LUŠČENJE PODATKOV S SKRITIMI MARKOVSKIMI MODELI

Mitja Luštrek
Odsek za inteligentne sisteme
Institut Jožef Stefan
Jamova 39, 1000 Ljubljana, Slovenija
Telefon: +386 1 4773380, telefaks: +386 1 4251038
E-pošta: mitja.lustrek@ijs.si

POVZETEK

Prispevek obravnava uporabo skritih markovskih modelov za luščenje podatkov – iskanje določenih podatkov v besedilu. Iz besedil z označenimi iskanimi zaporedji besed se je mogoče naučiti model, ki posploši učna besedila in zna za nova oceniti, katera zaporedja besed so iskana. Predstavljen je sistem Marko, ki je uporabljen za ugotavljanje, kakšni parametri skritega markovskega modela se dobro obnesejo pri luščenju podatkov.

1. UVOD

Luščenje podatkov (*information extraction*) je postopek za iskanje določenih podatkov v besedilu, npr. za iskanje govornika, kraja ali časa v najavi seminarja. Področje se je razcvetelo v 90. letih prejšnjega stoletja, v dobršni meri po zaslugi Message Understanding Conferences [7], ki so se pod okriljem ameriške agencije DARPA odvijale v letih 1987–1998. Na teh konferencah so organizatorji pripravili naloge luščenja podatkov, udeleženci pa so primerjali, kako se na zadanih nalogah obnesejo njihovi sistemi in kako se računalniški programi obnesejo napram ljudem.

Za luščenje podatkov se uporabljajo mnoge metode, med drugim induciranje ovojnic [8][4] ter različni načini učenja pravil in vzorcev [2][12]. Skriti markovski modeli se že dolgo rabijo pri prepoznavanju govora [10], za luščenje podatkov so jih leta 1997 uporabili Bikel et al. [1] in Leek [9]. Morda največ sta se z njimi ukvarjala Freitag in McCallum [6][5].

V poglavju 2 so predstavljeni skriti markovski modeli in razloženo je, kako se uporabljajo za luščenje podatkov. V poglavju 3 je popisan sistem za luščenje podatkov Marko ter kako na njegovo delovanje vplivajo nekateri parametri in kako se obnesejo različna ogrodja modela. Poglavje Tabela 6 prispevek sklene in poda nekaj možnosti za nadaljnje delo.

2. SKRITI MARKOVSKI MODELI

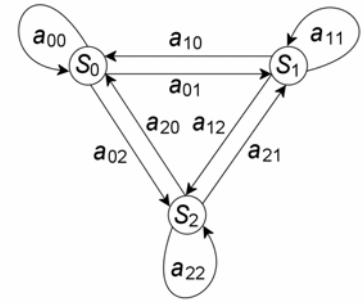
Zamislimo si sistem, ki je v vsakem trenutku v enem od N stanj iz množice $S = \{S_0, S_1, ..., S_{N-1}\}$. Ob časih $t = 0, 1, ..., T - 1$ prehaja iz enega stanja v drugo. Stanje v času t označimo s q_t. Če je stanje v času t odvisno le od stanja v času $t - 1$ in če je zveza med njima neodvisna od t, je tak sistem markovski model. Določa ga matrika prehodov med stanji $A = [\, a_{ij} \,]$, kjer je a_{ij} opisan z enačbo (1).

$$a_{ij} = \mathrm{P}(q_t = S_j \mid q_{t-1} = S_i);\ 0 \le i, j \le N-1 \tag{1}$$

Začetne verjetnosti stanj so opisane z enačbo (2).

$$\pi_j = \mathrm{P}(q_0 = S_j);\ 0 \le j \le N-1 \tag{2}$$

Primer markovskega modela je na sliki 1.

