total de indivizi (în cazul nostru documente) din clasa respectivă.

# 4 Procesări specifice textelor

În vederea obținerii unor rezultate mai bune (și mai relevante) este necesar să efectuăm anumite procesări asupra textelor cu care operăm. Pentru o implementare mai facilă, este recomandat ca în temă să utilizați biblioteci destinate prelucrării în limbaj natural, precum **nltk** sau **spacy**.

**Tokenizarea** este procesul de spargere a textului în cuvinte [3, 4]. Un proces similar este și cel de împărțire a textului în propoziții (eng. *sentencizer*) [5, 6].

**Eliminarea cuvintelor neinformative** (eng. *stop-words*) este o tehnică prin care se încearcă reducerea influenței statistice pe care o au anumite cuvinte considerate puțin importante din punct de vedere semantic. Astfel, documentele sunt filtrate pentru eliminarea cuvintelor precum: *to*, *at*, *from*, *and*, *by*, etc.

**Lematizarea** [7, 8] este procesul prin care toate formele flexionare ale unui cuvânt sunt grupate sub aceeași entitate. De exemplu, în limba engleză, cuvintele *walk*, *walkes*, *walked*, *walking*, vor fi toate legate la entitatea *walk*. În acest fel, referențiem împreună forme ușor diferite dar cu semnatică similară crescând relevanța lor statistică.

**Utilizarea n-gramelor** este o tehnică prin care se dorește realizarea unor calcule statistice care să țină cont și de contextul local al unui cuvânt, dat de cuvintele vecine. Astfel, dacă modelele cu unigrame lucrează cu cuvinte individiuale, modelele cu bigrame utilizează toate perechile de cuvinte adiacente din documente.

# 5 Sumarizarea documentelor

În vederea sumarizării documentelor urmărim aceleași ecuații din secțiunea 2 a căror implementare va fi similară cu cele din secțiunea 3. Astfel, obiectul clasificării (intrarea) nu mai este un document, ci o propoziție dintr-un document,

iar în loc de cele 5 clase din problema de clasificare în sumarizare ne interesează doar dacă propoziția respectivă va face parte sau nu din abstract. Întrucât și propozițiile sunt formate tot din cuvinte, probabilitățile de verosimilitate se pot calcula similar cu cele din problema clasificării.

Evaluarea performanțelor este asemănătoare celei din problema clasificării, utilizându-se scorul *ROUGE-N* [9].

# 6 Cerințe

Pentru temă veți implementa un clasificator de știri și o metodă de sumarizare bazată pe algoritmul Naive Bayes. Setul de date cu care veți lucra este *BBC News Summaries*, pe care îl veți găsi în arhiva temei. Acesta cuprinde știri din 5 categorii (*business*, *entertainment*, *politics*, *sport*, *tech*). Setul de date este împărțit în 2 directoare, unul cu știrile originale iar celălalt cu sumarizările acestora. În arhivă găsiți și un document care conține printre cele mai comune *stop-words* din limba engleză. Setul de date îl veți împărți aleator în două subseturi: unul cu date de antrenare (75%) și unul cu date de testare (25%). Evident, parametrii modelului vor fi determinați pe baza subsetului de antrenare, în timp ce performanțele le veți raporta bazat pe subsetul de testare.

**Cerința 1 (4p)** Implementați încărcarea setului de date în memorie, grupând documentele după clasa lor. Fiecărui document trebuie să îi asociați și sumarizarea aferentă. Pentru documentele încărcate aplicați pașii de tokenizare (cuvinte și propoziții), eliminare cuvinte neinformative (eng. *stop-words*) și lematizare. Reprezentarea datelor în implementare nu este impusă, însă găsirea unor forme optimizate este încurajată.

**Cerința 2 (1.5p)** Implementați algoritmul Naive Bayes pentru clasificarea știrilor din setul de date în cele 5 clase disponibile. Calculați valorile de precizie și regăsire a informației (eng. *recall*) pentru variantele cu sau fără eliminarea *stop-words* (fără lematizare), cu lematizare (dar *stop-words* eliminate).

**Cerința 3 (3p)** Implementați algoritmul Naive Bayes pentru sumarizarea știrilor din setul de date. Calculați valorile ROUGE-N pentru variantele cu sau fără eliminarea *stop-words* (fără lematizare), cu lematizare (dar *stop-words* eliminate). Toate aceste rezultate trebuie calculate folosind, pe rând, unigrame apoi bigrame.

**Cerința 4 (1.5p)** Redactați un raport cu rezultatele experimentale în care să includeți, atât pentru clasificare cât și pentru sumarizare, grafice comparative cu rezultatele metodelor testate. Pentru clasificare, generați și includeți în raport o matrice de confuzie [10] care să descrie modul în care documente sunt clasificate în funcție de clasa din care provin.

**Bonus (2p)** Completați raportul de la Cerința 4 cu rezultatele testelor obținute pentru clasificare și pentru sumarizare folosind procedeul *5-fold cross-validation* [11]. Practic, veți împărți setul de date în 5 părți egale (pe cât posibil). La o rulare a testelor, una din părți este selectată ca fiind pentru testare, iar celelalte 4 vor fi utilizate pentru antrenare. Se vor realiza astfel 5 rulări diferite pentru ca fiecare din cele 5 părți să fie utilizată, o dată, pentru

testare. Rezultatul final pentru valorile căutate (de exemplu, precizie) este dat de media și deviația standard a celor 5 valori obținute.

# Referințe


[1] https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_652.

[2] https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.

[3] www.nltk.org/api/nltk.tokenize.html.

[4] https://spacy.io/api/tokenizer.

[5] http://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.punkt.

[6] https://spacy.io/api/sentencizer.

[7] http://www.nltk.org/api/nltk.stem.html#module-nltk.stem.wordnet.

[8] https://spacy.io/api/lemmatizer.

[9] http://www.ccs.neu.edu/home/vip/teach/DMcourse/5_topicmodel_summ/notes_slides/What-is-ROUGE.pdf.

[10] https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html.

[11] https://scikit-learn.org/stable/modules/cross_validation.html.