Capitole Speciale de Informatică

Curs 2: Determinarea vocabularului de termeni şi a listelor de postări

4 octombrie 2018

Reamintim că listele de indecşi inversaţi se construiesc în 4 paşi:

1. Se colectează o mulţime D de documente care se vor indexa.
2. Se tokenizează textul din documentele colecţiei.
3. Tokenurile se preprocesează lingvistic.
4. Se indexează documentele în care apare fiecare termen.

Scopul acestui curs este să indice

- cum se determină unităţile de măsură a documentelor, şi secvenţele de caractere pentru conţinutul lor.
- aspectele importante ale tokenizării şi preprocesării lingvistice.
- implementarea listelor de postări şi a listelor extinse de postări.
- structuri de date utile pentru interogări ce includ fraze şi operatori de proximitate.

1 Determinarea documentelor şi decodificarea secvenţei de caractere

De obicei, documentele în format electronic sunt secvenţe de octeţi stocate în un fişier sau pe un browser web. Se folosesc mai multe metode de codificare/decodificare a secvenţelor de caractere în secvenţe de caractere: ASCII, Unicode UTF-8, UTF-16, etc. Preoprocesarea acestor documente presupune:

1. Determinarea codificării corecte, urmată de decodificare. Determinarea codificării dorecte este o problemă de clasificare în machine learning, care poate fi rezolvată în mai multe feluri: cu metode euristice, pe baza unei selecţii a utilizatorului, sau pe baza metadatelor asociate cu documentul respectiv.

2. Uneori, secvenţa de caractere produsă prin decodificare mai trebuie preprocesată. De exemplu, documentele XML conţin entităţi care se înlocuiesc cu carctere speciale. De exemplu, `&` se înlocuieşte cu `&`.

   In general, produsele software comerciale trebuie să ştie să recunoască şi să decodifice un număr mare de tipuri de documente şi codificări. De obicei, decodificarea în un şir de caractere se face cu biblioteci software achiziţionate cu licenţă în acest scop. Aceste biblioteci ştiu şi cum să decodifice texte scrise în arabă sau japoneză, care nu sunt simple şiruri de caractere scrise de la stânga la dreapta.

3. Să se decidă ce unitate de măsură a documentelor (engl. *document unit*) să folosească. Adeseori, un document no corespunde unui fişier: un ficşier `zip` poate arhiva mai multe documente; mesajele din mail-boxul unui utilizator sunt toate stocate în un singur fişier; unele programe (precum `latex2html`) descompun un document în mai multe fişiere separate.

   Prin urmare, se pune problema determinării *granularităţii*: cât de mare să fie un document; cum definim şi cum determinăm o unitate de măsură pentru documente.

2 Determinarea vocabularului de termeni

Secvenţa de caractere produsă prin decodificare se desparte în tokenuri. De obicei, tokenurile se sunt separate prin semne de punctuaţie şi secvenţe de caractere speciale (eng. *whitespace*).

De exemplu, tokenizarea textului "Friends, Romans, Countrymen, lend me your ears." produce secvenţa de tokenuri

Friends	Romans	Countrymen	lend	me	your	ears

Apoi, se rulează un proces de normalizare care extrage o secvenţă de **termeni** din lista de tokenuri. Termenii vor fi elementele incluse în dicţionarul sistemului de IR.

Tokenizarea este o problemu a dificilă: cum definim şi cum determinăm tokenurile? De exemplu, cum tokenizarea frazei

> "Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing."

se poate face corect doar după ce se detectează limbajul în care ese scris textul. O tehnică foarte eficientă de **detecţie a limbajului** este cea bazată pe clasificatori ce utilizează subsecvenţe scurte de caractere.

Alte probleme complexe rezolvate de algoritmii avansaţi de tokenizare sunt:

- interpretarea corectă a liniuţei de subliniere. De exemplu *co-education* şi *Hewlett-Packard* sunt considerate un singur token.
- recunoaşterea grupurilor de cuvinte care constituie un singur token. De exemplu *Los Angeles* sau *New York University*.