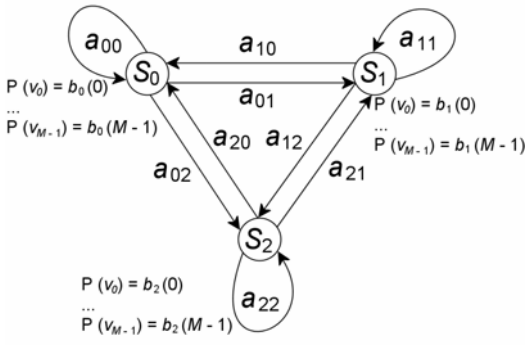


Slika 1. Markovski model

Pri običajnih markovskih modelih je stanje opazovalcu poznano. Splošnejši so skriti markovski modeli, kjer opazovalec pozna le neko verjetnostno funkcijo stanja, samo stanje pa mu je skrito. Ta verjetnostna funkcija je izražena s izhodnimi simboli abecede $V = \{v_1, v_2, ..., v_{M-1}\}$, ki jih model oddaja. Verjetnosti oddajanja simbolov določa matrika $B = [\, b_j(k) \,]$, kjer je $b_j(k)$ opisan z enačbo (3).

$$b_j(k) = \mathrm{P}(v_k \text{ v času } t \mid q_t = S_j);\quad 0 \le j \le N-1,\ 0 \le k \le M-1 \tag{3}$$

Primer skritega markovskega modela je na sliki 2.



Slika 2. Skriti markovski model

Zaporedje besed v besedilu lahko predstavimo kot zaporedje izhodnih simbolov skritega markovskega modela. Pri luščenju podatkov moramo rešiti dva problema.

1. S pomočjo učnih besedil moramo sestaviti model. V jeziku skritih markovskih modelov to pomeni, da moramo iz zaporedja izhodnih simbolov $O = O_0\ O_1\ ...\ O_{T-1}$ določiti model $\lambda = (A, B, \pi)$, tako da maksimiziramo $P(O \mid \lambda)$. Ogrodje modela pripravimo vnaprej: določimo število stanj, dovoljene prehode med njimi in katera stanja so ciljna. V ciljnih stanjih model oddaja simbole, ki sestavljajo zaporedja besed, ki jih želimo izluščiti iz novih besedil. Optimalne parametre modela se da določiti le približno, z iterativnimi algoritmi, kakršen je Baum-Welchev.

2. S pomočjo modela moramo iz novih besedil izluščiti iskana zaporedja besed. V jeziku skritih markovskih modelov to pomeni, da moramo za dani model λ in zaporedje izhodnih simbolov O določiti zaporedje stanj $Q = q_0\ q_1\ ...\ q_{T-1}$, tako da maksimiziramo $P(Q \mid O, \lambda)$. Tisti simboli, ki so oddani v ciljnih stanjih, sestavljajo iskana zaporedja besed. Za to se uporablja Viterbijev algoritem.

2.1. Baum-Welchev algoritem

Najprej določimo začetne vrednosti A, B in π. Marko na začetku vsem prehodom iz vozlišča, ki jih ogrodje modela dopušča, pripiše enako verjetnost. Prav tako pripiše v vsakem vozlišču enako verjetnost vsem izhodnim simbolom. Ker je začetno stanje (naj bo to S_0) določeno že v ogrodju modela, velja $\pi_0 = 1$ in $\pi_{1,\,2,\,...,\,N-1} = 0$.

Naj bo $O = O_0\ O_1\ ...\ O_{T-1}$ učno zaporedje izhodnih simbolov. Definirajmo verjetnost, da je model, ki odda to zaporedje simbolov, v času t v stanju S_i in v času $t + 1$ v stanju S_j – izračuna se po enačbi (4).

$$\xi_{ij}(t) = \mathrm{P}(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) = \\ = \frac{\alpha_i(t) a_{ij} b_j(O_{t+1}) \beta_j(t+1)}{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \alpha_i(t) a_{ij} b_j(O_{t+1}) \beta_j(t+1)} \tag{4}$$

$\alpha_i\ (t)$ je verjetnost, da model do časa t odda zaporedje simbolov $O_0\ O_1\ ...\ O_t$ in se znajde v stanju S_i. $\beta_j\ (t + 1)$ pa je verjetnost, da model od časa $t + 1$ odda zaporedje simbolov $O_{t+2}\ O_{t+3}\ ...\ O_{T-1}$, pri čemer je v času $t + 1$ v stanju S_j.

Vrednosti $\alpha_i\ (t)$ izračunamo rekurzivno po enačbah (5).

$$\alpha_j(0) = \pi_j b_j(O_0); 0 \le j \le N-1$$

$$\alpha_j(t+1) = \begin{cases} \left(\sum_{i=0}^{N-1} \alpha_i(t) a_{ij} \right) b_j(O_{t+1}); L(S_j) = L(O_{t+1}) \\ 0; \text{ sicer} \end{cases} \tag{5}$$

$$0 \le t \le T-2,\ 0 \le j \le N-1$$

$L\ (x)$ je funkcija, ki lahko zavzame dve vrednosti: eno, če je stanje x ciljno ali če je simbol x ciljen, in drugo, če ni. Z njeno uporabo dosežemo, da se ciljni izhodni simboli oddajajo le v ciljnih stanjih.

Na podoben način kot $\alpha_i\ (t)$ izračunamo tudi vrednosti $\beta_i\ (t)$. Ker je v Marku tudi končno stanje (naj bo to S_{N-1}) določeno že z ogrodjem modela, velja $\beta_{N-1}\ (T - 1) = 1$ in $\beta_{0,\,1,\,...,\,N-2}\ (T - 1) = 0$. Preostale β izračunamo rekurzivno po enačbi (6).

$$\beta_i(t) = \begin{cases} \sum_{j=0}^{N-1} a_{ij} b_j(O_{t+1}) \beta_j(t+1); L(S_j) = L(O_{t+1}) \\ 0; \text{ sicer} \end{cases} \tag{6}$$

$$0 \le t \le T-2,\ 0 \le i \le N-1$$

Števec enačbe (4) je tako produkt verjetnosti, da model odda zaporedje $O_0\ O_1\ ...\ O_t$ in se znajde v stanju S_i, da preide v stanje S_j in pri tem odda simbol O_{t+1} ter da začenši v stanju S_j odda zaporedje $O_{t+2}\ O_{t+3}\ ...\ O_{T-1}$. To pa moramo deliti z verjetnostjo, da model sploh odda zaporedje O, kar se izrazi kot verjetnost, da do kateregakoli stanja odda zaporedje $O_0\ O_1\ ...\ O_t$, da preide v katerokoli drugo stanje in pri tem odda simbol O_{t+1} ter da nato odda zaporedje $O_{t+2}\ O_{t+3}\ ...\ O_{T-1}$.

Definirajmo še verjetnost, da je model v času t v stanju S_i – podana je z enačbo (7).

$$\gamma_i(t) = \sum_{j=0}^{N-1} \xi_{ij}(t) \tag{7}$$

Razmerje vsote $\xi_{ij}\ (t)$ po vseh t in vsote $\gamma_i\ (t)$ po vseh t je enako pričakovanemu razmerju števila prehodov iz S_i v S_j in iz S_i v katerokoli drugo stanje, to pa je enako popravljenemu a_{ij}, kot kaže enačba (8).

$$a_{ij}' = \frac{\sum_{t=0}^{T-2} \xi_{ij}(t)}{\sum_{t=0}^{T-2} \gamma_i(t)} \tag{8}$$

Popravljeno vrednost $b_j\ (k)$ izračunamo kot pričakovano razmerje obiskov stanja S_j, kadar je oddan simbol v_k, in vseh obiskov stanja S_j, kot kaže enačba (9).

$$b_j(k)' = \frac{\sum_{\substack{t=0 \\ O_t = v_k}}^{T-1} \gamma_i(t)}{\sum_{t=0}^{T-1} \gamma_i(t)} \tag{9}$$

Matriki [a_{ij}'] in [$b_j\ (k)'$] sta vhodna podatka naslednje iteracije algoritma. Algoritem se izvaja, dokler se vrednosti ne ustalijo.

2.2. Viterbijev algoritem

Naj bo $O = O_0\ O_1\ ...\ O_{T-1}$ zaporedje izhodnih simbolov iz katerega si prizadevamo izluščiti iskana zaporedja besed. Pri opisu algoritma si bomo pomagali z največjo verjetnostjo, da neko zaporedje stanj do časa t, ko je model v stanju S_i, odda zaporedje simbolov $O_0\ O_1\ ...\ O_{t-1}$, ki je definirana z enačbo (10).

$$\delta_j(t) = \max_{q_0,q_1,\ldots,q_{T-1}} \left[\mathrm{P}(q_0 q_1 \ldots q_t = S_i \mid O_0 O_1 \ldots O_t, \lambda)\right] \quad (10)$$

Vrednosti δ_i (t) računamo rekurzivno. V vsakem koraku moramo shraniti še argument, ki δ_i (t) maksimizira. Postopek opisujejo enačbe (11).

$$\begin{aligned} &\delta_j(0) = \pi_j b_j(O_0); 0 \le j \le N-1 \\ &\delta_j(t+1) = \max_{0\le i\le N-1}\left[\delta_i(t) a_{ij}\right] b_j(O_{t+1}) \\ &\psi_j(t+1) = \arg\max_{0\le i\le N-1}\left[\delta_i(t) a_{ij}\right], \\ &0 \le j \le N-1, 1 \le t \le T \end{aligned} \quad (11)$$

Stanja najverjetnejšega zaporedja se rekurzivno izračunajo po enačbah (12).

$$\begin{aligned} &q_T{}^* = \arg\max_{0\le i\le N-1}\left[\delta_i(T)\right] \\ &q_t{}^* = \psi_{q^*_{t+1}}(t+1); 0 \le t \le T-2 \end{aligned} \quad (12)$$

Iskana zaporedja besed sestavljajo tisti simboli O_t, za katere velja, da so stanja q_t* ciljna.

3. SISTEM MARKO

Za potrebe sistema za luščenje podatkov moramo prijeme iz poglavja 2 nekoliko prirediti. Vrednosti α_i (t) in β_i (t) v Baum-Welchovem algoritmu so produkti mnogih števil manjših od ena, zato pri velikih t postanejo manjše od natančnosti računalnika. To se zlahka reši, če jih množimo s primernimi vrednostmi, α_i (t) denimo s c (t) določenim z enačbo (13) (za β_i (t) velja podobno). Vrednosti c (t) se v enačbah (8) in (9) okrajšajo.

$$c(t) = \frac{1}{\sum_{i=0}^{N-1} \alpha_i(j)}; 0 \le t \le T-1 \quad (13)$$

V Viterbijevem algoritmu se pojavlja podoben problem, ki ga rešimo tako, da δ_j (t) definiramo z enačbo (14) in ustrezno popravimo enačbe (11). Na končni rezultat takšno logaritmiranje ne vpliva.

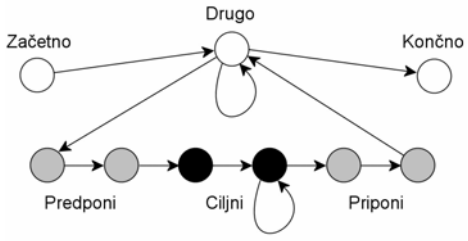
$$\delta_j(t) = \max_{q_0,q_1,\ldots,q_{T-1}} \left[\log \mathrm{P}(q_0 q_1 \ldots q_t = S_i \mid O_0 O_1 \ldots O_t, \lambda)\right] \quad (14)$$

Ker pri učenju navadno uporabimo več besedil in imamo torej opraviti z več zaporedji izhodnih simbolov, α_i (t), β_i (t), ξ_{ij} (t) in γ_i (t) izračunamo za vsakega posebej, nato pa v enačbah (8) in (9) v števcu in imenovalcu preprosto uporabimo vsoto po vseh zaporedjih.

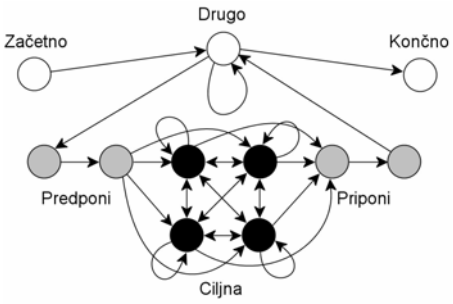
Naučeni modeli poznajo le besede, ki so nastopale v učnih besedilih, v besedilih, na katerih jih potem uporabljamo, pa seveda lahko nastopajo nove besede. Do neke mere sicer pomaga krnjenje (*stemming*), ki ga Marko pri angleških besedilih uporablja. Poleg tega Markovi modeli vsebujejo šest splošnih besed, ki se prilegajo vsaki besedi iz samih velikih črk, vsaki besedi z veliko začetnico, vsaki besedi iz samih malih črk, vsaki številki, vsakemu zaporedju ločil in vsakemu drugemu nizu. Verjetnost, da se v nekem stanju odda splošna beseda, je za vsako od njih sorazmerno številu običajnih besed, oddanih v tistem stanju, ki se ji prilegajo. Verjetnost najverjetnejša splošne besede je enaka verjetnosti najmanj verjetne (a še vedno možne) običajne besede pomnoženi s parametrom s. Verjetnosti oddajanja besed se zgladijo: Marko naučeni porazdelitvi prišteje enakomerno porazdelitev pomnoženo s parametrom e.

Marka sem preizkusil na zbirki 485 napovedi seminarjev na CMU [11], kjer so označeni govornik, kraj, začetek in konec. Za vsako od iskanih zaporedij besed se je Marko naučil svoj model. 10% napovedi je bilo uporabljenih za preizkušanje, za učenje pa v večini poizkusov (povsod, kjer ni izrecno navedeno drugače) zaradi večje hitrosti le 50%.

Izmed več preizkušenih ogrodij modela sta se najbolje izkazali *preprosto* na sliki 3 in *povezano* na sliki 4. Zaradi večje preglednost je so slikah izpuščene povezave, ki pokrivajo zgolj nekatere posebne primere.



Slika 3. Preprosto ogrodje modela



Slika 4. Povezano ogrodje modela

Za oba modela je bilo potrebno določiti parametra s in e. Izbrana je bila začetna vrednost e = 0,5, nato pa so bile preizkušene različne vrednosti s. V tabeli 1 so povprečja mere F1 za vsa štiri iskana zaporedja besed.

s	0,00	0,25	0,50	0,75	1,00
Preprosto	0,6030	**0,6037**	0,5827	0,5718	0,5719
Povezano	0,6825	0,6954	**0,6972**	0,6902	0,6894

Tabela 1. Določanje parametra s

Za preprosto ogrodje je bilo nastavljeno s = 0,25 in za povezano s = 0,50, nato pa so bile preizkušene različne vrednosti parametra e. V tabeli 2 so povprečja mere F1.

e	0,00	0,25	0,50	0,75	1,00
Preprosto	0,5668	**0,6716**	0,6037	0,6039	0,5936
Povezano	0,5198	**0,7057**	0,6972	0,6902	0,6894

Tabela 2. Določanje parametra e

Za obe ogrodji je bil tako nastavljen e = 0,25. Nato je bil preizkušen vpliv števila predpon in pripon (sliki 3 in 4). Povprečne mere F1 iz tega poizkusa so zbrane v tabeli 3.

Predpone, pripone	1	2	3	4
Preprosto	0,3370	**0,6716**	0,4290	0,4101
Povezano	0,6259	**0,7057**	0,6323	0,6971

Tabela 3. Določanje števila predpon in pripon

V obeh primerih se je izkazalo, da je bila začetna odločitev za po dve predponi in priponi pravilna. Podoben poizkus je bil opravljen še za število ciljnih stanj. Rezultati so v tabeli 4.

Preprosto					
Ciljna stanja	1	2	3	4	
F1	0,5630	**0,6716**	0,4290	0,4101	
Povezano					
Ciljna stanja	2	3	4	5	6
F1	0,6329	0,6578	**0,7057**	0,6956	0,6971

Tabela 4. Določanje števila ciljnih stanj

Tudi v tem primeru so se začetne nastavitve izkazale za najboljše. V tabeli 5 so podani natančnejši rezultati za modele, naučene na vseh 90% učnih primerov.