- Segmentarea unor cuvinte compuse lungi. De exemplu, cuvântul german "Lebensversicherungsgesellschaftsangestellter" înseamnă "angajat al unei companii de asigurare pe viaţă" şi poate fi descompus în mai mute tokenuri.
- Eliminarea unor cuvinte de legătură folosite frecvent. Cuvinte precum "şi" sau articole precum "al" se omit din lista de tokenuri.
- ...

3 Normalizarea tokenurilor

Normalizarea este procesul de împărţire a tokenurilor în clase de echivalenţă, adică mulţimi de tokenuri care au acelaşi înţeles sau înţelesuri foarte asemănătoare. De exemplu, tokenurile `maşinã` şi `automobil` sunt echivalente Prin urmare, când căutam documente despre `maşinã`, vrem să găsim şi documente despre `automobil`.

O soluţie la această problemă este să reprezentăm toţi termenii din o clasă de echivalenţă cu un singur termen. Înlocuirea unui termen cu reprezentantul clasei sale de echivalenţă se numeşte **normalizare**.

Altă soluţie este să reţinem care termeni sunt sinonimi. De obicei, acest lucru se realizează în 2 feluri:

1. Se indexează toate tokenurile nenormalizate şi se asociază o *listă de expansiune* pentru toţi termenii admişi în o cerere de informare. De exemplu, `maşinã` se va putea înlocui cu `maşinã OR automobil`.
2. Expansiunea tokenurilor se face pe durata indexării. De exemplu, un document care conţine `maşinã` se indexează şi pentru tokenul `automobil`.

Unele probleme de normalizare trebuie să ţină cont de limbajul din care provin tokenurile:

- În engleză, accentele şi diacriticile au de obicei un rol marginal ⇒ normalizarea se poate face eliminând diacriticile: "cliche" în loc de "cliché", "naive" în loc de "naïve."

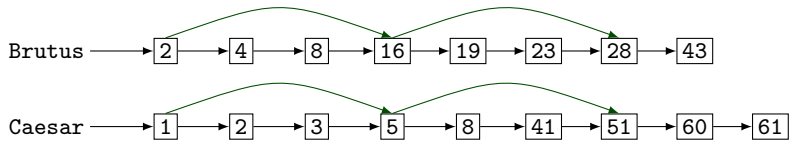
  În alte limbi, precum spaniola, un accent poate schima complet sensul unui cuvânt: "peña" înseamnă "'faleză", iar "pena" înseamnă "durere."

...

4 Liste extinse de postări

În cursul 1 am vazut că intersecţia a 2 liste de postări de lungimi m şi n se poate face în timp $O(m+n)$. Dacă dorim timpi şi mai scurţi (sub-liniari) de calcul al intersecţiei, putem folosi liste cu pointeri de ocolire (engl. **skip lists**):

- pointerii de ocolire sunt "scurtături" efective care ne permit să evităm să procesăm porţiuni mari din listele de postări care nu apar în rezultatul intersecţiei. Figura de mai jos ilustrează 2 liste de postări cu pointeri de ocolire:



Apar 2 probleme noi: (1) unde se plasează pointerii de ocolire, şi (2) cum se poate face intersecţie rapidă cu pointeri de ocolire.

Algoritmul de intersecţie cu pointeri de ocolire se poate defini în mai multe feluri. O posibilitate este ilustrată mai jos:

```
IntersectCuOcoliri(p1, p2)
 1 raspuns := ⟨⟩
 2 while p1 ≠ Nil and p1 ≠ Nil do
 3       if docID(p1) = docId(p2)
 4          adaugă docId(p1) la raspuns
 5          p1 := next(p1)
 6          p2 := next(p2)
 7       else if docID(p1) < docId(p2)
 8          if hasSkip(p1) and (docID(skip(p1)) ≤ docId(p2))
 9             while hasSkip(p1) and (docID(skip(p1)) ≤ docId(p2)) do
10                   p1 := skip(p1)
11          else p1 := next(p1)
12       else if hasSkip(p2) and (docID(skip(p2)) ≤ docId(p1))
13          while hasSkip(p2) and (docID(skip(p2)) ≤ docId(p1)) do
14                p2 := skip(p2)
15          else p2 := next(p2)
16 return raspuns
```