Preprosto				
	Govornik	**Kraj**	**Začetek**	**Konec**
Natančnost	0,6542	0,7800	0,7264	0,4271
Priklic	0,7970	0,4590	0,6056	0,9939
F1	0,7185	**0,5534**	0,6605	0,5974
Povezano				
	Govornik	**Kraj**	**Začetek**	**Konec**
Natančnost	0,8497	0,9257	0,8766	0,6727
Priklic	0,6600	0,2982	0,7803	0,8970
F1	**0,7429**	0,4511	**0,8256**	**0,7688**

Tabela 5. Podrobni rezultati učenja na 90% primerov

Izkaže se, da se preprosto ogrodje obnese bolje na najbolj problematičnem iskanem zaporedju besed, kraju, drugje pa ga povezano prekosi. Za konec pa še povprečne mere F1 pri Marku in nekaterih drugih sistemih za luščenje podatkov: Boosted Wrapper Induction (BWI) [8], Rapier [2], Whisk [12] in LP^2 [3].

Marko	**BWI**	**Rapier**	**Whisk**	**LP^2**
0,7227	0,8465	0,7857	0,6590	0,8355

Tabela 6. Primerjava z drugimi sistemi

4. ZAKLJUČEK

V prispevku so predstavljeni skriti markovski modeli in njihova uporaba za luščenje podatkov. Pojasnjeni so tudi nekateri problemi, ki nastopijo pri implementaciji, in kako so rešeni v sistemu Marko.

Podane so ugotovitve o primerni vrednosti dveh ključnih parametrov sistema. Analizirana je učinkovitost različnih ogrodij skritega markovskega modela. Najbolje se obnese povezano ogrodje s štirimi ciljnimi vozlišči, vendar je tudi preprosto z dvema dokaj dobro. Za obe velja, da delujeta najbolje s po dvema predponama in priponama.

Sistem bi v prihodnje veljalo preizkusiti še na drugih domenah. Poleg tega bi bilo koristno izdelati nazoren prikaz naučenih modelov, ki bi utegnil razkriti pomanjkljivosti in možnosti za izboljšave.

5. LITERATURA

[1] Bikel, D. M., Miller, S., Schwartz, R. in Weischedel, R. (1997). Nymble: A High-Performance Learning Name Finder. Proceedings of ANLP-97

[2] Califf, M. E. in Mooney, R. J. (1999). Relational Learning of Pattern-Match Rules for Information Extraction. Proceedings of the Sixteenth National Conference on Artificial Intelligence

[3] Ciravegna, F. (2001). Adaptive Extraction from Text by Rule Induction and Generalization. Proceedings of 17th International Joint Conference on Artificial Intelligence

[4] Freitag, D. in Kusmerick, N. (2000). Boosted Wrapper Induction. Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence

[5] Freitag, D. in McCallum, A. (1999). Information Extraction with HMMs and Shrinkage. AAAI'99 Workshop on Machine Learning for Information Extraction

[6] Freitag, D. in McCallum, A. (2000). Information Extraction with HMM Structures Learned by Stohastic Optimization. AAAI-2000

[7] Harman, D. K. SAIC Information Extraction. http://www.itl.nist.gov/iad/894.02/related_projects/muc/ [2004-09-15]

[8] Kushmerick, N., Weld., D. in Doornebos, B. (1997). Wrapper Induction for Information Extraction. Proceedings of the 15th International Joint Conference on Artificial Intelligence

[9] Leek, T. R. (1997). Information Extraction Using Hidden Markov Models. Magisterij, UC San Diego

[10] Rabiner, L (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2)

[11] Freitag, D. Seminar Announcements dataset. http://www-2.cs.cmu.edu/~dayne/SeminarAnnouncements/__Source__.html [2004-09-17]

[12] Soderland, S. (1999). Learning Information Extraction Rules for Semi-Structured and Free Text. Machine Learning 34