Observaţii:

- Pointerii de ocolire sunt folositori pentru a face mai rapida operaţia de intersecţie; nu pot fi folosiţi pentru operaţia de reuniune.
- Mai mulţi pointeri de ocorire implică
  - rute de ocolire mai scurte
  - mai multe teste de ocolire ş mai mult spaţiu ocupat de pointerii de ocolire.
- O euristică simplă care funcţionează bine în practică este ca pentru liste de postări de lungime P să alocăm $\sqrt{P}$ pointeri de ocolire care sar peste căte $\sqrt{P}$ postări.

O altă extensie utilă a listelor simple de postări sunt **listele de indecşi poziţionali**. O listă de indecşi poziţionali pentru un termen t reţine mai multe informaţii:

- frecvenţa lui t în colecţia de documente, adică în câte documente apare t.
- pentru fiecare document cu identificatorul $docID$, o postare de forma

$$docId, n : \langle pos_1, pos_2, \ldots \rangle$$

  unde

  - pos_1, pos_2, ... sunt poziţiile (sortate crescător) unde apare termenul t in documentul cu identificatorul $docId$.
  - n este numărul de aparţii al lui t in documentul cu identificatorul $docId$.

De exemplu, listele de indecşi poziţionali ai termenilor `to` şi `be` în o colecţie de documente ar putea arăta astfel:

`to`,993427:
⟨1,6:⟨7, 18, 33, 72, 86, 231⟩;
2,5:⟨1, 17, 74, 222, 255⟩;
4,5:⟨8, 16, 190, 429, 433⟩;
...⟩
`be`,178239:
⟨1,2:⟨17, 25⟩;
4,5:⟨17, 191, 291, 430, 434⟩;
5,3:⟨14, 19, 101⟩;
...⟩

Listele de indeşi poziţionali permit găsirea documentelor care conţin fraze, adică secvenţe de termeni.

De exemplu, găsirea documentelor care conţin "`to be`" poate decurge astfel: se caută poziţiile în lista de indecşi poziţionali ai lui `be` care sunt imediat după o poziţie în lista de indecşi poziţionali ai lui `to` în acelaşi document. De pildă, pentru listele ilustrate mai sus, rezultatul căutării va fi

`to`:⟨..., 4 : ⟨..., 429, 433⟩, ...⟩
`be`:⟨..., 4 : ⟨..., 430, 434⟩, ...⟩

Un algoritm mai general, care găseşte toate perechile de poziţii din 2 liste de indecşi poziţionali care sunt la cel mult k cuvinte distanţă una de cealaltă, este ilustrat în figura 1.

```
PositionalIntersect(p1, p2, k)
 1 raspuns := ⟨⟩
 2 while p1 ≠ Nil and p1 ≠ Nil do
 3       if docID(p1) = docId(p2)
 4          l := ⟨⟩
 5          pp1 := pozitii(p1)
 6          pp2 := pozitii(p2)
 7          while pp1 ≠ Nil do
 8                  while pp2 ≠ Nil do
 9                          if |pos(pp1) − pos(pp2)| ≤ k
10                             adaugă pos(pp2) la l
11                          else if pos(pp2) > pos(pp1)
12                             break
13                          pp2 := next(pp2)
14                  while l ≠ ⟨⟩ and |l[0] − pos(pp1)| > k do
15                          şterge l[0] din l
16                  for each ps ∈ l do
17                          adaugă ⟨docId(p1), pos(pp1), ps⟩ la raspuns
18                          pp1 := next(pp1)
19          p1 := next(p1)
20          p2 := next(p2)
21       else if docID(p1) < docID(p2)
22               p1 := next(p1)
23       else p2 := next(p2)
24 return raspuns
```

Figure 1: Algoritm de intersecţie de proximitate dintre 2 liste extinse de postări.

Bibliografie

1. Capitolul 2 din
   Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze: *An Introduction to Information Retrieval*. Ediţie online (c) 2009 Cambridge UP.
   `http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf`