Jezikovne tehnologije

6. junij 2024

Borko Bošković, Jani Dugonik, Klemen Berkovič, Janez Brest

O predmetu

Predmet

- Jezikovne tehnologije
- ECTS točke: 6
- Predavatelj: doc. dr. Borko Bošković (borko.boskovic@um.si)
- Več informacij: `https://estudij.um.si/`

Vsebina

- Uvod
  - Pregled jezikovnih tehnologij in osnovno procesiranje jezika
- Modeliranje jezika
  - Verjetnostni jezikovni modeli, ocenjevanje modelov in metode glajenja
- Popravljanje pravopisa
  - Besedne in nebesedne pravopisne napake, šumni kanal
- Klasifikacija besedil
  - Metode klasificiranja besedil in ocenjevanje metod klasificiranja
- Analiza sentimenta
  - Osnovne metode analize sentimenta in leksikoni sentimenta
- Semantika in WordNet
  - Strukture za opis semantike, interpretacija semantike, pomen
- Statistično strojno prevajanje
  - Metode in ocenjevanje strojnega prevajanja
- Ekstrakcija informacij in prepoznavanje imenskih entitet
- Napredne jezikovne tehnologije
  - Globoko učenje, vektorji besed

Jezikovne tehnologije

- Cilj:
  - Osnovni principi procesiranja naravnega jezika
  - Uporaba jezikovnih tehnologj.
- Znanja in razumevanje:
  - Opisati smernice razvoja za procesiranje naravnega jezika,
  - opisati več standardnih metod, ki jih uporabljamo v sistemih za procesiranje naravnega jezika za opis morfologije in sintakse,
  - izkazati razumevanje pomembnosti pragmatike pri razumevanju naravnega jezika,
  - razumeti in izkazati znanje o različnih metodologijah, ki jih najdemo pri procesiranju naravnega jezika, in
  - kako te metodologije uporabiti v različnih aplikacijah.
- Usmeritve:
  - Samostojnost in
  - individualnost.

Način ocenjevanja

- Laboratorijske vaje 50 %
  - Seminarska naloga 50 %
  - Naloge 50 %
- 1. vmesni izpit 25 %
- 2. vmesni izpit 25 %
- Pozitivno opravljena vmesna izpita:
  - Povprečna uspešnost nad 50 % in vsak posamezno nad 35 %.
- Če študent ni pozitivno opravil vmesnih izpitov, jih nadomesti s pisnim izpitom v deležu 50 %.

Literatura

- C. D. Manning, H. Schütze: Foundations of statistical natural language processing, Sixth Edition, MIT Press, Cambridge,2003.
- P. Jackson, I. Moulinier: Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization, Second Edition, John Benjamins, cop., Amsterdam, 2007.
- Daniel Jurafsky and James H. Martin. Speech and Language Processing, 2nd edition. Pearson Prentice Hall, 2008.
- Steven Bird, Ewan Klein in Edward Loper. Natural Language Processing with Python. O'Reilly Media, 2009.
- Philipp Koehn, Statistical Machine Translation, Cambridge University Press, 2010
- Li Deng and Yang Liu. Deep Learning in Natural Language Processing, 1st edition, Springer, 2018

Spletni viri

- Slovenska raziskovalna infrastruktura za jezikovne vire in tehnologije
  `http://www.clarin.si/info/o-projektu/`
- Slovensko društvo za jezikovne tehnologije:
  `http://www.sdjt.si/SDJT.html`
- Statistical natural language processing and corpus-based computational linguistics: An annotated list of resources:
  `http://www-nlp.stanford.edu/links/statnlp.html`
- European Language Resources Association:
  `http://www.elra.info/`
- Sistem za vaje:
  `https://estudij.um.si/`

Pregled jezikovnih tehnologij

(angl. Overview of Natural Language Processing)

Računalniška obdelava naravnega jezika

- Računsko jezikoslovje (angl. *Computational Linguistics*)
  - Veja računalništva in informatike, ki se navezuje na jezikoslovje.
  - Cilj: modeliranje naravnega jezika za različne računalniške aplikacije (črkovalniki, prevajalniki, lematizatorji itd.).
- Jezikovne tehnologije (JT) (angl. *Language Technologies*)
  - Skupek tehnologij, ki so namenjena procesiranju naravnega jezika.
  - Računalnik uporablja naravni jezik kot vhodno izhodni tok podatkov.
  - Računalniški jeziki $\leftrightarrow$ računalnik : naravni jezik $\leftrightarrow$ računalnik razumevanje naravnega jezika (angl. *Natural Language Understanding*)
  - angl. *Natural Language Processing* - NLP
  - angl. *Human Language Technology* - HLT

Google Prevajalnik

- Jezikovne tehnologije so namenjena samodejnemu procesiranju naravnega jezika.
- Language technologies are designed to automatically process natural language.
- Jezikovne tehnologije so zasnovane tako, da samodejno obdelujejo naravni jezik.

Naravni jezik

- Predstavlja nepogrešljivo orodje za komunikacijo med ljudmi.
- Za človeka je preprost, računalnikom povzroča preglavice.
- Jezik je živ in se spreminja skozi čas.
- Različni jeziki se med seboj lahko zelo razlikujejo.
- Problemi, ki izstopajo pri jezikovnih tehnologijah
  - Večpomenskost: mnoge besede imajo več pomenov.
    gori na gori gori
  - Parafraze: mnoge vsebine je mogoče izraziti na več načinov.
    nafta $\rightarrow$ *črna kri industrijskega sveta*
    kača $\rightarrow$ *nevarna plazilka*
  - Nedoločenost: mnoga jezikovna sredstva imajo nedoločen pomen, ki ga razberemo šele iz sobesedila.
  - Metafora: je raba besedne zveze namesto druge na podlagi kake njune skupne pomenske lastnosti.
    Denar je sveta vladar.

Strukturalistični in empirični vidiki jezikoslovja

- Strukturalistični pristop:
  - Jezik je omejen in urejen sistem, ki temelji na pravilih.
  - Avtomatska obdelava jezika je mogoča s pomočjo pravil.
  - Pravila se oblikuje v skladu s človeško jezikovno intuicijo.
- Empirični pristop:
  - Jezik je vsota vseh svojih udejanjanj (v govorjenih in pisnih besedilih)
  - Posplošitve o jeziku so mogoče le na podlagi velikih besedilnih zbirk, ki nam služijo kot **vzorec** jezika $\rightarrow$ **korpusi**
  - Strojno učenje (angl. *Machine Learning*):
    - "data-driven automatic inference of rules"

Raziskovalna področja

- Oblikoslovje: besednovrstno označevanje (angl. *part-of-speech tagging*), lematizacija, razčlemba sestavljenih besed
- Skladnja: razpoznavanje stavčnih členov, slovničnih funkcij (osebek/povedek/...); popolna skladenjska analiza
- Glasoslovje: razpoznavanje in tvorjenje govora, pogovorni sistemi
- Pomenoslovje: razreševanje večpomenskosti, avtomatska izdelava semantičnih virov (tezavrov, ontologij)
- Večjezikovne tehnologije: luščenje ustreznic iz korpusov, strojno prevajanje in tolmačenje
- Jezik in internet: iskanje podatkov, rudarjenje besedil (angl. *Text Mining*), napredni spletni iskalniki

Aplikacije

- Cilj je razumevanje naravnih jezikov. To vključuje tudi govorjenje in pisanje v naravnih jezikih.
- Aplikacije: urejevalnik besedil, elektronski slovar, črkovalniki, slovnični pregledovalniki, sintetizator govora, razpoznavalnik govora, razpoznavalnik tekstovnega besedila, prevajalnik naravnih jezikov itd.
- Delitev glede na podatkovne vire:
  - Tekstovni viri (knjige, spletne strani, poročila, elektronska pošta, podnapisi filmov itd.)
  - Dialog oz. komunikacija s človekom.

Aplikacije - tekstovni jezikovni viri

- Poiskati ustrezno besedilo, ki pripada določni tematiki (spletno stran, knjigo itd.).
- Olajšati vnos besedila, zaradi vrste naprave ali telesnih hib.
  - Najbolj prodajana aplikacija za Android v letu 2012 je bila **SwiftKey**.
- Izluščiti določene informacije iz besedila ali članka, ki pripada določenemu področju (npr. športne novice o nogometaših).
- Prevajanje dokumentov iz enega v drug jezik (npr. prevajanje spletnih strani).
- Podajanje povzetka določenega besedila (npr. 3 stransko poročilo 100 stranskega dokumenta).

Aplikacije, ki komunicirajo s človekom.

- Sistemi vprašanj in odgovorov (angl. *Question-answering systems*), kjer naravni jezik predstavlja povpraševalni jezik.
- Samodejni telefonski servis (npr. naročanje izdelkov iz kataloga).
- Učni sistem, kjer je računalnik v interakciji s študenti (npr. učni sistem za matematiko).
- Krmiljenje stroja ali računalnika s pomočjo glasa.
- Sistem za reševanje splošnih problemov (npr. sistem za pomoč pri načrtovanju in razporejanju opravil).

Viri

- Špela Vintar, FF, ULJ, 2006
- Janez Brest, FERI, MB, 2012
- T. Erjavec; IJS: `http://nl.ijs.si/et/teach/mps10-hlt/`
- Wikipedija
  - `http://en.wikipedia.org/wiki/Natural_language_processing`
  - `http://en.wikipedia.org/wiki/Computational_linguistics`
- P. Jackson, I. Moulinier: Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization, Second Edition, John Benjamins, cop., Amsterdam, 2007.
- Language Technology World: `http://www.lt-world.org`
- Natural Language Toolkit: `http://www.nltk.org`

Osnovno procesiranje besedila

(angl. Basic Text Processing)

Regularni izrazi

- Formalen jezik za opis niza znakov.
- Kako najdemo skupino besed (npr.: Hiša, hiša, Hiše, hiše, Hiši, hiši, Hišica, hišica)?
- Kaj pa naslednje besedilo:
  Primož pa Petra potujeta po poročnem potovanju. Potem pa pof, prelepo potovanje postane problem. Pretresena, prašna, prešvicana pririneta pred petrolovo postojanko. Prijazen prodajalec ponudi pomoč "potrebujeta pomoč". Petra predlaga Primožu pogledati po petrolovi prodajalni. Povsod polne prodajne police. Popravljeno, pokliče prodajalec. Pa Primož pa Petra ponovno potujeta po poročnem potovanju.

Regularni izrazi

- [A-z], [0-9], [^a-z] ...
- a | b , Ab | cD, ...
- [Hh](iša | išica)
- Hišk?a
- To+, Go+I, Nee*
- Hiš.
- ^[Pp].+
- \.$
- V besedilu poiščite vse veznike ”in“.

Regularni izrazi - napake

Dva tipa napak:

- Ujemanje z nezaželenimi nizi znakov.
- Zaželene nize znakov ne razpoznamo.

Slovarji

- “Gori na gori gori.”
- Koliko besed?
  - 3 besede ali 2 besedi?
  - 4 tipov besed ali 2 tipa besed?
- Google N-grams: 10^{12} besed in $13 * 10^6$ tipov besed.
  `https://catalog.ldc.upenn.edu/LDC2006T13`

Enostavni leksikalni analizator

- $ time ‘cat jrc-acquis.sl | tr -sc ’A-Za-z’ ’\n’ | sort | uniq -c | sort -rn > frekvence1.txt‘
  real 0m55.666s
  user 0m59.478s
  sys 0m0.901s
- $ time ‘cat jrc-acquis.sl | sed s/[^A-Za-zČŠŽčšž]/*n*/g | sort | uniq -c | sort -rn > frekvence2.txt‘
  real 0m45.020s
  user 0m46.727s
  sys 0m2.888s
- The JRC-Acquis Multilingual Parallel Corpus
  `https://ec.europa.eu/jrc/en/language-technologies`
- Europarl: A Parallel Corpus for Statistical Machine Translation
  `http://www.statmt.org/europarl/`
- Katere besede se najpogosteje uporabljajo?

Frekvence besed

$ head frekvence1.txt

- 741954 in
- 690527 v
- 616613 za
- 437074 je
- 396580 na
- 363879 ki
- 347304 se
- 289553 z
- 245837 o
- 224690 da

$ head frekvence2.txt

- 719929 in
- 687614 v
- 545570 za
- 412470 je
- 345747 se
- 340045 ki
- 327220 na
- 289014 z
- 232087 o
- 224558 da

Problemi leksikalnih analizatorjev

- Različna pravila za različne jezike
- Krajši zapis besed (npr.: oz. $\leftrightarrow$ oziroma)
- Velike in male črke (npr: Gori na gori gori.)
- Posebni znaki (npr.: ČŠŽčšž)
- Smer pisanja besedila
- Delimiterji oz. ločilni znaki
  - Algoritem maksimalnega ujemanja (npr. v kitajščini in japonščini)

Normalizacija besedila

- Skoraj vsak program jezikovnih tehnologij izvaja normalizacijo besedila.
- Leksikalna analiza oz. segmentacija besed
- Normalizacija formatov besed
- Segmentacija povedi

Normalizacija besedila

- Iskanje informacij (angl. *Information Retrieval*)
  - Indeksiranje besedila
  - Povpraševanje po izrazih mora imeti določeno obliko
- Implicitna definicija sorodnih izrazov
  - npr.: tj. $\leftrightarrow$ To je; itd. $\leftrightarrow$ in tako dalje
- Razširitve
  - npr.: Miza $\leftrightarrow$ miza, mize, Mizi, mizica
- Zmogljivejši algoritem / manjša učinkovitost / večja kompleksnost

Preslikava velikih v male črke

- *Case folding*
- Aplikacije, ki se ukvarjajo z iskanjem informacij, spreminjajo velike v male črke.
- Izjeme/problemi:
  - npr.: Prekmurska gibanica, Pomurka
- Analiza sentimenta, strojno prevajanje, ekstrakcija informacij
  - Razlikovanje velikih in malih črk je koristno

Lematizacija (geslenje)

- Lematizacija predstavlja korak v normalizaciji besedila, kjer so pregibne besedne oblike v besedilu, npr. mize ali mizi, poenotene na svojo lemo. Te leme se nato lahko uporabljajo pri leksikalni analizi besedil, npr. kot iskalna funkcija konkordančnikov, avtomatski gradnji leksikonov, pri strojnem prevajanju, itd.
- Osnovno obliko besede imenujemo lema
- Krnjenje besed ne preoblikuje besede v njihovo slovarsko obliko, ampak besedam zgolj odreže končnico.
- hodim, hodiš, hodita, hodimo $\leftrightarrow$ lema: hoditi, krn: hodi
- Leksem je kombinacija leme z besedno vrsto.
- Lematizacija je iskanje pravilne slovarske osnovne oblike besede.
- Uporabno pri strojnem prevajanju.

Morfologija

- Veda, ki preučuje tvorbo besed, tipe besed in njihove oblike.
- Morfem - najmanjša pomenska enota oz. najmanjši del besede, ki ima svoj pomen.
- Male smiselne enote, ki tvorijo besede.
  - Jedro besede
  - Razširitve besed (ponavadi slovnične)

Segmentacija povedi

- Kje je konec povedi?
- Znaki ki končujejo povedi: “?”, “!” in “.”.
- Izjeme “npr.”, “dr.”.
- Uporaba klasifikatorjev.
  - Ročno ugotavljanje (piki sledi velika začetnica, dolžina besede pred piko itd.).
  - Odločitvena drevesa.
  - Nevronske mreže.

Viri

- Dan Jurafsky, Chris Manning, Natural Language Processing
- Sašo Džeroski, Tomaž Erjavec, Strojno učenje lematizacije neznanih slovenskih besed, Odsek za inteligentne sisteme, Institut "Jožef Stefan", Jamova 39, 1000 Ljubljana

Razdalja urejanja

(angl. Edit Distance)

Kako podobna sta si dva niza znakov?

- Uporablja se za črkovanje, strojno prevajanje, razpoznavanje govora itd.
- Katera beseda je najbolj podobna besedi jezik:
  - jezikoslovje
  - ježiček
  - jez
- Problem poravnave nizov.
  - lopar_____
  - ___parazit

Razdalja urejanja

- Minimalna razdalja urejanja med dvema nizoma (angl. *Minimum Edit Distance*).
- Najmanjše število operacij urejanja.
  - Vstavljanje (angl. *Inserton*)
  - Brisanje (angl. *Deletion*)
  - Zamenjava (angl. *Substitution*)
- Koliko operacij je potrebnih, da spremenimo en niz znakov v drugega?

Minimalna razdalja urejanja

- `samodejnost`
- `samostojnost`
- `--zzb---`
- Cena vsake operacije je ena $\Rightarrow$ razdalja je 3 (Levenshtein).
- Cena zamenjave je 2 $\Rightarrow$ razdalja je 5.

Uporabnost razdalje urejanja

- Ocenjevanje strojnega prevajanja in razpoznave govora.
- Pri strojnem prevajanju se izračuna razlika urejanja besed med stavkom strojnega prevajanja in stavkom, ki ga je prevedel človek.
- Pri razpoznavanju govora ugotavljamo razliko urejanja besed med stavkom, ki ga je govorec povedal in stavkom, ki ga je računalnik razpoznal.

Določanje minimalne razlike urejanja

- Iskanje zaporedja operacij urejanja, da iz začetnega niza znakov dobimo želeni niz znakov.
  - Začetno stanje (beseda, ki jo transformiramo).
  - Operacije: vstavljanje, brisanje, zamenjava.
  - Ciljno stanje: beseda, ki jo iščemo.
  - Cena poti: število operacij urejanja.
- Drevo preiskovanja

Minimalna razdalja urejanja

- Velikost prostora vseh sekvenc urejanja je ogromna.
- Naivna ali požrešna metoda je nesprejemljiva.
- Dosti različnih poti najde isto besedo.
  - Ne želimo slediti vsem možnim potem.
  - Zanimajo nas le najkrajše poti.

Minimalna razdalja urejanja

- Imamo dva niza znakov X in Y,
- Dolžina nizov V_x in V_y.
- Razdalja med nizoma je D_{V_x,V_y}.
  - Razdalja med podnizi: $D_{i,j}$; $i \in \{1, ..., V_x\}, j \in \{1, ..., V_y\}$.

Dinamično programiranje

- *Dynamic programming*
- Dinamično programiranje - končna rešitev je sestavljena iz komponent rešitve oz. iz delnih rešitev. Ko na tekočem koraku ugotovimo, da delna rešitev oz. komponenta neke rešitve ne vodi h cilju, to delno rešitev zavržemo.
- 2D polje razdalj.
- Problem rešujemo s pomočjo rešitev podproblemov.
- Metoda od spodaj navzgor (angl. *Bottom-up*).
  - Izračunamo D(i,j) za male i,j.
  - Za večje se izračuna glede na rezultate manjših (i,j).
  - Izračunamo D(i,j) za vse vrednosti i in j (i $\in \{0, ..., V_x\}$; j$\in \{0, ..., V_y\}$).

Razdalja urejanja

- Izračunajte razdaljo urejanja za naslednji besedi: Windows in Linux.

Algoritem za minimalno razdaljo urejanja

for i = 0 to V_x **do**
  D[i,0] = i;
end for
for j = 0 to V_y **do**
  D[0,j] = j
end for
for i = 1 to V_x **do**
  for j = 1 to V_y **do**

$$D[i,j] = min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1] + \begin{cases} 2, & X[i] \; != \; Y[j] \\ 0, & X[i] \; == \; Y[j] \end{cases} \end{cases}$$

  end for
end for

Algoritem za minimalno razdaljo urejanja

	/	s	a	m	o	s	t	o	j	n	o	s	t
/	0	1	2	3	4	5	6	7	8	9	10	11	12
s	1												
a	2												
m	3												
o	4												
d	5												
e	6												
j	7												
n	8												
o	9												
s	10												
t	11												

$$D[i,j] = min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1] + \begin{cases} 2, & X[i] \; != \; Y[j] \\ 0, & X[i] \; == \; Y[j] \end{cases} \end{cases}$$

Algoritem za minimalno razdaljo urejanja

	/	s	a	m	o	s	t	o	j	n	o	s	t
/	0	1	2	3	4	5	6	7	8	9	10	11	12
s	1	0	1	2	3	4	5	6	7	8	9	10	11
a	2	1	0	1	2	3	4	5	6	7	8	9	10
m	3	2	1	0	1	2	3	4	5	6	7	8	9
o	4	3	2	1	0	1	2	3	4	5	6	7	8
d	5	4	3	2	1	2	3	4	5	6	7	8	9
e	6	5	4	3	2	3	4	5	6	7	8	9	10
j	7	6	5	4	3	4	5	6	5	6	7	8	9
n	8	7	6	5	4	5	6	7	6	5	6	7	8
o	9	8	7	6	5	6	7	6	7	6	5	6	7
s	10	9	8	7	6	5	6	7	8	7	6	5	6
t	11	10	9	8	7	6	5	6	7	8	7	6	5

Primer

Prikažite način delovanja algoritma za računanje razdalje urejanja za besedi Linux in Windows.

Poravnava nizov

- Razdalja urejanja nam ponavadi ne zadošča.
  - Zanima nas kako poravnati dve besedi oz. katere operacije urejanja je potrebno narediti, da spremenimo en niz v drugega.
- S pregledovanjem poti vračanja ugotovimo, katere operacije so potrebne.
- Glede na smer gibanja in vrednosti razdalje ugotavljamo operacije.

Poravnava nizov

	/	s	a	m	o	s	t	o	j	n	o	s	t
/	0 -	1	2	3	4	5	6	7	8	9	10	11	12
s	1	0 -	1	2	3	4	5	6	7	8	9	10	11
a	2	1	0 -	1	2	3	4	5	6	7	8	9	10
m	3	2	1	0 -	1	2	3	4	5	6	7	8	9
o	4	3	2	1	0 -	1 V	2	3	4	5	6	7	8
d	5	4	3	2	1	2	3 Z	4	5	6	7	8	9
e	6	5	4	3	2	3	4	5 Z	6	7	8	9	10
j	7	6	5	4	3	4	5	6	5 -	6	7	8	9
n	8	7	6	5	4	5	6	7	6	5 -	6	7	8
o	9	8	7	6	5	6	7	6	7	6	5 -	6	7
s	10	9	8	7	6	5	6	7	8	7	6	5 -	6
t	11	10	9	8	7	6	5	6	7	8	7	6	5-

Algoritem poravnave nizov

for i = 0 to V_x **do**
    D[i,0] = i;
end for
for j = 0 to V_y **do**
    D[0,j] = j
end for
for i = 1 to V_x **do**
    for j = 1 to V_y **do**

$$D[i,j] = min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1] + \begin{cases} 2, & X[i] \; != \; Y[j] \\ 0, & X[i] \; == \; Y[j] \end{cases} \end{cases}$$

$$kazalec(i,j) = \begin{cases} levo & \rightarrow vstavljanje \\ gor & \rightarrow brisanje \\ diagonalno & \rightarrow zamenjava \end{cases}$$

    end for
end for

Primer

Prikažite način delovanja algoritma za računanje poravnave za besedi Linux in Windows.

Zahtevnost algoritma

- Časovna zahtevnost: $O(n \cdot m)$.
- Prostorska zahtevnost: $O(n \cdot m)$.
- Ugotavljanje operacij urejanja: $O(n + m)$.

Utežena razdalja urejanja

- Določene operacije so pri določenih opravilih in črkah verjetnejše.
- Črkovalnik.
  - Razporeditev tipk na tipkovnici.
  - Naravni jezik, ki ga uporabljamo (frekvenca znakovnih bigramov).

Minimalna razdalja urejanja z utežmi

D[0,0] = 0;
for i = 0 to V_x **do**
  D[i,0] = D[i-1,0]+brisanje(x[i]);
end for
for j = 0 to V_y **do**
  D[0,j] = D[0,j-1]+vstavljanje(y[j]);
end for
for i = 1 to V_x **do**
  for j = 1 to V_y **do**

$$D[i,j] = min \begin{cases} D[i-1,j] + brisanje(x[i]) \\ D[i,j-1] + vstavljanje(y[j]) \\ D[i-1,j-1] + zamenjava(x[i], y[j]) \end{cases}$$

  end for
end for

Vir

- Dan Jurafsky, Chris Manning, Natural Language Processing
  `http://web.stanford.edu/~jurafsky/NLPCourseraSlides.html`

Jezikovni modeli

(angl. Language Models)

Verjetnostni jezikovni model

- Zaporedju besed dodeljuje verjetnosti.
  - Strojno prevajanje (angl. *Machine Translation*) - katera poved ima večjo verjetnost, glede na jezikovni model?
  - Preverjanje pravopisnih napak (angl. *Spelling Correction*) - kako ena beseda spremeni verjetnost povedi?
  - Razpoznavanje govora (angl. *Speech Recognition*) - z uporabo verjetnostnega modela izboljša učinkovitost razpoznavanja.
  - Uporabno tudi na drugih področjih jezikovnih tehnologij (npr. povzemanje besedila, sistem vprašanj in odgovorov).

Kako izračunamo verjetnosti

- $P(dan|danes\ je\ lep) = \frac{\text{število}(danes\ je\ lep\ dan)}{\text{število}(danes\ je\ lep)}$
- Preveč povedi in možnosti za tak izračun.
- Potrebna enormna količina podatkov.
- Potrebna je alternativa.
  - $P(dan|danes\ je\ lep) \approx P(dan|lep)$
  - $P(dan|danes\ je\ lep) \approx P(dan|je\ lep)$
- Markova predpostavka.
  - $P(b_1\ b_2\ b_3\ \ldots\ b_n) \approx \prod_i P(b_i|b_{i-k}\ \ldots\ b_{i-1})$
  - $P(b_i|b_1\ b_2\ b_3\ \ldots\ b_{i-1}) \approx P(b_i|b_{i-k}\ \ldots\ b_{i-1})$

N-grami

- Uporabimo lahko trigrame, štirigrame in petgrame.
  - Trigrami: $P(b_i|b_1\ b_2\ b_3\ \ldots\ b_{i-1}) \approx P(b_i|b_{i-2}b_{i-1})$
  - Štirigrami: $P(b_i|b_1\ b_2\ b_3\ \ldots\ b_{i-1}) \approx P(b_i|b_{i-3}b_{i-2}b_{i-1})$
  - Petgrami: $P(b_i|b_1\ b_2\ b_3\ \ldots\ b_{i-1}) \approx P(b_i|b_{i-4}b_{i-3}b_{i-2}b_{i-1})$
- V splošnem je to model jezika, ki ni popoln.
  - Jezik ima daljše odvisnosti.
- Iz praktičnih razlogov uporabimo logaritme.
  - $log(p_1 \cdot p_2 \cdot p_3 \cdot p_4) = log(p_1) + log(p_2) + log(p_3) + log(p_4)$
  - Izognemo se malim vrednostim.
  - Seštevanje je hitrejše od množenja.
- Cenilka po metodi največjega verjetja (angl. *Maximum Likelihood Estimate*).

$$P(b_i|b_{i-1}) = \frac{\text{število}(b_{i-1},b_i)}{\text{število}(b_{i-1})}\ ; \qquad P(b_i) = \frac{\text{število}(b_i)}{\sum_{b_j \in V} \text{število}(b_j)}$$

- Orodje za jezikovno modeliranje: SRILM `http://www.speech.sri.com/projects/srilm/`.

Kako dober je jezikovni model?

- Ali model daje prednosti dobrim ali slabim povedim?
- Dodeli večje verjetnosti “realnim” oz. “zelo pogostim” povedim ali “nepravilnim”, ki se ne pojavljajo pogosto?
- Model naučimo na učni množici (angl. *Training set*).
- Model preizkusimo na še ne videnih podatkih.
  - Testna množica (angl. *Test set*) so ne videni podatki, ki se razlikujejo od učne množice.
  - Metrike (angl. *Evaluation metric*) ocenijo kako dobro se je naš model izkazal na testni množici.

Zunanje ovrednotenje n-gram modelov

- Primerjava modela A in B.
  - Vsak od modelov opravi svojo nalogo.
    - Črkovanje, razpoznavanje govora, strojno prevajanje.
  - Določanje uspešnosti modela A in B.
    - Število napačno črkovanih besed.
    - Število napačno prevedenih besed.
  - Primerjava uspešnosti modelov A in B.

Zunanje ovrednotenje n-gram modelov

- Je časovno zahtevno - lahko traja dneve ali tedne.
- Uporabimo notranje ovrednotenje oz. perpleksnost (angl. *Perplexity*).
  - Slabi približki.
    - Uporabimo učno množico kot testno množico.
    - Daje oceno, ki nam lahko nekaj pove o modelu.
    - Primerno za začetne eksperimente.

Perpleksnost

- Boljši model daje boljšo napoved ne videne povedi.
- Perpleksnost je inverz verjetnosti testne množice, ki je normaliziran s številom besed.

$$PP(S) = P(b_1\ b_2\ b_3\ \ldots\ b_n)^{-\frac{1}{n}} = \sqrt[n]{\frac{1}{P(b_1\ b_2\ b_3\ \ldots\ b_n)}}$$

$$PP(S) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(b_i \mid b_1\ b_2\ b_3\ \ldots\ b_{i-1})}}$$

- Trigrami

$$PP(S) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(b_i \mid b_{i-2}\ b_{i-1})}}$$

- Maksimiranje verjetnosti je enako kot minimizacija perpleksnosti.

Perpleksnost kot povprečni vejitveni faktor

- Koliko možnosti imamo v povprečju, da nadaljujemo poved?
- Imejmo "poved", ki jo definira naslednji regularni izraz: [0-9]+.
- Kakšna je perpleksnost glede na model pri katerem je verjetnost pojavitve določene števke $\frac{1}{10}$?
  $PP(S) = P(b_1 b_2 b_3 .... b_n)^{-\frac{1}{n}} = ((\frac{1}{10})^n)^{-\frac{1}{n}} = (\frac{1}{10})^{-1} = 10$
- Manjša perpleksnost pomeni boljši model.

Nevarnosti prekomernega prilagajanja

- angl. *overfitting*
- Modeli, ki temeljijo na n-gramih, delujejo dobro v primerih ko je testna množica podobna učni množici.
- V splošnem se testna množica razlikuje od učne mnižice.
- Problem so ne videni podatki - podatki, ki se ne pojavljajo v učni množici in jih najdemo v testni množici.
  - $P(dan \mid danes\ je\ lep) = 0$
  - Ni možno izračunati perpleksnosti (deljenje z 0).
- Potrebujemo bolj robusten model.

Glajenje Add-one (Laplace)

- Glajenje (angl. *Smoothing*).
- Zamislimo si, da smo vsako besedo videli enkrat več, kot smo jo dejansko videli.
- Enostavno vsakemu števcu dodamo 1.
- Cenilka po metodi največjega verjetja (angl. *Maximum Likelihood Estimate*):

$$P_{MLE}(b_i|b_{i-1}) = \frac{\text{število}(b_{i-1},b_i)}{\text{število}(b_{i-1})}; \quad P_{MLE}(b_i) = \frac{\text{število}(b_i)}{\sum_{b_j \in V} \text{število}(b_j)}$$

- Verjetnostna ocena z glajenjem Laplace:

$$P_{Add-one}(b_i) = \frac{\text{število}(b_i)+\mathbf{1}}{\sum_j (\text{število}(b_j)+1)} = \frac{\text{število}(b_i)+\mathbf{1}}{\mathbf{N}+\mathbf{V}}$$

$$P_{Add-one}(b_i|b_{i-1}) = \frac{\text{število}(b_{i-1},b_i)+\mathbf{1}}{\text{število}(b_{i-1})+\mathbf{V}}$$

- Ni primerno za modele, ki temeljijo na n-gramih (imamo ogromno "ne videnih" podatkov).
- Uporablja se za druge modele jezikovnih tehnologij.
  - Klasifikacija besedila.
  - V domenah kjer ni dosti "ne videnih podatkov".

Primer za besedni bigram

Besedilo:
I want to see a movie.
I love chocolate cake with vanilla cream.
I have to wake up early in the morning.
Microsoft and delays go together like ice cream and apple pie.
They have the best ice cream in town.
Their creamy milk is made into a delicious ice cream.

Katera poved je bolj verjetna?

- ☐ I have to go to sleep.
- ☐ I want ice cream.

Primer za besedni bigram

- Povedi najprej predprocesiramo tako, da odstranimo vsa ločila (vejice, pike, klicaje, itd.) in posebne znake, in spremenimo vse velike črke v male črke.
- Ker računamo besedne bigrame, je potrebno na začetku in na koncu povedi dodati znak za začetek <s> oz. konec povedi </s>.
- To naredimo zato, da izračunamo, kolikokrat se beseda pojavi na začetku ali na koncu povedi.

Primer za besedni bigram

Predprocesirano besedilo:

<s>i want to see a movie</s>
<s>i love chocolate cake with vanilla cream</s>
<s>i have to wake up early in the morning</s>
<s>microsoft and delays go together like ice cream and apple pie</s>
<s>they have the best ice cream in town</s>
<s>their creamy milk is made into a delicious ice cream</s>
$V = 40$

Katera poved je bolj verjetna?
<s>i have to go to sleep</s>
<s>i want ice cream</s>

Primer za besedni bigram

$$P(\text{<s> i have to go to sleep </s>}) = \frac{4}{46} \cdot \frac{2}{43} \cdot \frac{2}{42} \cdot \frac{1}{42} \cdot \frac{1}{41} \cdot \frac{1}{42} \cdot \frac{1}{40}$$
$$= 6.7e^{-11}$$
$$P(\text{i} \mid \text{<s>}) = \frac{3+1}{6+40} = \frac{4}{46}$$
$$P(\text{have} \mid \text{i}) = \frac{1+1}{3+40} = \frac{2}{43}$$
$$P(\text{to} \mid \text{have}) = \frac{1+1}{2+40} = \frac{2}{42}$$
$$P(\text{go} \mid \text{to}) = \frac{0+1}{2+40} = \frac{1}{42}$$
$$P(\text{to} \mid \text{go}) = \frac{0+1}{1+40} = \frac{1}{41}$$
$$P(\text{sleep} \mid \text{to}) = \frac{0+1}{2+40} = \frac{1}{42}$$
$$P(\text{</s>} \mid \text{sleep}) = \frac{0+1}{0+40} = \frac{1}{40}$$

Primer za besedni bigram

$P(\text{<s> i want ice cream </s>}) = \frac{4}{46} \cdot \frac{2}{43} \cdot \frac{1}{41} \cdot \frac{4}{43} \cdot \frac{3}{44} = 6.3e^{-7}$

$\text{<s> i} = P(\text{i} \mid \text{<s>}) = \frac{3+1}{6+40} = \frac{4}{46}$

$\text{i want} = P(\text{want} \mid \text{i}) = \frac{1+1}{3+40} = \frac{2}{43}$

$\text{want ice} = P(\text{ice} \mid \text{want}) = \frac{0+1}{1+40} = \frac{1}{41}$

$\text{ice cream} = P(\text{cream} \mid \text{ice}) = \frac{3+1}{3+40} = \frac{4}{43}$

$\text{cream </s>} = P(\text{</s>} \mid \text{cream}) = \frac{2+1}{4+40} = \frac{3}{44}$

$P(\text{<s> i have to go to sleep </s>}) = 6.7e^{-11}$

$P(\text{<s> i want ice cream </s>}) = 6.3e^{-7}$

Večjo verjetnost ima poved "I want ice cream".

Naloga za besedni bigram

nogomania.com:

Ronaldo veliki junak trilerja.
Messi rešil Barcelono.
Ronaldo zabija več kot klubi.
Ronaldo noče oditi.
Lahko zabijam kot Messi in Ronaldo.
Ronaldo bo postal gostinec.
Messi in Higuain kot majhni punčki.

Katera poved je bolj verjetna?
Messi je najboljši.
Ronaldo je najboljši.

Naloga za besedni bigra

Predprocesirano besedilo:

<s>ronaldo veliki junak trilerja</s>
<s>messi rešil barcelono</s>
<s>ronaldo zabija več kot klubi</s>
<s>ronaldo noče oditi</s>
<s>lahko zabijam kot messi in ronaldo</s>
<s>ronaldo bo postal gostinec</s>
<s>messi in higuain kot majhni punčki</s>
$V = 24$

Katera poved je bolj verjetna?
<s>messi je najboljši</s>
<s>ronaldo je najboljši</s>

Izračun

$P(\text{<s> messi je najboljši </s>}) = \frac{3}{31} \cdot \frac{1}{27} \cdot \frac{1}{24} \cdot \frac{1}{24} = 6.22e^{-6}$
$P(\text{messi} \mid \text{<s>}) = \frac{2+1}{7+24} = \frac{3}{31}$
$P(\text{je} \mid \text{messi}) = \frac{0+1}{3+24} = \frac{1}{27}$
$P(\text{najboljši} \mid \text{je}) = \frac{0+1}{0+24} = \frac{1}{24}$
$P(\text{</s>} \mid \text{najboljši}) = \frac{0+1}{0+24} = \frac{1}{24}$

$P(\text{<s> ronaldo je najboljši </s>}) = \frac{5}{31} \cdot \frac{1}{28} \cdot \frac{1}{24} \cdot \frac{1}{24} = 1e^{-5}$
$P(\text{ronaldo} \mid \text{<s>}) = \frac{4+1}{7+24} = \frac{5}{31}$
$P(\text{je} \mid \text{ronaldo}) = \frac{0+1}{4+24} = \frac{1}{28}$
$P(\text{najboljši} \mid \text{je}) = \frac{0+1}{0+24} = \frac{1}{24}$
$P(\text{</s>} \mid \text{najboljši}) = \frac{0+1}{0+24} = \frac{1}{24}$

Večjo verjetnost ima poved "Ronaldo je najboljši".

Izračun z uporabo logaritma

$$\log(P(\text{<s>messi je najboljši</s>})) =$$
$$log\,\tfrac{3}{31} + log\,\tfrac{1}{27} + log\,\tfrac{1}{24} + log\,\tfrac{1}{24} = -5,2$$

$$\log(P(\text{<s>ronaldo je najboljši</s>})) =$$
$$log\,\tfrac{5}{31} + log\,\tfrac{1}{28} + log\,\tfrac{1}{24} + log\,\tfrac{1}{24} = -5$$

"Backoff" in interpolacija

- V primeru neznanih besednih zvez uporabimo informacije manjših besednih zvez.
- "Backoff"
  - Uporabimo trigrame, če imamo primerne podatke
  - v nasprotnem uporabimo bigrame in
  - na koncu uporabimo unigrame.
- Interpolacija
  - Kombinacija unigramov, bigramov in trigramov
  - Interpolacija daje boljše rezultate

Linearna interpolacija

- Enostavna interpolacija
  $\hat{P}(b_n|b_{n-2}b_{n-1}) =$
  $\lambda_1 P(b_n|b_{n-2}b_{n-1}) + \lambda_2 P(b_n|b_{n-1}) + \lambda_3 P(b_n)$
  $\sum_i \lambda_i = 1$
- Interpolacija odvisna od vsebine
  $\hat{P}(b_n|b_{n-2}b_{n-1}) = \lambda_1(b_{n-2}^{n-1})P(b_n|b_{n-2}b_{n-1}) +$
  $\lambda_2(b_{n-2}^{n-1})P(b_n|b_{n-1}) + \lambda_3(b_{n-2}^{n-1})P(b_n)$

Primer enostavne interpolacije

- <s> jaz sem janez </s>
- <s> janez je moje ime </s>
- <s> priimka pa ne povem, sem skrivnosten </s>
- Kakšna je verjetnost P(janez|jaz sem), če vzamemo $\lambda_i = \frac{1}{3}$?
- P(janez|jaz sem) $= \frac{1}{3} \cdot \frac{2}{19} + \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{1}$

Kako nastaviti λ_i?

- Učna množica (učni korpus)
- Razvojna množica (angl. Held-Out set) - uporablja se za nastavljanje meta parametrov
- Testna množica (testni korpus)
- Izberemo take vrednosti za λ_i, da maksimiramo verjetnosti za razvojno množico
  - Ugotovimo verjetnosti n-gramov (učna množica)
  - Iščemo vrednosti λ_i, ki dajejo največjo verjetnost za razvojno množico: $\log P(b_1 ... b_n | M(\lambda_1 ... \lambda_k)) = \sum_i \log P_{M(\lambda_1 ... \lambda_k))}(b_i | b_{i-1})$

Neznane besede?

- Poznamo vse besede vnaprej
  - Fiksna dolžina slovarja
- Ponavadi vseh besed ne poznamo vnaprej
  - Dolžina slovarja ni znana
- Ustvarimo neznani leksikalni simbol (angl. *Unknown Token*) <UNK>
  - Učenje verjetnosti <UNK>
    - Ustvarimo leksikon L fiksne dolžine V
    - Pri normalizaciji besedila, vsako neznano besedo, ki ni v slovarju zamenjamo z <UNK>
    - Nato ugotavljamo njene verjetnosti enako kot pri ostalih besedah
  - Faza dekodiranja
    - Za vse neznane besede uporabimo verjetnosti besed <UNK>

Ogromni spletni n-grami

- Kako obravnavamo ogromne korpuse n-gramov?
- Klestenja (angl. *Pruning*)
  - Shranimo le tiste n-grame katerih število je večje od določenega praga
  - Klestenje na osnovi entropije
- Učinkovitost
  - Učinkovite podatkovne strukture kot so npr. drevesa
  - Približni jezikovni modeli
  - Shranjevanje besed s pomočjo indeksov (Huffmanovo kodiranje)
  - Predstavitev verjetnosti - števila predstavimo z manj biti (npr. 4 - 8)

Glajenje namenjeno spletnim n-gramom

- "Stupid backoff" (Brants in ostali 2007)
- Enostavno uporabimo relativne frekvence

$$S(b_i|b_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{število}(b_{i-k+1}^{i})}{\text{število}(b_{i-k+1}^{i-1})} & \text{če število}(b_{i-k+1}^{i}) > 0 \\ 0{,}4S(b_i|b_{i-k+2}^{i-1}) & \text{drugače} \end{cases}$$

$$S(b_i) = \frac{\text{število}(b_i)}{N}$$

Glajenje n-gramov

- Glajenje ”Add-1“
  - Primerno za klasifikacijo dokumentov, ni primerno za jezikovne modele
- Najbolj uporabljene metode uporabljajo interpolacijo
- Za velike korpuse se uporablja ”Stupid backoff“

Napredni jezikovni modeli

- Diskriminatorni modeli
  - Uporablja uteži n-gramov in se ne prilagaja učni množici
- Model temelječ na razpoznavanju (angl. *Parsing-based models*)
- Model, ki uporablja predpomnjenje
  - Besede, ki so se pred kratkim uporabile imajo večjo verjetnost pojavitve

$$P_{pred}(b|zgodovina) = \lambda P(b_i|b_{i-2}b_{i-1}) + (1+\lambda)\frac{\text{število}(b \in zgodovina)}{|zgodovina|}$$

  - Daje slabe rezultate pri razpoznavanju govora

Posplošitev glajenja "Add - 1"

- Glajenje "Add - 1":
  - $P_{Add-1}(b_i|b_{i-1}) = \frac{\text{število}(b_{i-1},b_i)+1}{\text{število}(b_{i-1})+V}$
- Glajenje "Add - k":
  - $P_{Add-k}(b_i|b_{i-1}) = \frac{\text{število}(b_{i-1},b_i)+k}{\text{število}(b_{i-1})+k\cdot V}$
  - $P_{Add-k}(b_i|b_{i-1}) = \frac{\text{število}(b_{i-1},b_i)+m\frac{1}{V}}{\text{število}(b_{i-1})+m}$
- Glajenje "Unigram prior":
  - $P_{\text{UnigramPrior}}(b_i|b_{i-1}) = \frac{\text{število}(b_{i-1},b_i)+m\cdot P(b_i)}{\text{število}(b_{i-1})+m}$

Napredni algoritmi glajenja

- Temeljijo na intuiciji
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell
- V izračunih uporabljajo stvari, ki so videne le enkrat
  - Pomagajo ovrednotiti stvari, ki niso bile videne

Glajenje "Good-Turing"

- N_c - število stvari, ki se je ponovilo c-krat
- Za rojstni dan ste dobili naslednja darila:
  10 x čokolada, 5 x šampanjec, 2 x pivo, **1 x vino, 1 x knjiga, 1 x ura**
- Kakšna je verjetnost, da naslednji gost prinese vino?

Glajenje "Good-Turing"

- N_c - število stvari, ki se je ponovilo c-krat
- Za rojstni dan ste dobili naslednja darila:
  10 x čokolada, 5 x šampanjec, 2 x pivo, **1 x vino, 1 x knjiga, 1 x ura**
- Kakšna je verjetnost, da naslednji gost prinese vino?
  - $\frac{1}{20}$

Glajenje "Good-Turing"

- N_c - število stvari, ki se je ponovilo c-krat
- Za rojstni dan ste dobili naslednja darila:
  10 x čokolada, 5 x šampanjec, 2 x pivo, **1 x vino, 1 x knjiga, 1 x ura**
- Kakšna je verjetnost, da naslednji gost prinese vino?
  - $\frac{1}{20}$
- Kakšna je verjetnost, da naslednji gost prinese darilo, ki se bo razlikovalo od prejšnjih?
  - Uporabimo informacije, ki so se ponovile 1x

Glajenje "Good-Turing"

- N_c - število stvari, ki se je ponovilo c-krat
- Za rojstni dan ste dobili naslednja darila:
  10 x čokolada, 5 x šampanjec, 2 x pivo, **1 x vino, 1 x knjiga, 1 x ura**
- Kakšna je verjetnost, da naslednji gost prinese vino?
  - $\frac{1}{20}$
- Kakšna je verjetnost, da naslednji gost prinese darilo, ki se bo razlikovalo od prejšnjih?
  - Uporabimo informacije, ki so se ponovile 1x
  - $\frac{3}{20}$ ($N_1 = 3$)

Glajenje "Good-Turing"

- P^*_{GT}(unigrami z ničelno frekvenco) $= \frac{N_1}{N}$
- Za nevideno stvar:
  - c $= 0$
  - $MLE = \frac{0}{20} = 0$
  - $P^*_{GT} = \frac{3}{20}$
- $c^* = \frac{(c+1)N_{c+1}}{N_c}$ (za ne ničelno frekvenco)
- $P^*_{GT} = \frac{c^*}{N}$
- Za uro:
  - c $= 1$
  - $MLE = \frac{1}{20}$
  - $c^* = \frac{(c+1)N_{c+1}}{N_c} = \frac{2 \cdot N_2}{N_1} = \frac{2 \cdot 1}{3} = \frac{2}{3}$
  - $P^*_{GT}(ura) = \frac{\frac{2}{3}}{20} = \frac{1}{30}$
- P^*_{GT}(n-grami z ničelno frekvenco) $= \frac{c^*}{N}$
  N_0 ocenimo. Npr. za bigrame $N_0 = V * V - N^b$
  N^b - število unikatnih bigramov

Problemi glajenja "Good-Turing"

- Koliko je P^*_{GT} za najbolj pogosto besedo korpusa?
- Problem nastopi pri besedah, ki se zelo pogosto pojavljajo
- Za male k velja $N_k > N_{k+1}$
- Za velike k, imamo velike skoke in ničelne ocene.
- Rešitev: vrednosti problematičnih empiričnih N_k se nadomestijo z vrednostmi določene prilagoditvene funkcije

Števila pri glajenju "Good-Turing"

- Povzeto iz: Church in Gale (1991)

Števec c	Good Turing c^*
0	0,0000270
1	0,446
2	1,26
3	2,24
4	3,24
5	4,22
6	5,19
7	6,21
8	7,24
9	8,25

- Zdi se, da je $c^* = c - 0,75$

Prihranek s pomočjo interpolacije

- Prihranimo nekaj časa s preprostim odštevanjem 0,75 ali neko vrednostjo d!
  $P_{prihranek} = (b_i|b_{i-1}) = \frac{število(b_{i-1},b_i)-d}{število(b_{i-1})} + \lambda(b_{i-1})P(b)$
- $\lambda(b_{i-1})$ je interpolacijska utež.
- Mogoče je vseeno potrebno imeti določene posebne vrednosti d za manjše vrednosti števcev.
- Ali res moramo uporabiti običajni P za unigrame?

Glajenje Kneser-Ney

- Boljše ocenjevanje verjetnosti unigramov, ki se manjkrat pojavijo.
  - Beseda "sobota" je bolj splošna kot beseda "krila".
  - Toda verjetnost, da besedi "Murska" sledi "Sobota", je zelo velika.
- Unigrami so uporabni v primeru, ko določenega bigrama nismo videli.
- Namesto verjetnosti $P(b)$ (kako verjetna je beseda b), uporabimo verjetnost $P_{nadaljevanja}(b)$ (kako verjetna je beseda b, kot nadaljevanje nečesa).
  $P_{nadaljevanja}(b) \propto |\{b_{i-1} : število(b_{i-1}, b) > 0\}|$

Glajenje Kneser-Ney

- Kolikokrat se je beseda b pojavila kot nadaljevanje nečesa:
  $P_{nadaljevanja}(b) \propto |\{b_{i-1} : število(b_{i-1}, b) > 0\}|$
- Normalizacija s številom vseh bigramov:
  $|\{(b_{j-1}, b_j) : število(b_{j-1}, b_j) > 0\}|$

$$P_{nadaljevanja}(b) = \frac{|\{b_{i-1} : število(b_{i-1}, b) > 0\}|}{|\{(b_{j-1}, b_j) : število(b_{j-1}, b_j) > 0\}|}$$

Glajenje Kneser-Ney

Za bigrame:

$$P_{KN} = (b_i|b_{i-1}) = \frac{max(število(b_{i-1}, b_i) - d, 0)}{število(b_{i-1})} + \lambda(b_{i-1}) P_{nadaljevanja}(b_i)$$

$\lambda(b_{i-1})$ je normalizirana konstanta:

$$\lambda(b_{i-1}) = \frac{d}{število(b_{i-1})} |\{b : število(b_{i-1}, b) > 0\}|$$

Glajenje Kneser-Ney

Rekurzivna enačba:

$$P_{KN}(b_i|b_{i-n+1}^{i-1}) = \frac{max(\check{s}t_{KN}(b_{i-n+1}^{i}) - d, 0)}{\check{s}t_{KN}(b_{i-n+1}^{i-1})} + \lambda(b_{i-n+1}^{i-1})P_{KN}(b_i|b_{i-n+2}^{i-1})$$

$$\check{s}t_{KN}(\bullet) = \begin{cases} \check{s}tevilo(\bullet) & za\ višje\ stopnje\ n\text{-}gramov \\ \check{s}tevilo_{nadaljevanja}(\bullet) & za\ n\text{-}grame\ 1.\ stopnje \end{cases}$$

$\check{s}tevilo_{nadaljevanja}$ je število unikatnih samostojnih besed v kontekstu z $\bullet$. Način izračuna je prikazan na enem od prejšnjih slajdov.

Viri

- Dan Jurafsky, Chris Manning, Natural Language Processing `http://web.stanford.edu/~jurafsky/NLPCourseraSlides.html`
- Mirjam Sepesy Maučec, Statistično jezikovno modeliranje pri razpoznavanju govora, Center za interdisciplinarne in multidisciplinarne raziskave in študije, Univerza v Mariboru, Razlagova 22, 2000 Maribor, Slovenija

Popravljanje pravopisa

(angl. Spelling Correction)

Pravopisne naloge

- Zaznavanje pravopisnih napak
- Popravljanje pravopisnih napak
  - Samodejno popravljanje (an $\rightarrow$ na)
  - Predlaganje popravka
  - Predlaganje seznama popravkov

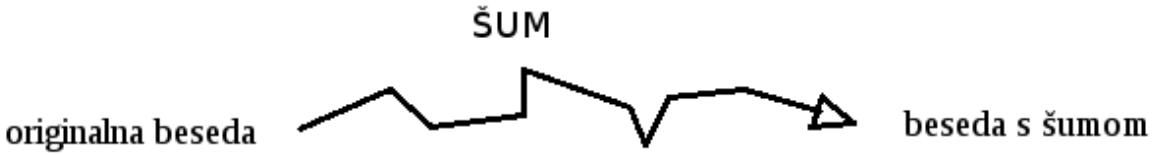
Tipi pravopisnih napak

- Nebesedne napake (angl. *non-word errors*)
  - marma $\rightarrow$ karma
- Besedne napake (angl. *real-word errors*)
  - Tipografske napake (angl. *typographical erros*)
    - praksa $\rightarrow$ praska
  - Kognitivne napake (angl. *cognitive errors*)
    - rob $\rightarrow$ rop
    - trk $\rightarrow$ trg

Nebesedne pravopisne napake

- angl. *non-word spelling errors*
- Zaznavanje nebesednih pravopisnih napak
  - vsaka beseda, ki ni v slovarju, je napaka
  - večji je slovar, boljše je
- Popravljanje nebesednih pravopisnih napak
  - Ustvarimo kandidatke $\rightarrow$ to so pravilne besede, ki so podobne napačni besedi
  - Izberemo tisto, ki je najboljša:
    - Najkrajša razdalja urejanja z utežmi
    - Največja verjetnost šumnega kanala

Šumni kanal (angl. *Noisy channel*)



x - napačno črkovana beseda

Poiščemo pravilno besedo b:

$$\hat{b} = \arg\max_{b \in V} P(b \mid x) = \arg\max_{b \in V} \frac{P(x \mid b)P(b)}{P(x)} = \arg\max_{b \in V} P(x \mid b)P(b)$$

Generiranje kandidatk

Na dva načina:

- Besede s podobnim črkovanjem
- Besede s podobno izgovorjavo
- 80% napak je na razdalji urejanja 1
- Skoraj vse napake so na razdalji urejanja 2
- Dovoljuje vstavljanje presledkov in stičnih vezajev
  - pridisem → pridi sem
  - črnobelo → črno-belo

Damerau-Levenshtein razdalja urejanja

Imamo besedo *plot* in nekaj primerov besed z razdaljo 1:

- Transpozicija (angl. *Transposition*): p**ol**t → p**lo**t
- Vstavljanje (angl. *Inserton*): pot → p**l**ot
- Brisanje (angl. *Deletion*): p**i**lot → plot
- Zamenjava (angl. *Substitution*): plo**d** → plo**t**

Jezikovni model (angl. *Language model*)

- Unigram, bigram, trigram
- Stupid backoff
- Ostali algoritmi za modeliranje jezika

Verjetnost urejanja

- Ustvarimo matrike zamenjav za vse štiri operacije (brisanje, vstavljanje, zamenjava, transpozicija)
- Ustvarimo matrike zamenjav za znakovne unigrame in bigrame

$$\mathit{brisanje}[a, b] = \text{število, kolikokrat se ab pojavi kot b}$$

$$\mathit{vstavljanje}[a, b] = \text{število, kolikokrat se a pojavi kot ab}$$

$$\mathit{zamenjava}[a, b] = \text{število, kolikokrat se a pojavi kot b}$$

$$\mathit{transpozicija}[a, b] = \text{število, kolikokrat se ab pojavi kot ba}$$

$$\text{š}\mathit{tevilo}[a] = \text{število, kolikokrat se pojavi znakovni unigram a}$$

$$\text{š}\mathit{tevilo}[ab] = \text{število, kolikokrat se pojavi znakovni bigram ab}$$

Kanalni model (angl. *Channel model*)

- Narobe črkovana beseda x $= x_1, x_2, x_3, ..., x_m$
- Pravilno črkovana beseda b $= b_1, b_2, b_3, ..., b_n$

Verjetnost urejanja P(x | b) izračunamo kot:

$$P(x \mid b) = \begin{cases} \frac{brisanje[b_{i-1},\ b_i]}{število[b_{i-1}\ \ b_i]} & \text{, če je brisanje} \\ \frac{vstavljanje[b_{i-1},\ x_i]}{število[b_{i-1}]} & \text{, če je vstavljanje} \\ \frac{zamenjava[x_i,\ b_i]}{število[b_i]} & \text{, če je zamenjava} \\ \frac{transpozicija[b_i,\ b_{i+1}]}{število[b_i\ \ b_{i+1}]} & \text{, če je transpozicija} \end{cases}$$

Besedne pravopisne napake

- angl. *real-word spelling errors*
- Za vsako besedo v povedi ustvarimo množico kandidatk:
  - Trenutna beseda
  - Poiščemo vse besede, ki se od trenutne razlikujejo za en znak in so besede določenega jezika
  - Enakoglasnica (angl. *homophones*) - upoštevamo izgovorjavo
- Izberemo najboljšo kandidatko:
  - Model šumnega kanala (angl. *noisy channel model*)
  - Klasifikator za specifične naloge (angl. *task-specific classifier*)

Model kanala

Nekaj faktorjev, kateri lahko vplivajo na verjetnost pravopisnih napak:

- Izvorna črka
- Ciljna črka
- Črke v okolici
- Položaj v besedi
- Tipke na tipkovnici, ki so si med seboj blizu
- Homologija na tipkovnici (razlike med tipkovnicami)
- Izgovorjava

Šumni kanal (angl. *Noisy channel*)

- Imamo poved $beseda_1, beseda_2, ..., beseda_n$
- Ustvarimo množico kandidatk za vsako besedo $beseda_n$:
  - $kandidatka(beseda_1) = \{beseda_1, beseda'_1, beseda''_1, ...\}$
  - $kandidatka(beseda_2) = \{beseda_2, beseda'_2, beseda''_2, ...\}$
  - $kandidatka(beseda_n) = \{beseda_n, beseda'_n, beseda''_n, ...\}$
- Izberemo tisto sekvenco besed B iz nabora kandidatk, ki maksimizira P(B)
- Poenostavitev (angl. *Simplification)*
  - Izmed vseh možnih povedi, kjer smo zamenjali samo eno besedo, izberemo tisto sekvenco B, ki maksimizira P(B)

Izračun verjetnosti

- Jezikovni model
  - unigram
  - bigram
  - itd.
- Kanalni model
  - Enako kot pri nebesednih pravopisnih napakah
  - Dodatno upošteva verjetnost za pravilne besede P(b | b), ki so odvisne od aplikacije
    - 0.90 (1 napaka na 10 bessed)
    - 0.95 (1 napaka na 20 besed)
    - 0.99 (1 napaka na 100 besed)
    - 0.995 (1 napaka na 200 besed)

Najsodobnejši šumni kanal

- angl. *state-of-the-art noisy channel*
- Nikoli ne množimo samo verjetnosti modela kanala in jezikovnega modela
- Predpostavka neodvisnosti $\rightarrow$ verjetnosti niso sorazmerne
- Utežimo jih:

$$\hat{b} = \underset{b \ \epsilon \ V}{\arg\max} \ P(x \mid b) \ P(b)^{\lambda}$$

- λ dobimo s pomočjo razvojne množice

Vir

- Dan Jurafsky, Chris Manning, Natural Language Processing
  `http://web.stanford.edu/~jurafsky/NLPCourseraSlides.html`

Klasifikacija besedila

(angl. Text Classification)

Ali je to nezaželena pošta?

mate presegajo meje svojega nabiralnika kvoto. Ne boste mogli pošiljati ali prejemati nova e-pošta, dokler ne boste povečali svoj nabiralnik Velikost,**Kliknite Tukaj** nadgraditi svoj račun.

Tehnična podpora

Klasifikacija elektronske pošte

Imamo 5 učnih dokumentov in 1 testni dokument.

Dokument	Zadeva	Sporočilo	Razred
d1	Must read	Get Viagra cheap	**spam**
d2	Gotta see this	Viagra you can get it at cut rates	**spam**
d3	Call me tomorrow	We need to talk about scheduling call me	**ni spam**
d4	That was hilarious	Just saw that link you sent me	**ni spam**
d5	dinner at 7	I got us a reservation tomrrow at 7	**ni spam**
d6	**See it to believe it**	**Best rates you will see**	**?**

Ali je testni dokument d6 spam ali ne?

Klasifikacija elektronske pošte

$$\mathrm{P}(\text{razred}) = \frac{N_{razred}}{N}$$
$$\mathrm{P}(\text{beseda} \mid \text{razred}) = \frac{frekvenca(beseda,razred)+1}{frekvenca(razred)+V}$$

$$P(spam) = \frac{2}{5}$$
$$P(ni\ spam) = \frac{3}{5}$$
$$\text{frekvenca}(\text{spam}) = 16$$
$$\text{frekvenca}(\text{ni spam}) = 32$$
$$\mathrm{V} = 37$$

Klasifikacija elektronske pošte

$$P(see \mid spam) = \frac{1+1}{16+37} = \frac{2}{53}$$
$$P(it \mid spam) = \frac{1+1}{16+37} = \frac{2}{53}$$
$$P(to \mid spam) = \frac{0+1}{16+37} = \frac{1}{53}$$
$$P(believe \mid spam) = \frac{0+1}{16+37} = \frac{1}{53}$$
$$P(it \mid spam) = \frac{1+1}{16+37} = \frac{2}{53}$$
$$P(best \mid spam) = \frac{0+1}{16+37} = \frac{1}{53}$$
$$P(rates \mid spam) = \frac{1+1}{16+37} = \frac{2}{53}$$
$$P(you \mid spam) = \frac{1+1}{16+37} = \frac{2}{53}$$
$$P(will \mid spam) = \frac{0+1}{16+37} = \frac{1}{53}$$
$$P(see \mid spam) = \frac{1+1}{16+37} = \frac{2}{53}$$

Klasifikacija elektronske pošte

$$P(see \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$
$$P(it \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$
$$P(to \mid ni\ spam) = \frac{1+1}{32+37} = \frac{2}{69}$$
$$P(believe \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$
$$P(it \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$
$$P(best \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$
$$P(rates \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$
$$P(you \mid ni\ spam) = \frac{1+1}{32+37} = \frac{2}{69}$$
$$P(will \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$
$$P(see \mid ni\ spam) = \frac{0+1}{32+37} = \frac{1}{69}$$

Klasifikacija elektronske pošte

$$P(d6 \mid spam) = \frac{2}{5} \cdot \frac{2}{53} \cdot \frac{2}{53} \cdot \frac{1}{53} \cdot \frac{1}{53} \cdot \frac{2}{53} \cdot \frac{1}{53} \cdot \frac{2}{53} \cdot \frac{2}{53} \cdot \frac{1}{53} \cdot \frac{2}{53} = \frac{2}{5} \cdot 3.66e^{-16}$$
$$P(d6 \mid ni\ spam) = \frac{3}{5} \cdot \frac{1}{69} \cdot \frac{1}{69} \cdot \frac{2}{69} \cdot \frac{1}{69} \cdot \frac{1}{69} \cdot \frac{1}{69} \cdot \frac{1}{69} \cdot \frac{2}{69} \cdot \frac{1}{69} \cdot \frac{1}{69} = \frac{3}{5} \cdot 1.64e^{-18}$$

Dokument d6 se klasificira kot spam.

Pohitritev

Kot vidimo, so vrednosti zelo majhne.
Z uporabo logaritma se lahko izognemo računanja z vrednostmi blizu nič (angl. *Underflow*):

$$P(poved) = log(P_1 \cdot P_2 \cdot .. \cdot P_n) = logP_1 + logP_2 + .. + logP_n$$

Tudi sicer je operacija seštevanja hitrejša kot operacija množenja.

Izračun z uporabo logaritma

$$P(d6 \mid spam) = log\frac{2}{5} + log\frac{2}{53} + log\frac{2}{53} + log\frac{1}{53} +$$
$$log\frac{1}{53} + log\frac{2}{53} + log\frac{1}{53} + log\frac{2}{53} + log\frac{2}{53} + log\frac{1}{53} + log\frac{2}{53} = log\frac{2}{5} - 15.4$$
$$P(d6 \mid ni\ spam) = log\frac{3}{5} + log\frac{1}{69} + log\frac{1}{69} + log\frac{2}{69} +$$
$$log\frac{1}{69} + log\frac{1}{69} + log\frac{1}{69} + log\frac{1}{69} + log\frac{2}{69} + log\frac{1}{69} + log\frac{1}{69} = log\frac{3}{5} - 17.8$$

Katerem naravnem jeziku pripada besedilo?

- Osnovna naloga procesiranja besedil.
- Omogoča obdelavo velike količine digitalnih dokumentov.
- Na osnovi klasifikacije se zmanjša napaka oz. izboljša učinkovitost algoritmov procesiranja besedila.
  (npr.: črkovalnik, ugotavljanje slovničnih napak, razpoznavanje besedila itd.)

Katerem naravnem jeziku pripada besedilo?

- Predstavljena bo metoda, ki temelji na znakovnih n-gramih.
- Lastnosti: hitrost in robustnost.
- Točnost metode je večja od 80%. V določenih primerih tudi večja od 99%.
- Temelji na ugotavljanju frekvenc n-gramov.

Katerem naravnem jeziku pripada besedilo?

Lastnosti klasifikatorja besedil

- Neobčutljiv na napake v besedilu.
- Učinkovitost (prostorska in časovna)
- Zmožen ugotoviti, da dokument ne pripada nobenemu razredu.
- Zmožen ugotoviti, da je dokument na meji med dvema razredoma.

Znakovni n-grami

- Del besedila, ki je sestavljen iz N-znakov.
- Vsebujejo nevidne znake.
  - Na začetku in koncu besede.
  - Ločevanje besed.
- Primer za: _besedilo_
  - bigrami: _b,be, es, se, ed, di, il, lo, o_
  - trigrami: _be, bes, ese, sed, edi, dil, ilo, lo_
  - štirigrami: _bes, bese, esed, sedi, edil, dilo, ilo_

Klasifikacija - frekvenca znakovnih n-gramov

- Za razlago ideje uporabimo Zipfov zakon
  - V vsakem naravnem jeziku je pogostost n-te najpogosteje uporabljane besede približno recipročno odvisna od n.
- V vsakem jeziku dominirajo frekvence določenih besed.
- Podobno lahko povzamemo za n-grame v besedilu.
- Besedila, ki sodijo v določen razred bodo imela verjetno podobno frekvenco besed.

Algoritem klasifikacije besedila

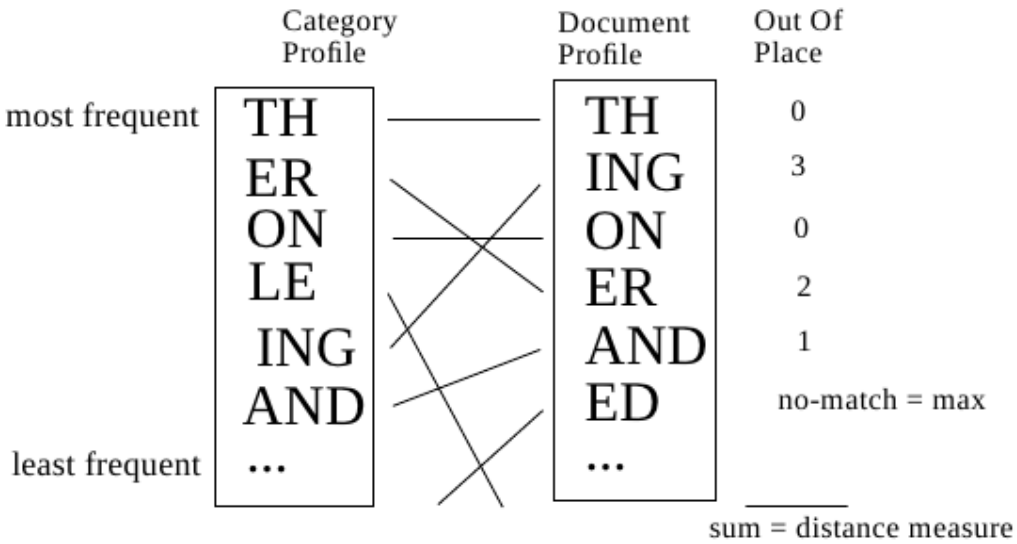
- Besedilo razdelimo na leksikalne simbole. Pri tem odstranimo posebne znake in števila. Vsakemu leksikalnemu simbolu dodamo na začetku in koncu presledek.
- Za vsak leksikalni simbol ustvarimo n-grame ($N \in \{1, \ldots, 5\}$).
- V sekljani tabeli shranjujte n-grame in štejte njihove pojavitve. Sekljana tabela vsebuje klasičen mehanizem, ki zagotavlja, da so v primeru kolizij vsi n-grami shranjeni.
- N-grame uredite po njihovih števcih-frekvencah padajoče in jih shranite skupaj v profil. Ponavadi shranimo 300 najbolj frekvenčnih n-gramov.

Algoritem klasifikacije besedila

- 300 najbolj frekvenčnih n-gramov ponavadi dovolj opisuje določen jezik.
- Najbolj frekvenčni n-grami so uni-grami in predstavljajo korelacijo med črkami. Nato sledijo pogoste predpone (angl. *prefix*) in pripone (angl. *sufix*) besed.
- Dolgi in frekvenčni n-grami se uvrstijo na rep seznama 300 najbolj frekvenčnih n-gramov.
- N-grami v okolici 300. mesta najbolj frekvenčnih n-gramov predstavljajo n-grame, ki so specifični za področje dokumenta.

Primerjava profilov dokumentov

- Izračun vsote razdalj med rangi n-gramov.
- V primeru neujemanja n-gramov se uporabi neka velika vrednost (300).
- Dokument dodelimo v razred, kjer je razlika profilov najmanjša.



William B. Cavnar in John M. Trenkle, N-Gram-Based Text Categorization

Ugotovite spol avtorja besedila

- Al-Eksandra je intimna pripoved ženske, ki je kot devetnajstletno dekle zagledalo plakat eksotične dežele, spakiralo mali kovček in odšlo. Brez pomislekov je zapustila Slovenijo, vse, kar ji je bilo znano, in se prepustila toku življenja v tujini in neznani deželi. Deželi, polni nasprotij, deželi, kjer je desetletja potekala vojna - vse to je Libanon.
- Genom mojega življenja je avtobiografska pripoved dr. J. Craiga Venterja, kalifornijskega jadralca, ki ga je služba bolničarja v Vietnamu tako prekalila, da se je s trmasto vztrajnostjo prelevil v pionirja genomike. V knjigi je najnovejša dognanja genetike uporabil kar na samem sebi, ko je avtobiografsko pripoved prepletel s kratkimi razlagami delov svojega genoma, ki ga je leta 2007 tudi objavil.

Pozitivna ali negativna ocena filma

- ✘ Torej ... nekako nisem dojel filma, tako da je moja ocena žal negativna.
- ✔ Film lahko ocenim zelo pozitivno, saj je zgodba ves čas v dogajanju in napetosti.
- ✘ Kljub komercialnemu uspehu je film Melanie se poroči s strani kritikov prejel v glavnem negativne ocene.
- ✘ Na žalost je ta film popoln polom.
- ✘ Vsaj kar se mene tiče, pa tudi ocene kritikov niso najbolj pozitivne.
- ✔ Krasen film, vreden ogleda.
- ✔ Od filma sem pričakovala manj, verjetno me je zato tako pozitivno presenetil.

Področje članka

Članek

Evolution: A Comparative Study on Numerical Benchmark Problems

Janez Brest, *Member, IEEE*, Sašo Greiner, Borko Bošković, Marjan Mernik, *Member, IEEE*, and Viljem Žumer, *Member, IEEE*

***Abstract*—We describe an efficient technique for adapting control parameter settings associated with differential evolution (DE). The DE algorithm has been used in many practical cases and has demonstrated good convergence properties. It has only a few control parameters, which are kept fixed throughout the entire evolutionary process. However, it is not an easy task to properly set control parameters in DE. We present an algorithm—a new version of the DE algorithm—for obtaining self-adaptive control parameter settings that show good performance on numerical benchmark problems. The results show that our algorithm with self-adaptive control parameter settings is better than, or at least comparable to, the standard DE algorithm and evolutionary algorithms from literature when considering the quality of the solutions obtained.**

***Index Terms*—Adaptive parameter control, differential evolution (DE), evolutionary optimization.**

I. Introduction

DIFFERENTIAL evolution (DE) is a simple yet powerful evolutionary algorithm (EA) for global optimization introduced by Price and Storn [1]. The DE algorithm has gradually become more popular and has been used in many practical cases, mainly because it has demonstrated good convergence properties and is principally easy to understand [2].

EAs [3] are a broad class of stochastic optimization algorithms inspired by biology and, in particular, by those biological processes that allow populations of organizms to adapt to their surrounding environments: genetic inheritance and survival of the fittest. EAs have a prominent advantage over other types of numerical methods. They only require information about the objective function itself, which can be either explicit or implicit. Other accessory properties such as differentiability or continuity are not necessary. As such, they are more flexible in dealing with a wide spectrum of problems.

When using an EA, it is also necessary to specify how candidate solutions will be changed to generate new solutions [4]. EA may have parameters, for instance, the probability of mutation, the tournament size of selection, or the population size. The values of these parameters greatly determine the quality of the solution obtained and the efficiency of the search [5]–[7]. Starting with a number of guessed solutions, the multipoint algorithm updates one or more solutions in a synergistic manner in the hope of steering the population toward the optimum [8], [9].

Choosing suitable parameter values is, frequently, a problem-dependent task and requires previous experience of the user. Despite its crucial importance, there is no consistent methodology for determining the control parameters of an EA, which are, most of the time, arbitrarily set within some predefined ranges [4].

In their early stage, EAs did not usually include control parameters as a part of the evolving object but considered them as external fixed parameters. Later, it was realized that in order to achieve optimal convergence, these parameters should be altered in the evolution process itself [5], [7].

The control parameters were adjusted over time by using heuristic rules, which take into account information about the progress achieved. However, heuristic rules, which might be optimal for one optimization problem, might be inefficient or even fail to guarantee convergence for another problem. A logical step in the development of EAs was to include control parameters into the evolving objects and allow them to evolve along with the main parameters [3], [10], [11].

Globally, we distinguish two major forms of setting parameter values: parameter *tuning* and parameter *control*. The former means the commonly practiced approach that tries to find good values for the parameters before running the algorithm, then tuning the algorithm using these values, which remain fixed during the run. The latter means that values for the parameters are changed during the run. According to Eiben *et al.* [5], [7], the change can be categorized into three classes.

1) *Deterministic parameter control* takes place when the value of a parameter is altered by some deterministic rule.
2) *Adaptive parameter control* is used to place when there is some form of feedback from the search that is used to determine the direction and/or the magnitude of the change to the parameter.
3) *Self-adaptive parameter control* is the idea that "evolution of the evolution" can be used to implement the self-adaptation of parameters. Here, the parameters to be adapted are encoded into the chromosome (individuals) and undergo the actions of genetic operators. The better values of these encoded parameters lead to better individuals which, in turn, are more likely to survive and produce offspring and, hence, propagate these better parameter values.

Manuscript received June 14, 2005; revised September 19, 2005 and November 9, 2005. This work was supported in part by the Slovenian Research Agency under Programme P2-0041, Computer Systems, Methodologies, and Intelligent Services.

The authors are with the Computer Architecture and Languages Laboratory, Institute of Computer Science, Faculty of Electrical Engineering and Computer Science, University of Maribor, SI-2000 Maribor, Slovenia (e-mail: janez.brest@uni-mb.si; saso.greiner@uni-mb.si; borko.boskovic@uni-mb.si; marjan.mernik@uni-mb.si; zumer@uni-mb.si).

Digital Object Identifier 10.1109/TEVC.2006.872133

⟹

Področje

- Diferencialna evolucija
- Genetski algoritmi
- Roji delcev
- Mravlje
- Čebelice

Klasifikacija besedila

- Določanje vsebine, kategorije, teme ali žanra
- Odkrivanje nezaželene pošte
- Identificiranje avtorjev
- Ugotavljanje starosti/spola
- Identificiranje jezika (angl. *Language Identification*)
- Analiza sentimenta - odkrivanje mnenja, ki ga podaja neko besedilo (angl. *Sentiment analysis*)
- ...

Definicija klasifikacije besedila

- **Vhod**:
  - Dokument d
  - Fiksna množica razredov $C = \{c_1, c_2, ..., c_n\}$
- **Izhod**:
  - Izbran razred $c \in C$ v katerega sodi podan dokument d

Klasifikacijske metode - ročno določena pravila

- Pravila, ki določajo kombinacijo zaporedja besed ali drugih lastnosti
  - Nezaželena pošta: naslov iz črnega seznama ali sporočilo vsebuje naslednji zaporedji besed: “nagrada” in “bili ste izbrani”
- Točnost lahko povečamo
  - Izpopolnimo pravila s pomočjo strokovnjakov
- Gradnja in vzdrževanje teh pravil je drago opravilo

Klasifikacijske metode - nadzorovano strojno učenje

- **Vhod**:
  - Dokument d
  - Fiksna množica razredov $C = c_1, c_2, ..., c_n$
  - Učna množica, ki vsebuje m ročno klasificiranih dokumentov $(d_1, c_1), (d_2, c_2), ..., (d_m, c_m)$
- **Izhod**:
  - Naučen klasifikator $\gamma : d \rightarrow c$
- Klasifikatorji:
  - Naïve Bayes
  - Logistična regresija
  - Support–vector machines
  - K-najbližjih sosedov

Naïve Bayes

- Enostavni ("Naïve") klasifikator, ki temelji na pravilih Bayes
- Temelji na zelo preprosti predstavitvi dokumenta
  - Vreča besed

Vreča besed

Vreča

Sem ga ravnokar pogledal in ga toplo priporočam vsakomur. Misterioznost, napeta zgodba, odlična stripovska podlaga. Ta film bo najbrž eden boljših letos. Še ena izmed meni osebno najljubših tem - alternativna zgodovina oz. anti-utopična družbena ureditev. Zgodba je odlična, čeprav nekoliko ozka, na ogromno vprašanj si mora gledalec odgovoriti sam.

$$\gamma(\quad) = C$$

Vreča besed

Vreča

Sem ga ravnokar pogledal in ga **toplo priporočam** vsakomur. Misterioznost, napeta zgodba, **odlična** stripovska podlaga. Ta film bo najbrž eden **boljših** letos. Še ena izmed meni osebno najljubših tem - alternativna zgodovina oz. anti-utopična družbena ureditev. Zgodba je **odlična**, čeprav nekoliko ozka, na ogromno vprašanj si mora gledalec odgovoriti sam.

$$\gamma(\quad) = C$$

Vreča besed

$$\gamma(\quad) = C$$

Vreča

Xxx xx xxxxxxxx xxxxxxxx xx xx **toplo priporočam** xxxxxxxx. Xxxxxxxxxxxxx, xxxxxx xxxxxx, **odlična** xxxxxxxxxx xxxxxxx. Xx xxxx xx xxxxxx xxxx **boljših** xxxxx. Xx xxx xxxxx xxxx xxxxxx **najljubših** xxx - xxxxxxxxxxxx xxxxxxxxx xx. xxxx-xxxxxxxx xxxxxxxx xxxxxxxx. Xxxxxx xx **odlična**, xxxxxx xxxxxxxx xxxx, xx xxxxxxx xxxxxxxx xx xxxx xxxxxxxx xxxxxxxxxx xxx.

Vreča besed

Vreča

$$\gamma\left(\begin{array}{|c|c|}\hline \text{odlična} & 2 \\ \text{toplo} & 1 \\ \text{priporočam} & 1 \\ \text{boljših} & 1 \\ \text{najljubših} & 1 \\ \hline\end{array}\right) = C$$

Pravila Bayes - dokumenti in razredi

- Za dokument d in razred c

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Klasifikator Naïve Bayes

$$c_{MAP} = \arg\max_{c \in C} P(c|d)$$

Pravila Bayes

$$= \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

P(d) je za vse razrede enak in ga lahko odstranimo

$$= \arg\max_{c \in C} P(d|c)P(c)$$

Uporabimo lastnosti dokumenta

$$= \arg\max_{c \in C} P(x_1, x_2, ..., x_n|c)P(c)$$

MAP je "maximum a posteriori" = najbolj verjetni razred

Multinominalni Naïve Bayes

Predpostavke neodvisnosti

$$P(x_1, x_2, ..., x_n|c)$$

- Predpostavka vreče besed: položaj besed ni pomemben
- Pogojna neodvisnost: za določen razred c so verjetnosti posameznih lastnosti $P(x_i|c_j)$ neodvisne

$$P(x_1, x_2, ..., x_n|c) = P(x_1|c) \cdot P(x_2|c) \cdot ... \cdot P(x_n|c)$$

Multinominalni Naïve Bayes

$$c_{MAP} = \arg\max_{c \in C} \; P(x_1, x_2, ..., x_n|c)P(c)$$

$$c_{NB} = \arg\max_{c \in C} \; P(c) \prod_{x \in X} P(x|c)$$

Multinominalni Naïve Bayes

Klasifikacija besedila

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in \text{položaj}} P(x_i|c_j)$$

Položaj vsebuje vse položaje besed v besedilu

Učenje modela Multinominal Naïve Bayes

- Ocenjevanje maksimalne verjetnosti
  - Preprosto uporabimo frekvence podatkov

$$\hat{P}(c_j) = \frac{\text{številoDokumentov}(C = c_j)}{\text{številoDokumentov}}$$

$$\hat{P}(b_i|c_j) = \frac{\text{število}(b_i, c_j)}{\sum_{b \in V} \text{število}(b, c_j)}$$

  - Vse dokumente, ki pripadajo določenemu razredu združimo v "mega" dokument
    - Na osnovi "mega" dokumenta se računajo frekvence besed

Metode maksimalnega verjetja - problem

- Problemi z ne videnimi besedami

$$\hat{P}(\text{“fantastično”}|\text{pozitivno}) = \frac{\text{število}(\text{“fantastično”},\text{pozitivno})}{\sum_{b \in V} \text{število}(b, \textit{pozitivno})} = 0$$

$$c_{MAP} = \arg\max_{c} \hat{P}(c) \prod_{i} \hat{P}(x_i|c)$$

Glajenje Laplace (add-1) za Naïve Bayes

$$\hat{P}(b_i|c) = \frac{\text{število}(b_i, c) + 1}{\sum_{b \in V}(\text{število}(b, c) + 1)}$$

$$= \frac{\text{število}(b_i, c) + 1}{(\sum_{b \in V} \text{število}(b, c)) + |V|}$$

Multinominal Naïve Bayes - učenje

- Iz učnega korpusa se izlušči slovar (angl. *Vocabulary*)
- Izračunajo se $P(c_j)$
  - $dok_j \leftarrow$ vsi dokumenti razredac_j
  - $P(c_j) = \frac{|dok_j|}{|\text{število vseh dokumentov}|}$
- Izračunajo se $P(b_k|c_j)$
  - $besedilo_j \leftarrow$ dokument, ki vsebuje vse dokumente dok_j
  - Za vsako besedo b_k iz slovarja
    - $n_k \leftarrow$ število pojavitev besede b_k v $besedilo_j$
    - $P(b_k|c_j) \leftarrow \frac{n_k+\alpha}{n+\alpha|slovar|}$
    - n - število leksikalnih simbolov v razredu c_j

Glajenje Laplace (add-1): neznane besede

- V slovar dodamo neznano besedo $b_{neznana}$

$$\hat{P}(b_{neznana}|c) = \frac{\text{število}(b_{neznana}, c) + 1}{(\sum_{b \in V} \text{število}(b, c)) + |V + 1|}$$

$$= \frac{1}{(\sum_{b \in V} \text{število}(b, c)) + |V + 1|}$$

Naïve Bayes in jezikovni modeli

- Naïve Bayes lahko uporablja različne značilke.
  - URL, E-pošta, slovarji, značilke omrežja itd.
- V našem primeru besede predstavljajo značilke.
- Podobnost klasifikatorja Naïve Bayes z jezikovnimi modeli
  - Vsak razred lahko enačimo z jezikovnim modelom (unigram).
  - Vsaki besedi dodelimo verjetnost $P(beseda|c)$.
  - Vsakemu stavku dodelimo verjetnost $P(stavek|c) = \prod P(beseda|c)$.

Lastnosti metode Naïve Bayes

- Mala časovna zahtevnost
- Mala prostorska zahtevnost
- Robusten na nepomembne lastnosti
  - Nepomembne lastnosti ne vplivajo na končni rezultat
- Je dobra metoda za klasifikacijo beselida
  - Obstajajo boljše metode

Kako oceniti uspešnost klasifikacije besedila

podatki / klasifikacija	nezaželena pošta	ni nezaželena pošta
nezaželena pošta	resnični pozitivni (tp)	lažno pozitivni (fp)
ni nezaželena pošta	lažno negativni (fn)	resnični negativni (tn)

- Točnost (angl. *Accuracy*): $\frac{tp+tn}{tp+tn+fp+fn}$

Kako oceniti uspešnost klasifikacije besedila

	znamke avtomobilov	ostalo
znamke avtomobilov	0 (tp)	0 (fp)
ostalo	10 (fn)	99.990 (tn)

- Točnost (angl. *Accuracy*):$Acc = \frac{tp+tn}{tp+tn+fp+fn} = \frac{99.990}{100.000} = 99,99\%$
- Vrednost ne izraža kvalitete klasifikatorja
- Preciznost (angl. *Precision*): razmerje med številom vseh pozitivnih primerov $\frac{tp}{tp+fp} = 0\%$
- Priklic (angl. *Recall*): razmerje med vsemi pozitivnimi primeri, ki smo jih pravilno napovedali in številom vseh primerov, za katere smo napovedali, da pripadajo pozitivnemu razredu $\frac{tp}{tp+fn} = 0\%$

Kako oceniti uspešnost klasifikacije besedila

	znamke avtomobilov	ostalo
znamke avtomobilov	8 (tp)	32 (fp)
ostalo	2 (fn)	99.960 (tn)

- Točnost (angl. *Accuracy*):$Acc = \frac{tp+tn}{tp+tn+fp+fn} = \frac{99.968}{100.002} = 99,99\%$
- Vrednost ne izraža kvalitete klasifikatorja
- Preciznost (angl. *Precision*): razmerje med številom vseh pozitivnih primerov $P = \frac{tp}{tp+fp} = \frac{8}{8+32} = 20\%$
- Priklic (angl. *Recall*): razmerje med vsemi pozitivnimi primeri, ki smo jih pravilno napovedali in številom vseh primerov, za katere smo napovedali, da pripadajo pozitivnemu razredu $R = \frac{tp}{tp+fn} = \frac{8}{10} = 80\%$
- Potrebujemo kompromis med preciznostjo in priklicem.

Mera F

- Kompromis med preciznostjo in priklicem (harmonično uteženo povprečje).
  - $F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$
- Ponavadi se uporablja uglašena mera F1:
  - npr. $\beta = 1$ oz. $\alpha = \frac{1}{2}$; $F = \frac{2PR}{P+R}$

Mikro in makro povprečenje

- angl. *Micro vs. Macro Averaging*
- Če imamo več kot eden razred, kako oceniti uspešnost klasifikatorja z eno vrednostjo?
- Makro povprečenje: izračunamo uspešnost za vsak razred posebej in nato izračunamo povprečje uspešnosti.
- Mikro povprečenje: Združimo dobljene rezultate klasifikatorja in na osnovi teh podatkov izračunamo uspešnost.

Mikro in makro povprečenje

	razred 1	ostalo
razred 1	10 (tp)	10 (fp)
ostalo	10 (fn)	970 (tn)

	razred 2	ostalo
razred 2	90 (tp)	10 (fp)
ostalo	10 (fn)	890 (tn)

	razed sum	ostalo
razred sum	100 (tp)	20 (fp)
ostalo	20 (fn)	1860 (tn)

- Makro povprečenje preciznosti: $\frac{\frac{10}{20}+\frac{90}{100}}{2} = 0,7$
- Mikro povprečenje preciznosti: $\frac{100}{120} = 0,83$
- Pri mikro povprečenju prevladujejo rezultati splošnih razredov.

Razvojna množica in navzkrižna validacija

- Metrike: P/R/F1
- Učna množica
- Ne videna testna množica
- Navzkrižna validacija skozi večkratno razdelitev učne množice in razvojne množice

Izgradnja realnega klasifikatorja

- Ni podatkov
  - Ročno
  - Velika časovna zahtevnost
- Na voljo zelo malo podatkov
  - Metoda Naïve Bayes
- Ustrezna količina podatkov
  - SVM (angl. *Support Vector Machines*)
  - Odločitvena drevesa
  - Urejena logistična regresija (angl. *Regularized Logistic Regression*)
- Enormna količina podatkov
  - Metoda Naïve Bayes

Premajhna števila

- Z množenjem verjetnosti, lahko dobimo zelo mala števila.
- $log(x \cdot y) = log(x) + log(y)$
- Klasifikator:
  $C_{NB} = \arg\max log\ P(C_j) + \sum_{i \in pozicija} log\ P(x_i|c_j)$

Kako izboljšati učinkovitost klasifikatorja

- Domensko specifične značilke in uteži
- Dodatno obtežiti določene besede
  - Besede v naslovu
  - Prvi stavek vsakega odstavka
  - Stavke, ki vsebujejo besede iz naslova
- Določene izraze zanemarimo
  - števila, enačbe, itd.

Viri

- Dan Jurafsky, Chris Manning, Natural Language Processing http://web.stanford.edu/∼jurafsky/NLPCourseraSlides.html
- Monika Bozhinova, NAIVNI BAYESOV KLASIFIKATOR, Diplomsko delo, 2015.
- William B. Cavnar in John M. Trenkle, N-Gram-Based Text Categorization

Primeri nalog

Poglavja

- Pregled jezikovnih tehnologij
- Osnovno procesiranje besedila
- Razdalja urejanja
- Modeliranje jezika
- Popravljanje pravopisa
- Klasifikacija

Primeri nalog

- 1. Napišite sekvenco ukazov v lupini bash za izračun frekvenc besed, ki vsebujejo šumnike. Izpis frekvenc naj bo padajoč. Pri tem upoštevajte, da so pri kodiranju UTF-8 šumniki predstavljeni z dvema znakoma.
- cat jrc-acquis.sl | sed s/[^A-Za-zČŠŽčšž]/\\n/g | grep [ČŠŽčšž] | sort | uniq -c | sort -rn > sequence.txt
- Rezultati
  - 114366 št
  - 110294 Člen
  - 91391 člena
  - 71652 članice
  - 68468 države
  - 67203 upoštevanju
  - 43287 če
  - 42985 držav
  - 38527 pomoči
  - 32886 členom
  - 28635 členu

Primeri nalog

- Pokažite način delovanja algoritma za minimalno razdaljo urejanja in poravnavo besed banana in salama.

	/	s	a	l	a	m	a
/	0	1	2	3	4	5	6
b	1	2 z	3	4	5	6	7
a	2	3	2 -	3	4	5	6
n	3	4	3	4 z	5	6	7
a	4	5	4	5	4 -	5	6
n	5	6	5	6	5	6 z	7
a	6	7	6	7	6	7	6 -

Naloga

- Imamo dva jezikovna modela M1 in M2. Za oba modela izračunajte perpleksnost in povejte kateri model je boljši.
- Poved: <s> Strupen jezik je nevarnejši od strupene kače </s>
- M1:
  P(Strupen| <s>)=$\frac{1}{13}$, P(jezik|Strupen)=$\frac{1}{14}$, P(je|jezik)=$\frac{1}{15}$,
  P(nevarnejši|je)=$\frac{2}{29}$, P(od|nevarnejši)=$\frac{2}{25}$, P(strupene|od)=$\frac{1}{79}$;
  P(kače|strupene)=$\frac{4}{41}$, P(</s> |kače)=$\frac{1}{128}$
- M2:
  P(Strupen| <s>)=$\frac{7}{10}$;P(jezik|Strupen)=$\frac{2}{10}$;P(je|jezik)=$\frac{1}{55}$;
  P(nevarnejši|je)=$\frac{3}{35}$, P(od|nevarnejši)=$\frac{4}{25}$;P(strupene|od)=$\frac{5}{44}$;
  P(kače|strupene)=$\frac{3}{13}$;P(</s> |kače)=$\frac{1}{99}$
- $PP(S_{M1}) = (\frac{1}{13} \cdot \frac{1}{14} \cdot \frac{1}{15} \cdot \frac{2}{29} \cdot \frac{2}{25} \cdot \frac{1}{79} \cdot \frac{4}{41} \cdot \frac{1}{128})^{-\frac{1}{8}} = 21,81475$
- $PP(S_{M2}) = (\frac{7}{10} \cdot \frac{2}{10} \cdot \frac{1}{55} \cdot \frac{3}{35} \cdot \frac{4}{25} \cdot \frac{5}{44} \cdot \frac{3}{13} \cdot \frac{1}{99})^{-\frac{1}{8}} = 10,09834$
- Manjša perpleksnost nakazuje boljši model.

Naloga

- <s>med igralci maribora ni bilo branilcev, pri olimpiji pa nosilcev</s>
  <s>trener olimpije je moral poseči po mladih močeh</s>
  <s>tekmo so z nekaj obetavnimi akcijami začeli igralci olimpije</s>
  <s>igralci maribora so resneje zapretili v 13 minuti</s>
  <s>mariborčani bi lahko povedli tudi v 22 minuti</s>
  <s>v 30 minuti je skomina pokazal na belo točko</s>
  <s>do vodstva maribora ni prišlo</s>
- Z uporabo modela bigram in glajenjem Add-1 izračunajte, kateri stavek ima večjo verjetnost, da bo izbran:
  - S_1 :<s>igralci maribora bodo prvaki</s>
  - S_2 :<s>igralci olimpije bodo prvaki</s>
- Velikost slovarja: 47

Naloga

- $P(\text{igralci}|<s>)=\frac{1+1}{7+47}$
- $P(\text{maribora}|\text{igralci})=\frac{2+1}{3+47}$
- $P(\text{bodo}|\text{maribora})=\frac{0+1}{3+47}$
- $P(\text{prvaki}|\text{bodo})=\frac{0+1}{0+47}$
- $P(</s>|\text{prvaki})=\frac{0+1}{0+47}$
- $P(S_1) = \frac{1+1}{7+47} \cdot \frac{2+1}{3+47} \cdot \frac{0+1}{3+47} \cdot \frac{0+1}{0+47} \cdot \frac{0+1}{0+47} = 2,011971e-08$
- $P(\text{igralci}|<s>)=\frac{1+1}{7+47}$
- $P(\text{olimpije}|\text{igralci})=\frac{1+1}{2+47}$
- $P(\text{bodo}|\text{olimpije})=\frac{0+1}{2+47}$
- $P(\text{prvaki}|\text{bodo})=\frac{0+1}{0+47}$
- $P(</s>|\text{bodo})=\frac{0+1}{0+47}$
- $P(S_2) = \frac{1+1}{7+47} \cdot \frac{1+1}{2+47} \cdot \frac{0+1}{2+47} \cdot \frac{0+1}{0+47} \cdot \frac{0+1}{0+47} = 1,3966203171e-08$

Naloga

- Podan imate naslednji korpus:
  - <s>jaz sem ivan</s>
  - <s>moje ime je ivan</s>
  - <s>sem ivan</s>
  - <s>torej moje ime je ivan</s>
- S pomočjo interpoliranega glajenja Kneser-Ney izračunajte $P_{KN}(ivan|je)$. Pri tem upoštevajte, da je
  - $d = 1$
  - število$(je, ivan) = 2$
  - število$(je) = 2$
  - število$(ivan) = 4$
  - $|\{b : \text{število}(je, b) > 0\}| = 1$
  - $|\{(b_{j-1}, b_j) : \text{število}(b_{j-1}, b_j) > 0\}| = 11$
  - $|\{b_{i-1} : \text{število}(b_{i-1}, ivan) > 0\}| = 2$

Naloga

$$P_{KN}(b_i|b_{i-1}) = \frac{max(število(b_{i-1}, b_i) - d, 0)}{število(b_{i-1})} + \lambda(b_{i-1}) P_{nadaljevanja}(b_i)$$

$\lambda(b_{i-1})$ je normalizirana konstanta:

$$\lambda(b_{i-1}) = \frac{d}{število(b_{i-1})} |\{b : število(b_{i-1}, b) > 0)\}|$$

$$P_{nadaljevanja}(b) = \frac{|\{b_{i-1} : število(b_{i-1}, b) > 0\}|}{|\{(b_{j-1}, b_j) : število(b_{j-1}, b_j) > 0\}|}$$

$$P_{KN}(ivan|je) = \frac{max(2-1, 0)}{2} + \frac{1}{2} \cdot 1 \cdot \frac{2}{11} = \frac{1}{2} + \frac{2}{22}$$

Naloga

- S pomočjo rekurzivne enačbe Kneser-Ney izračunajte $P_{KN}(\check{s}tudent|je\ ivan)$.

$$P_{KN}(b_i|b_{i-n+1}^{i-1}) = \frac{max(\check{s}t_{KN}(b_{i-n+1}^{i}) - d, 0)}{\check{s}t_{KN}(b_{i-n+1}^{i-1})} + \lambda(b_{i-n+1}^{i-1}) P_{KN}(b_i|b_{i-n+2}^{i-1})$$

$$\check{s}t_{KN}(\bullet) = \begin{cases} \check{s}tevilo(\bullet) & za\ višje\ stopnje\ n\text{-}gramov \\ \check{s}tevilo_{nadaljevanja}(\bullet) & za\ n\text{-}grame\ 1.\ stopnje \end{cases}$$

Kako se izognemo ničelnim vrednostim:

- **V fazi učenja vse n-grame, ki se ponovijo zelo malo krat, obravnavamo kot neznane n-grame. Informacijo o neznanih n-gramih nato uporabimo za izračun verjetnosti neznanih n-gramov.**
- $P_{KN}(\mathrm{b}) = P_{nadaljevanja}(\mathrm{b}) + \lambda(\varepsilon)P(\varepsilon)$
  $P(\varepsilon) = \frac{1}{|V|}; \quad \lambda(\varepsilon) = d$

Naloga

$$P_{KN}(\text{študent}|\textit{je ivan}) = \frac{max(\text{število}(\text{je ivan študent}) - d, 0)}{\text{število}(\text{je ivan})} + \lambda(\text{je ivan}) *$$

$$\left(\frac{max(\text{število}(\text{ivan študent}), 0)}{\text{število}(\text{ivan})} + \lambda(\textit{ivan}) * \right.$$

$$\left. \left(P_{nadaljevanja}(\text{študent}) + \lambda(\varepsilon) P(\varepsilon) \right) \right) =$$

$$\frac{max(\text{število}(\text{je ivan študent}) - d, 0)}{\text{število}(\text{je ivan})} + \frac{d * |\{b : \text{število}(\textit{je}, \textit{ivan}, b) > 0\}|}{\text{število}(\text{je ivan})} *$$

$$\left(\frac{max(\text{število}(\text{ivan študent}) - d, 0)}{\text{število}(\text{ivan})} + \frac{d * |\{b : \text{število}(\textit{ivan}, b) > 0\}|}{\text{število}(\textit{ivan})} * \right.$$

$$\left. \left(\frac{|\{b_{i-1} : \textit{število}(b_{i-1}, \text{študent} > 0\}|}{|\{(b_{j-1}, b_j) : \textit{število}(b_{j-1}, b_j) > 0\}|} + d * \frac{1}{|V|} \right) \right) =$$

Naloga

$$P_{KN}(\text{študent}|\textit{je ivan}) = \frac{max(0-1,0)}{2} + \frac{1*1}{2} * \left(\frac{max(0-1,0)}{4} + \frac{1*1}{4} * \left(\frac{0}{11} + 1 * \frac{1}{7} \right) \right) = 0 + \frac{1}{2} * \left(0 + \frac{1}{4} * \left(0 + \frac{1}{7} \right) \right) = \frac{1}{2} * \frac{1}{4} * \frac{1}{7} = \frac{1}{56}$$

Naloga

- Izračunajte verjetnost podanega stavka še s pomočjo metode "Stupid backoff“ (bigram in trigram)
- Izračunajte verjetnost podanega stavka še s pomočjo metode "Good-Turing“ (bigram)

Naloga

$$S(b_i|b_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{število}(b_{i-k+1}^{i})}{\text{število}(b_{i-k+1}^{i-1})} & \text{če število}(b_{i-k+1}^{i}) > 0 \\ 0{,}4S(b_i|b_{i-k+2}^{i-1}) & \text{drugače} \end{cases}$$

$$S(b_i) = \frac{\text{število}(b_i)}{N}$$

Naloga

Primer za bigram P(ivan|je):

$$S(\text{ivan}|\text{je}) = \frac{\text{število(je, ivan)}}{\text{število(je)}} = \frac{2}{2} = 1$$

Primer za trigram P(ivan|kriv je):

$$S(\text{ivan}|\text{kriv je}) = \frac{\text{število(kriv, je, ivan)}}{\text{število(kriv, je)}}$$

število(kriv, je, ivan) = 0 (ni večje od 0, zato uporabimo informacije o bigramih)

$$S(\text{ivan}|\text{kriv je}) = 0,4 \cdot S(\textit{ivan}|\textit{je}) = 0,4 \cdot \frac{\text{število(je, ivan)}}{\text{število(je)}} = 0,4$$

Naloga

- $P^*_{GT}(\text{stvari z ničelno frekvenco}) = \frac{N_1}{N}$
- $c^* = \frac{(c+1)N_{c+1}}{N_c}(\text{Stvari z ne ničelno frekvenco})$

$$
\begin{aligned}
c &= 2 \\
N_3 &= 0, \text{ po prilagoditveni funkciji dobimo} N_3 = 2 \\
c^* &= \frac{(c+1)N_{c+1}}{N_c} = \frac{3 \cdot 2}{4} = \frac{6}{4} = \frac{3}{2} \\
N &= 14 \\
P^*_{GT}(ivan|je) &= \frac{c^*}{N} = \frac{\frac{3}{2}}{14} = \frac{3}{28}
\end{aligned}
$$

Naloga

- Recimo, da želimo zgladiti verjetnosti model šumnega kanala za črkovanje:
- Imejmo dve besedi, x in y, kjer se x od y razlikuje v dveh črkah.
- Črka x_i v besedi x je zamenjana s črko y_i v besedi y.
- Želimo aplicirati glajenje add-1 na $P(x|y)$ oz. verjetnost, da sta zamenjani črki x_i in y_i.
- Metoda največjega verjetja: $P(x|y) = \frac{zamenjava[x_i,y_i]}{\text{število}(y_i)}$.
- $zamenjava[x_i, y_i]$ je število, kolikokrat je črka x_i v korpusu zamenjana s črko y_i in število(y_i) je število pojavitev črke y_i v korpusu.
- Kakšna je enačba za $P(x|y)$, če uporabimo glajenje add-1 v primeru zamenjave?
- Upoštevajte, da so v korpusu uporabljene samo slovenske besede in znaki, in da ima slovar V besed.

Naloga

- $\frac{zamenjava[x_i,y_i]}{\text{število}(x_i,y_i)}$
- $\frac{zamenjava[x_i,y_i]+1}{\text{število}(y_i)+V}$
- $\frac{zamenjava[x_i,y_i]+1}{\text{število}(y_i)+25}$
- $\frac{zamenjava[x_i,y_i]+1}{\text{število}(y_i)*V}$
- $\frac{zamenjava[x_i,y_i]+1}{\text{število}(y_i)+50}$

Naloga

- ☐ $\frac{zamenjava[x_i,y_i]}{število(x_i,y_i)}$ ✘
- ☐ $\frac{zamenjava[x_i,y_i]+1}{število(y_i)+V}$ ✘
- ☐ $\frac{zamenjava[x_i,y_i]+1}{število(y_i)+25}$ ✘
- ☐ $\frac{zamenjava[x_i,y_i]+1}{število(y_i)*V}$ ✘
- ☐ $\frac{zamenjava[x_i,y_i]+1}{število(y_i)+50}$ ✔

Naloga

- Recimo, da imamo naslednje verjetnosti besed v besedilih za ocenjevanje filmov:

	poz	neg
mi	0,09	0,16
so	0,07	0,06
všeč	0,29	0,06
komedije	0,08	0,11

- Imamo naslednji stavek: **všeč so mi komedije.**
- Z uporabo metode Naïve Bayes in upoštevanjem, da sta verjetnosti posameznega razreda enaki, določite v kateri razred sodi podana poved.

Naloga

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in \text{položaj}} P(x_i|c_j)$$

$$P(pos) = 0,5 \cdot 0,09 \cdot 0,07 \cdot 0,29 \cdot 0,08 = 7,308e-05$$

$$P(neg) = 0,5 \cdot 0,16 \cdot 0,06 \cdot 0,06 \cdot 0,11 = 3,168e-05$$

Poved je klasificirana kot pozitivna kritika.

Analiza sentimenta

(angl. Sentiment Analysis)

Analiza sentimenta

Različna imena

- Analiza sentimenta (angl. *Sentiment Analaysis*)
- Pridobivanje mnenja (angl. *Opinion extraction*)
- Rudarjenje mnenj (angl. *Opinion mining*)
- Rudarjenje sentimenta (angl. *Sentiment mining*)
- Analiza subjektivnosti (angl. *Subjectivity analysis*)

Analiza sentimenta

- Je kritika filma pozitivna ali negativna?
- Kakšno mnenje imajo študenti na FERI o Androidu
- Kakšno je zaupanje potrošnikov (narašča/pada)
- Kakšno je mnenje ljudi o politikih
- Napoved rezultatov volitev

Tipologija čustev (Scherer)

- Emocije (jeza, žalost, veselje, strah, sram, ponos, vznesenost)
- Razpoloženje (veselo, mračno, razdražljivo, brezvoljno, depresivno)
- Medosebna stališča (prijaznost, spogledljivost, oddaljenost, hladnost, toplost)
- Odnos (naklonjenost, ljubezen, sovraštvo, spoštovanje)
- Osebnostne lastnosti (živčnost, nestrpnost, nepremišljenost, čemerenost, sovražnost, ljubosumnost)

Analiza sentimenta

- Analiza sentimenta je odkrivanje odnosov
- Kakšno mnenje imajo študenti na FERI o Androidu
- Kakšno je zaupanje potrošnikov (narašča/pada)
- Kakšno je mnenje ljudi o politikih
- Napoved rezultatov volitev

Analiza sentimenta

Analiza sentimenta je odkrivanje **odnosov**.

- Nosilec (izvor)
- Cilj (aspekt)
  - cena tiskalnika, enostavnost uporabe tiskalnika itd.
- Tipi aspektov
  - Množica tipov (naklonjenost, ljubezen, sovraštvo, spoštovanje)
  - Obtežena polarnost (pozitivno, negativno, nevtralno)
- Besedilo, ki vsebuje opis odnosa
  - Stavek ali celoten dokument

Analiza sentimenta

- Enostavnejša naloga
  - Ali je odnos v besedilu pozitiven ali negativen
- Zahtevnejša naloga
  - Rangirati odnos v besedilu (npr. od 1 do 10)
- Naprednejša naloga
  - Odkrivanje vira, cilja, kompleksnih tipov odnosov

Osnovni algoritem analize sentimenta

- *BoPang and Lillian Lee. 2004.A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts.*
- Klasifikacija sentimenta s pomočjo filmskih kritik
- Ugotavljanje polarnosti
  - Ali je kritika filma (IMDB) pozitivna ali negativna
- Podatki: `http://www.cs.cornell.edu/people/pabo/movie-review-data`

Osnovni algoritem analize sentimenta

- Prilagodila sta ga Pang in Lee
- Leksikalna analiza
- Ugotavljanje značilk
- Klasifikacija z uporabo različnih klasifikatorjev
  - Naïve Bayes
  - MaxEnt (Max Entropy)
  - SVM (Support Vector Machine)

Leksikalna analiza

- Operacije nad datotekami HTML in XML
- Twitter (uporabniška imena, označevanja)
- Ohraniti velike črke
- Telefonske številke, datumi
- Čustveni simboli (angl. *Emotions*)

Ugotavljanje značilk za analizo sentimenta

- Kako ugotovimo negacije?
  - Ta film mi ni všeč
  - Ta filmi mi je zelo všeč
- Katere besede uporabiti?
  - Samo pridevnike
  - Vse besede (izkaže se, da uporaba vseh besed daje boljše rezultate)
- Metoda TF-IDF (angl. term frequency–inverse document frequency) in predstavitev s pomočjo vektorjev

Negacija

- Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).

  Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?

  Sentiment Classification using Machine Learning Techniques. EMNLP-2002,

  79-86.
- Dodajanje besede NOT_ vsaki besedi med negacijo in ločilom
  - didn't like this movie, but I
  - didn't NOT_like NOT_this NOT_movie but I

Naïve Bayes (ponovitev)

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in \text{položaj}} P(x_i|c_j)$$

$$\hat{P}(c_j) = \frac{\text{številoDokumentov}(C = c_j)}{\text{številoDokumentov}}$$

$$\hat{P}(b_i|c_j) = \frac{\text{število}(b_i, c_j)}{\sum_{b \in V} \text{število}(b, c_j)}$$

Binariziran Multinominal Naïve Bayes

Učenje

- Pojavitev besed je pomembnejša kot njihove frekvence
- Iz učnega korpusa se izlušči slovar (angl. *Vocabulary*)
- Izračunajo se $P(c_j)$
  - dok_j = vsi dokumenti razredac_j
  - $P(c_j) = \frac{|dok_j|}{|\text{število vseh dokumentov}|}$
- Izračunamo verjetnost $P(b_k|c_j)$
  - $besedilo_j$ = dokument, ki vsebuje vse dokumente dok_j
  - Za vsako besedo b_k iz slovarja
    - n_k = število pojavitev besede b_k v $besedilo_j$
    - $P(b_k|c_j) = \frac{n_k+\alpha}{n+\alpha|slovar|}$
    - n - število leksikalnih simbolov v razredu c_j

Binariziran Multinominal Naïve Bayes

Učenje

- Pojavitev besed je pomembnejša kot njihove frekvence
- Iz učnega korpusa se izlušči slovar (angl. *Vocabulary*)
- Izračunajo se $P(c_j)$
  - dok_j = vsi dokumenti razredac_j
  - $P(c_j) = \frac{|dok_j|}{|\text{število vseh dokumentov}|}$
- Izračunamo verjetnost $P(b_k|c_j)$
  - Odstranimo vse podvojene besede znotraj *dok*
    - Za vsako besedo b v dok_j ohranimo eno instanco besede b
  - $besedilo_j$ = dokument, ki vsebuje vse dokumente dok_j
  - Za vsako besedo b_k iz slovarja
    - n_k = število pojavitev besede b_k v $besedilo_j$
    - $P(b_k|c_j) = \frac{n_k+\alpha}{n+\alpha|slovar|}$
    - n - število leksikalnih simbolov v razredu c_j

Binariziran Multinominal Naïve Bayes

Klasifikacija

- Odstranimo vse podvojene besede v testnem dokumentu d
- Uporabimo enačbo za Multinominal Naïve Bayes

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in \text{položaj}} P(x_i|c_j)$$

Klasifikatorji

- Klasifikatorja MaxEnt in SVM ponavadi dajeta boljše rezultate kot Naïve Bayes
- Problemi
  - Včasih je težko iz besedila izluščiti značilke
    Če berete ta oglas samo zato ker je avto znamke Ferrari, si morate sliko avtomobila zalepiti na steno.
  - Včasih značilke kažejo na nasprotno klasifikacijo besedila
    Ta film bi lahko bil dober, nastopajo odlični igralci in tudi zgodba je odlična.

Leksikon sentimenta

- Kateri pomen sentimenta imajo besede
- Leksikoni
  - The General Inquirer (`http://www.wjh.harvard.edu/~inquirer`)
    Prosto dostopen za raziskovanje
  - Linguistic Inquiry and Word Count (`http://www.liwc.net/`)
    Cena $30 oz. $90
  - MPQA Subjectivity Cues Lexicon
    (`http://www.cs.pitt.edu/mpqa/subj_lexicon.html`)
    GNU GPL
  - Bing Liu Opinion Lexicon
    (`http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar`)
  - SentiWordNet
    (`http://sentiwordnet.isti.cnr.it/`)
    Vsem WordNet sinsetom je dodeljena stopnja pozitivnosti, negativnosti in nevtralnosti

Leksikoni sentimenta

- Želimo zgraditi lasten leksikon sentimenta
- Delno nadzorovano učenje
  - Imamo na voljo malo količino informacij (nekaj označenih primerov ali ročno zgrajenih vzorcev)

Intuicija za identifikacijo besedne polarizacije

- Intuicija (Hatzivassiloglou in McKeown)
  - Pridevniška povezanost z besedo "and" ima enako polarizacijo (corrupt and brutal)
  - Pridevniška povezanost z besedo "but" nima enako polarizacijo (fair but brutal)
- 1. korak: ročno označimo "semensko" množico pridevnikov
- 2. korak: razširimo semensko množico s povezanimi pridevniki
  - kako pogosto se pojavljajo v povezavi z besedo "and" in besedo "but"
- 3. korak: nadzorovano klasificiranje glede na "podobnost polarnosti"
- 4. korak: združevanje besed v dve množici (pozitivno/negativno)

Algoritem ki temelji na polariteti fraz

- *Turney (2002): Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews*
- Glede na kritike, pridobivanje fraznega leksikona
- Učenje polaritete fraz
- Rangiranje kritik, glede na povprečnost polaritete fraz

Algoritem Turney

- Pridobivanje dvo-besednih fraz skupaj s pridevniki
  - Uporabimo samodejno oblikoslovno označevanje (angl. *part of speech tagging*)
- Kako merimo polariteto fraz v katerih se pojavi beseda "odlično"?
- Kako merimo polariteto fraz v katerih se pojavi beseda "slabo"?

Pointwise Mutual Information

- Medsebojne informacije (angl. *Mutual information*) med dvema naključnima spremenljivkama X in Y

$$I(X,Y) = \sum_x \sum_y P(x,y) \cdot log_2 \frac{P(x,y)}{P(x) \cdot P(y)}$$

- Točkovna medsebojna informacija (angl. *Pointwise mutual information*)
  - Kako pogosto se pojavita dogodka x in y skupaj v primerjavi z njihovo neodvisno pojavitvijo

$$PMI(X,Y) = log_2 \frac{P(x,y)}{P(x) \cdot P(y)}$$

Pointwise Mutual Information

- Točkovna medsebojna informacija (angl. *Pointwise mutual information*)
  - Kako pogosto se pojavita dogodka x in y skupaj v primerjavi z njihovo neodvisno pojavitvijo

$$PMI(X, Y) = log_2 \frac{P(x, y)}{P(x) \cdot P(y)}$$

- PMI med dvema besedama
  - Kako pogosto se pojavita besedi x in y skupaj v primerjavi z njihovo neodvisno pojavitvijo

$$PMI(beseda_1, beseda_2) = log_2 \frac{P(beseda_1, beseda_2)}{P(beseda_1) \cdot P(beseda_2)}$$

Kako ocenimo PMI

- Lahko uporabimo spletni iskalnik
  - P(*beseda*) ocenimo s številom zadetkov $\frac{zadetki(beseda)}{N}$
  - P($beseda_1$,$beseda_2$) ocenimo s številom zadetkov $\frac{zadetki(beseda_1 \ \ BLIZU \ \ beseda_2)}{N^2}$
  - Primer (Google): apple AROUND(4) iphone
  - N - $zadetki(beseda_1) + zadetki(beseda_2)$

$$PMI(beseda_1, beseda_2) = log_2 \frac{\frac{1}{N^2} \cdot zadetki(beseda_1 \ \ BLIZU \ \ beseda_2)}{\frac{1}{N} \cdot zadetki(beseda_1) \cdot \frac{1}{N} \cdot zadetki(beseda_2)}$$

Fraza v relaciji z besedama *dobro* in *slabo*

Ali se fraza pojavlja pogosteje z besedo *dobro* ali z besedo *slabo*?

$$Polariteta(fraza) = PMI(fraza, dobro) - PMI(fraza, slabo) =$$

$$= log_2 \frac{zadetki(fraza\ \ BLIZU\ \ dobro)}{zadetki(fraza) \cdot zadetki(dobro)} - log_2 \frac{zadetki(fraza\ \ BLIZU\ \ slabo)}{zadetki(fraza) \cdot zadetki(slabo)}$$

$$= log_2 \frac{zadetki(fraza\ \ BLIZU\ \ dobro)}{zadetki(fraza) \cdot zadetki(dobro)} \frac{zadetki(fraza) \cdot zadetki(slabo)}{zadetki(fraza\ \ BLIZU\ \ slabo)}$$

$$= log_2 \frac{zadetki(fraza\ \ BLIZU\ \ dobro) \cdot zadetki(slabo)}{zadetki(dobro) \cdot zadetki(fraza\ \ BLIZU\ \ slabo)}$$

Algoritem Turney

- Uspešnejši od osnovnega algoritma
- Uporablja fraze namesto besed
- Uči se na domensko specifičnih informacijah
- Na podoben način lahko učimo tudi WordNet

Analiza sentimenta

- Naprednejše metode (ugotavljanje aspektov, atributov in cilja sentimenta)
- V osnovi je modelirana kot klasifikacija ali regresija
- Značilnosti:
  - Negacija je pomembna
  - Uporaba vseh besed s pomočjo Naïve Bayes daje boljše rezultate
  - Iskanje podmnožic besed pomaga v določenih nalogah
    - Ročno zgrajeni leksikoni polarnosti
    - Uporaba semen in delno-nadzorovano učenje za izgradnjo leksikona

Viri

- Dan Jurafsky, Chris Manning, Natural Language Processing http://web.stanford.edu/∼jurafsky/NLPCourseraSlides.html
- Park K., Hong J., and Kim W., A Methodology Combining Cosine Similarity with Classifier for Text Classification. Applied Artificial Intelligence, 2020. 34(5): 396-411.

Diskriminatorni klasifikatorji

(angl. Discriminative classifiers)

Uvod

- Do sedaj smo obravnavali “generativne modele”
  - Jezikovni modeli, Naïve Bayes
- Bolj uporabni so diskriminatorni (pogojni) modeli
  - So bolj točni
  - Omogočajo enostavno vključitev jezikovno pomembnih značilk
  - Omogočajo samodejno izgradnjo jezikovno neodvisnih modelov

Generativni in diskriminatorni modeli

- Klasifikatorji se v fazi sklepanja odločajo glede na verjetnost $P(c|d)$
- Generativni modeli
  - Uporaba pogojne gostote $P(d|c)$ skupaj s priorno (predhodno) verjetnostjo $P(c)$
  - Uporaba Bayesovega pravila za izračun posteriorne verjetnosti $P(c|d) = \frac{P(d|c)P(c)}{P(d)}$
  - n-gram modeli, klasifikator Naïve Bayes, skriti Markovi modeli,verjetnostne kontekstno proste gramatike, modeli v strojnem prevajanju
- Diskriminatorni modeli
  - Neposreden izračun verjetnosti $P(c|d)$
  - Maksimiramo pogojno verjetnost
  - Težje narediti
  - Izboljša zmogljivost klasifikatorja - Naïve Bayes $73,6 \Rightarrow 76,1$ (2,5%)
  - Logistična regresija, Perceptron, SVM, ...

Značilke besedila za diskriminatorni model

- Značilka besedila f predstavlja elementarne vidike povezanosti videnega besedila d z razredom c.
- Značilko predstavimo s pomočjo funkcije, ki vrne realno vrednost na določenem intervalu: $f : CxD \rightarrow \mathbb{R}$.
- Primeri:
  - $f_1(c, d) \equiv [c = \text{MESTO} \wedge b_{i-1} = \text{"v"} \wedge \ velikaZačetnica(b_i)]$
  - $f_2(c, d) \equiv [c = \text{DRŽAVA} \wedge b_{i-2} = \text{"glavno"} \wedge b_{i-1} = \text{"mesto"} \wedge velikaZačetnica(b_i) \wedge b_{i+1} = \text{"je"}]$
  - Glavno mesto Slovenije je Ljubljana.
  - Potujem v Pariz.
- Model vsaki značilki dodeli utež, ki je lahko pozitivna (značilka je pravilna) ali negativna (značilka ni pravilna).

Značilke besedila za diskriminatorni model

- Empirično štetje (pričakovanje) značilk:
  empirični $E(f_i) = \sum_{(c,d) \in observed(C,D)} f_i(c,d)$
- Model za pričakovanje značilk
  $E(f_i) = \sum_{(c,d) \in observed(C,D)} P(c,d) f_i(c,d)$
- Splošna predstavitev značilk
  $f_i(c,d) \equiv [\Phi(d) \wedge c = c_j] \rightarrow [vrednost\ 0\ ali\ 1]$
  - Funkcija $\Phi(d)$ vrne logično vrednost
  - Kateremu razredu pripadajo podatki $c = c_j$

Linearni klasifikator, ki temelji na značilkah

Linearni klasifikator v fazi sklepanja

- Linearna funkcija za množico značilk (f_i) in razred c
- Vsaki značilki f_i dodaj utež λ_i
- Obravnavamo vse razrede za podatek d
- Za par (c, d), značilke glasujejo z uporabo uteži: $glasovanje(c) = \sum \lambda_i f_i(c, d)$
- Izbere se razred z največjo vrednostjo glasovanja: $c = \arg\max_{c \in C} \sum \lambda_i f_i(c, d)$

Linearni klasifikator, ki temelji na značilkah

- Kako nastaviti uteži?
- Perceptorn: poišče trenutno napačno obravnavane primere in spremeni uteži v smeri pravilne klasifikacije
- Metode na osnovi oddaljenosti (Support Vector Machines)

Linearni klasifikator, ki temelji na značilkah

- Eksponentni (loglinearni,maxent,logistični,Gibbs) model
- Funkcija *exp* spremeni rezultat v pozitivno vrednost (uteži imajo lahko negativne vrednosti)
- Verjetnostni model normaliziramo glede na linearno kombinacijo $\sum \lambda_i f_i(c, d)$:

$$P(c|d, \lambda) = \frac{exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} exp \sum_i \lambda_i f_i(c', d)}$$

- Uteži λ_i izberemo tako, da maksimiramo pogojno verjetnost podatkov na podanem modelu

Primer

- $P(\text{MESTO}|\text{v Maribor}) = \frac{e^{0,8}}{e^{0,8}+e^{0,6}} = 0,55$
- $P(\text{DRŽAVA}|\text{v Maribor}) = \frac{e^{0,6}}{e^{0,8}+e^{0,6}} = 0,45$
- $f_1(c,d) \equiv [c = \text{MESTO} \wedge b_{i-1} = \text{"v"} \wedge \ velikaZačetnica(b_i)] => \sum \lambda_1 f_1(c,d) = 0,8$
- $f_2(c,d) \equiv [c = \text{DRŽAVA} \wedge b_{i-2} = \text{"glavno"} \wedge b_{i-1} = \text{"mesto"} \wedge velikaZačetnica(b_i) \wedge b_{i+1} = \text{"je"}] => \sum \lambda_2 f_2(c,d) = 0,6$

Izgradnja modela Maxent

- Definiramo značilke
  - Besede
  - Beseda, ki vsebuje števila
  - Besede z določenimi končnicami (npr. “ing”, “s”)
- Značilke označimo z unikatnimi oznakami
  - Vsak podatek je lahko povezan z več značilkami $\Phi(d)$
  - Vsaka značilka vrne realno vrednost $f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$
- Značilke se dodajajo v času razvoja modela
  - Model preizkusimo na razvojni množici
  - Poskusimo dodati oz. popraviti obstoječe značilke
  - Ta postopek iterativno ponavljamo

Eksponentni verjetnostni model

Za podan model izberemo take vrednosti parametrov, da maksimirajo pogojno verjetnost modela.

$$\log P(C|D,\lambda) = \sum_{(c,d)\in(C,D)} \log P(c|d,\lambda) =$$

$$\sum_{(c,d)\in(C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)} =$$

$$\sum_{(c,d)\in(C,D)} \log \exp \sum_i \lambda_i f_i(c,d) - \sum_{(c,d)\in(C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c',d) =$$

$$N(\lambda) - M(\lambda)$$

Odvod števca

$$\frac{\partial N(\lambda)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \log \exp \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i} =$$

$$\frac{\partial \sum_{(c,d)\in(C,D)} \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i} =$$

$$\sum_{(c,d)\in(C,D)} \frac{\partial \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i} = \sum_{(c,d)\in(C,D)} f_i(c,d) =$$

empirični *števec*(f_i, C)

Odvod imenovalca

$log'(x) = \frac{1}{x}$

$$\frac{\partial M(\lambda)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}{\partial \lambda_i}$$

$(exp(x) * f(x))' = exp(x) * f(x)'$

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \frac{\partial \sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c', d)}{1} \frac{\partial \sum_i \lambda_i f_i(c', d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c', d)}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \frac{\partial \sum_i \lambda_i f_i(c', d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \sum_{c'} P(c'|d, \lambda) f_i(c', d) = \textbf{napovedan } \check{s}tevec(f_i, \lambda)$$

Odvod

- $$\frac{\partial \log P(C|D,\lambda)}{\partial \lambda_i} =$$

  empirični *števec*(f_i, C) − **napovedan** *števec*(f_i, λ)

- Optimalni parametri so tisti, pri katerih za vsako značilko velja, da je **napovedano pričakovanje** enako **empiričnemu pričakovanju**.
- Te modele imenujemo tudi modeli maksimalne entropije (angl. maximum entropy model). Razlog temu je, da moramo najti model z maksimalno entropijo oz. zadostiti naslednje omejitve:

$$E_p(f_j) = E_{\tilde{p}}(f_j), \forall j$$

Optimalni parametri

- Želimo izbrati parametre $\lambda_1, \lambda_2, \lambda_3, ...$, ki maksimirajo pogojno logaritemsko verjetnost učne množice

$$CLogVer(D) = \sum_{1}^{n} \log P(c_i|d_i)$$

- Da bi to lahko naredili moramo znati izračunati funkcijsko vrednost in parcialne odvode (gradient)
- Poiščemo optimalne parameter s pomočjo določene optimizacijske metode

Viri

- Dan Jurafsky, Chris Manning, Natural Language Processing `http://web.stanford.edu/~jurafsky/NLPCourseraSlides.html`
- Asist. dr. Igor Locatelli, mag. farm., Logistična regresija, 2014
- Prof. Jurij F. Tasič, Osnove Linearnih klasifikacijskih modelov, 2013

Strojno prevajanje

Vsebina

- Strojno prevajanje
- Leksikalna dvoumnost
- Različni vrstni red besed
- Skladenjske razlike
- Ločljivost samostalnikov
- Klasični pristopi za prevajanje s pravili
- Statistično strojno prevajanje (angl. *Statistical machine translation*)

Strojno prevajanje

- *Machine Translation* (MT)
- Prevajanje je zahtevno in ustvarjalno dejanje.
- Postopek, pri katerem računalniški program analizira besedilo in brez posredovanja človeka ustvari ciljno besedilo.
- Sistemi za strojno prevajanje vključujejo:
  - eno ali večjezične leksikone,
  - programe za morfološko analizo in sintezo,
  - programe za sintaktično analizo in sintezo,
  - programe za razreševanje večpomenskosti,
  - programe za prepoznavanje večbesednih semantičnih enot,
  - itd.

Strojno prevajanje

- Strojno prevajanje lahko v nekaterih primerih prevajalcu olajša delo ali pa ga celo popolnoma nadomesti:
  - grob prevod, ki ga kasneje pregleda in popravi prevajalec,
  - osnutek, ki služi kot pomoč pri prevajanju,
  - določene besedilne vrste, pri katerih je izrazje močno omejeno (vremenska napoved, navodila za uporabo, računalniški programi, zdravniška poročila, itd).
- Strojni prevajalniki so sposobni prepoznati kontekst, frazeme in idiome v izvirnem jeziku ter ustvariti koheziven in razumljiv prevod.
- Strojni prevodi uradnih listin in pravnih aktov so tako razumljivejši in pravilnejši kot pri govorjenem jeziku.

Različen vrstni red besed

- angleški vrstni red: subject - verb - object
- Primer:
  book the flight -> rezerviraj
  read the book -> knjiga

Leksikalna dvoumnost

- V slovenščini ne obstaja točno določeno zaporedje stavčnih členov, velja pa t.i. členitev po aktualnosti: najprej v stavku nastopajo stavčni členi, ki nosijo že znano informacijo, na koncu pa tisti, ki povedo prejemniku nek nov podatek.
- slovenski vrstni red: besedni vrstni red se prilagaja poudarjeni besedi
- Primer:
  - Vrstni red besed v stavku seveda obstaja.
  - Vrstni red besed seveda v stavku obstaja.
  - Seveda vrstni red besed v stavku obstaja.
  - V stavku seveda obstaja vrstni red besed.
  - Seveda obstaja v stavku vrstni red besed.
  - Seveda obstaja vrstni red besed v stavku.

Različen vrstni red besed

- Nobena od teh možnosti ni napačna, kako sodijo v kontekst, je pa drugo.
- Še najmanj zaznamovana je prva poved.
- V drugi povedi poudarjamo samoumevnost obstanka vrstnega reda besed (z besedo seveda).
- V tretji želimo poudariti, da obstaja.
- V četrti poudarjamo vrstni red besed.
- V peti in šesti pa poudarek velja tako na vrstnem redu besed, kot temu, da obstaja.

Skladenjske razlike

- Tisto, kar hočemo poudariti, damo ponavadi na konec, včasih pa na začetek.
- Primeri:
  - Spoznali smo tri fante med vojaki, ki so pravkar imeli počitek. Kdo je imel počitek? Trije fantje ali vojaki?
  - Lepa dekleta ljubijo barabe. Kdo koga ljubi? Lepa dekleta barabe ali barabe lepa dekleta?
- Primeri pri prevajanju (Google translate in popravljena imena):
  - Petra ljubi Marka -> Petra loves Marko.
  - Marka ljubi Petra -> Marko loves Petra.

Ločljivost samostalnikov

The computer outputs the data, **it** is fast. Računalnik izpiše podatke, **je** hiter.

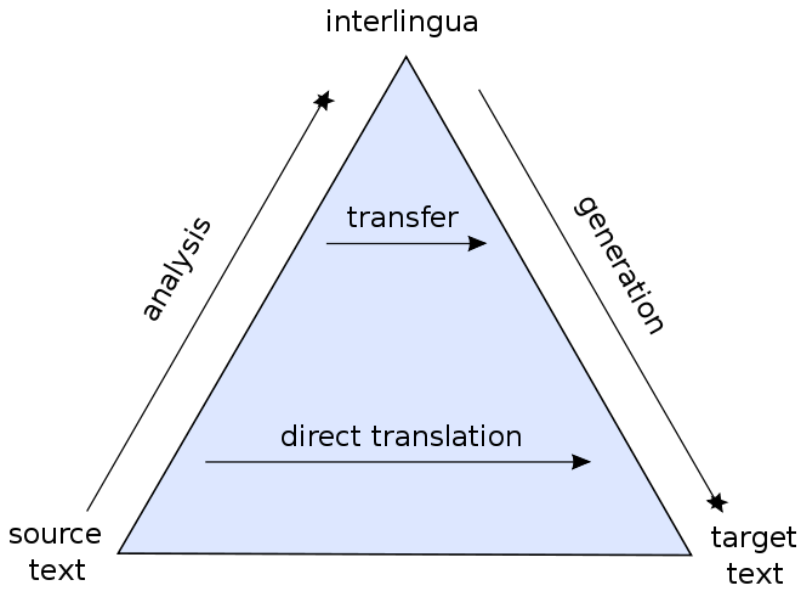
- **it** -> the computer ali the data?
- Vidi se, da je mišljeno za računalnik, zato se prevede kot "je".

The computer outputs the data, **it** is stored in ASCII. Računalnik izpiše podatke, **so** shranjeni v ASCII.

- **it** -> the computer ali the data?
- Vidi se, da je mišljeno za podatke, zato se prevede kot ”so“.

Klasični pristopi za prevajanje s pravili

- Neposredno strojno prevajanje (angl. *Dictionary-Based method*)
- Transferno strojno prevajanje (angl. *Transfer-Based method*)
- Medjezikovno strojno prevajanje (angl. *Interlingua-Based method*)



Neposredno strojno prevajanje

- Prevaja se besedo po besedo.
- Zelo malo analize izvornega besedila.
  - Brez skladenjske ali semantične analize.
- Zanaša se na dvojezični slovar.
  - Za vsako besedo v izvornem jeziku, slovar določa množico pravil za prevod te besede.
- Primer:
  - how much -> if (prejšnjaBeseda == "how") return "koliko";
  - as much -> if (prejšnjaBeseda == "as") return "toliko";

Neposredno strojno prevajanje

Slabosti:

- Pomanjkanje analize izvornega besedila.
  - Težko oz. nemogoče je zajeti dolge prerazporeditve, ker nimamo nobenega skladenjskega znanja.
- Primer:
  - angleščina: Sources said that IBM bought Lotus yesterday.
  - japonščina: Sources yesterday IBM Lotus bought that said.
- Besede so prevedene brez znanja o ločevanju.
- Primer:
  - They said that I like ice-cream.
  - They like that ice-cream.

Transferno strojno prevajanje

3 faze prevajanja:

- Analiza
  - Analiziranje izvornega stavka, npr. sintaktično (skladenjsko) drevo
- Prenos
  - Preoblikujemo razčlenitveno drevo izvornega stavka v razčlenitveno drevo ciljnega stavka tako, da uporabimo množico pravil, ampak ta pravila so zgrajena iz razčlenitvenega drevesa izvornega stavka.
- Ustvarjanje
  - S pomočjo razčlenitvenega drevesa ciljnega stavka ustvarimo ciljni stavek.

Medjezikovno strojno prevajanje

2 fazi prevajanja:

- Analiza
  - Analiziramo izvorni stavek v predstavitev njegovega pomena, pri čemer upamo na to, da je neodvisna od jezika (angl. *language-independed representation*).
- Ustvarjanje
  - Preoblikujemo predstavitev pomena v izhodni stavek.
- Če hočemo zgraditi sistem za prevajanje, ki prevaja med N jeziki, moramo razviti N sistemov za analizo in ustvarjanje.

Koncepti pri medjezikovnem prevajanju

- Nemščina ima dve besedi za steno -> notranja in zunanja stena
- Japonščina ima dve besedi za brata -> starejši in mlajši brat
- Španščina ima dve besedi za nogo -> človeška in živalska noga
- Vsak jezik ima svoje koncepte, zato je potrebno biti previden pri grajenju sistema za prevajanje -> zaradi tega ta pristop ni enostaven

Statistično strojno prevajanje

- *Statistical Machine Translation* (SMT)
- Je vrsta strojnega prevajanja, ki temelji na večji količini vzporednih besedil, iz katerih se s statističnimi algoritmi izračunavajo verjetnosti za posamezne jezikovne enote.
- Osnovna ideja je uporaba vzporednega korpusa za učenje sistema za prevajanje.

"..one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When we look at an article in Russian, we could says that it is written in English, but it has been coded in some strange symbols. We will now proceed to decode."(Warren Waver, 1949)

Dekodirnik (ali iskalni algoritem)

- Ideja: uporaba šumnega kanala za prevajanje
- Cilj: prevajalni sistem iz slovenščine v angleščino
- Ima 2 modela:
  - Jezikovni model $P(e)$
  - Prevajalni model $P(s|e)$
- Pravilo Bayes:

$$P(e|s) = \frac{P(e,s)}{P(s)} = \frac{P(e) \cdot P(s|e)}{P(s)}$$

- Iskalni algoritem:

$$\textit{dekodirnik} = \arg\max_e P(e|s) = \arg\max_e P(e) \cdot P(s|e)$$

Prevajalni model

- "backwards"model -> če prevajamo iz angleščine v slovenščino želimo izračunati $P(s|e)$, v bistvu pa računamo $P(e|s)$ oz. verjetnost, da je slovensko besedilo prevod angleškega besedila.
- Učimo ga na vzporednem korpusu.
- Jezikovni model lahko dopolni pomanjkljivosti prevajalnega modela.

Primer

Prevajamo naslednjo poved iz slovenščine v angleščino z uporabo prevajalnega modela:

Lačen sem. $P(slo|ang)$
$P(\text{Lačen sem}|\text{What hunger have}) = 0,000014$
$P(\text{Lačen sem}|\text{Hungry I am}) = 0,000001$
$P(\text{Lačen sem}|\text{I am hungry}) = 0,0000015$
$P(\text{Lačen sem}|\mathbf{Have\ I\ hunger}) = \mathbf{0,00002}$

Primer

Prevajamo naslednjo poved iz slovenščine v angleščino z uporabo jezikovnega in prevajalnega modela:

Lačen sem.
$P(ang) \cdot P(slo|ang)$
$P(\text{Lačen sem}|\text{What hunger have}) = 0,000001 \cdot 0,000014 = 1,4e^{-11}$
$P(\text{Lačen sem}|\text{Hungry I am}) = 0,0000014 \cdot 0,000001 = 1,4e^{-12}$
$P(\text{Lačen sem}|\textbf{I am hungry}) = 0,0001 \cdot 0,0000015 = \mathbf{1,5e^{-10}}$
$P(\text{Lačen sem}|\text{Have I hunger}) = 0,00000098 \cdot 0,00002 = 1,96e^{-11}$

Prevajalni model IBM

- Modela IBM 1 in 2 (vse skupaj jih je že 5)
  - Prva generacija statističnega strojnega prevajanja.
- Modeli, ki temeljijo na frazah
  - Druga generacija statističnega strojnega prevajanja,
  - boljši kot modeli IBM,
  - te modele je uporabljal prevajalnik Google.

Model IBM 1

- Se več ne uporablja za prevajanje, ampak za poravnavo.
- Poravnava a nam pove iz katere angleške besede so nastale slovenske besede.
  e = the dog barks
  s = pes laja
  $l = 3,\ m = 2$
- Angleški stavek ima l besed $(e_1, ..., e_l)$, medtem ko ima slovenski stavek m besed $(s_1, ..., s_m)$.
- Formalni zapis: $a \in \{a_1, a_2, ..., a_m\}$, kjer je $a_j \in \{0, 1, ..., l\}$
  Za vsako slovensko besedo določimo angleško besedo
- Število vseh možnih poravnav je $(l + 1)^m$.

Primer 1

$\mathrm{e} = \mathrm{the}_1\ \mathrm{dog}_2\ \mathrm{barks}_3$

$\mathrm{s} = \mathrm{pes}_1\ \mathrm{laja}_2$

$a \in \{a_1, a_2\}$, kjer je $a_1 = 2, a_2 = 3$.

$\mathrm{s} = \mathrm{pes}_1\ \mathrm{laja}_2$

$\mathrm{e} = \mathrm{the}_1\ \mathrm{dog}_2\ \mathrm{barks}_3$

$a \in \{a_1, a_2, a_3\}$, kjer je $a_1 = 0, a_2 = 1, a_3 = 2$.

Primer 2

$e = \text{and}_1\ \text{the}_2\ \text{program}_3\ \text{has}_4\ \text{been}_5\ \text{implemented}_6$

$s = \text{in}_1\ \text{program}_2\ \text{je}_3\ \text{bil}_4\ \text{implementiran}_5$

$l = 6,\ \ m = 5$

- Pravilna poravnava je $\{1, 3, 4, 5, 6\}$
  - Prva beseda v slovenščini je poravnana s prvo besedo v angleščini, druga beseda s tretjo besedo, itd.
- Napačna poravnava je $\{1, 1, 1, 1, 1\}$
  - Vsaka beseda v slovenščini je poravnana s prvo besedo v angleščini.

Verjetnost poravnave

- Verjetnost za poravnavo je $P(s, a|e, m)$, kjer je s slovenski stavek, a je poravnava, e je angleški stavek in m je število besed v slovenščini
- Razgradimo na dva verjetnostna modela:
  - $P(a|e, m)$: porazdelitev vseh možnih poravnav; $(l + 1)^m$ vrednosti za a
  - $P(s|a, e, m)$: pogojuje na poravnavo, angleški stavek in število besed v slovenščini
  - A je množica vseh poravnav

$$P(s, a|e, m) = P(a|e, m) \cdot P(s|a, e, m)$$

$$P(s|e, m) = \sum_{a \in A} P(a|e, m) \cdot P(s|a, e, m)$$

Verjetnost poravnave

- Ko imamo model $P(s, a|e, m)$, izračunamo za vsako poravnavo verjetnost:

$$P(a|s, e, m) = \frac{P(s, a|e, m)}{\sum_{a \in A} P(s, a|e, m)}$$

- Za podan par s, e lahko izračunamo najverjetnejšo poravnavo:

$$a^* = \arg\max_a P(a|s, e, m)$$

Modeli

$P(a|e,m) = \frac{1}{(l+1)^m}$

- vsaka poravnava je enako verjetna: slovenska beseda se lahko poravna z vsako besedo iz angleške besede
- t - ocenjena verjetnost poravnave besed (angl. translation parameter)

$P(s|a,e,m) = \prod_{j=1}^{m} t(s_j|e_{a_j})$

$e = the_1\, dog_2\, barks_3$

$s = pes_{s_1}\, laja_{s_2}$

$P(\text{ pes laja } | \{2,3\}, \text{ the dog barks }, 2) = t(pes|dog) \cdot t(laja|barks)$

Primer 1

e = the dog barks
s = pes laja

- Koliko je verjetnost $P(s|a, e, m)$ za zgornji primer?

$t(pes|the) = 0,3 \quad t(laja|the) = 0,4$
$t(pes|dog) = 0,8 \quad t(laja|dog) = 0,3$
$t(pes|barks) = 0,1 \quad t(laja|barks) = 0,7$

$$P(s|a, e, m) = t(pes|dog) \cdot t(laja|barks) = 0,8 \cdot 0,7 = 0,56$$

$$P(s, a|e, m) = P(a|e, m) \cdot P(s|a, e, m) = \frac{1}{(l+1)^m} \cdot \prod_{j=1}^{m} t(s_j|e_{a_j})$$

Primer 2

e = and the program has been implemented
s = in program je bil implementiran

$l = 6,\ m = 5$
$a = \{1, 3, 4, 5, 6\}$

Primer 2

$t(in|and) = 0,8 \quad t(in|the) = 0,1 \quad t(in|program) = 0,2$
$t(in|has) = 0,4 \quad t(in|been) = 0,3 \quad t(in|implemented) = 0,2$
$t(program|and) = 0,3 \quad t(program|the) = 0,1$
$t(program|program) = 0,7 \quad t(program|has) = 0,5$
$t(program|been) = 0,3 \quad t(program|implemented) = 0,4$
$t(je|and) = 0,5 \quad t(je|the) = 0,2 \quad t(je|program) = 0,3$
$t(je|has) = 0,9 \quad t(je|been) = 0,1 \quad t(je|implemented) = 0,4$
$t(bil|and) = 0,2 \quad t(bil|the) = 0,1 \quad t(bil|program) = 0,3$
$t(bil|has) = 0,6 \quad t(bil|been) = 0,8 \quad t(bil|implemented) = 0,4$
$t(implementiran|and) = 0,5 \quad t(implementiran|the) = 0,2$
$t(implementiran|program) = 0,3 \quad t(implementiran|has) = 0,4$
$t(implementiran|been) = 0,1 \quad t(implementiran|implemented) = 0,9$

Primer 2

$t(in|and) = 0,8$
$t(program|program) = 0,7$
$t(je|has) = 0,9$
$t(bil|been) = 0,8$
$t(implementiran|implemented) = 0,9$

Rešitev:

$P(s|a,e,m) = t(in|and) \cdot t(program|program) \cdot t(je|has) \cdot t(bil|been) \cdot t(implementiran|implemented) = 0,36288$

$$P(s,a|e,m) = P(a|e,m) \cdot P(s|a,e,m) = \frac{1}{(l+1)^m} \cdot \prod_{j=1}^{m} t(s_j|e_{a_j})$$

Model IBM 2

- Vpeljuje parametre poravnave in popačenja
  $q(i|j, l, m)$ = verjetnost, da je j-ta slovenska beseda povezana z i-to angleško besedo. Pri tem sta dollžini angleškega in slovenskaga stavka l in m.
- Definiramo:
  $p(a|e, m) = \prod_{j=1}^{m} q(a_j|j, l, m)$; kjer je $a = \{a_1, ..., a_m\}$
- Dobimo:
  $p(s, a|e, m) = \prod_{i=1}^{m} q(a_j|j, l, m)t(s_j, e_{a_j})$

Primer

e = And the program has been implemented
s = In program je bil implementiran
a = $\{1, 3, 4, 5, 6\}$

$p(a|e, 5) =$
$q(1|1, 6, 5) \cdot q(3|2, 6, 5) \cdot q(4|3, 6, 5) \cdot q(5|4, 6, 5) \cdot q(6|5, 6, 5)$

$p(s|a, e, 5) = t(In|And) \cdot t(program|program) \cdot t(je|has) \cdot$
$t(bil|been) \cdot t(implementiran|implemented)$

Končni model

- 1. korak:
  $p(a|e,m) = \prod_{j=1}^{m} q(a|j,l,m)$
- 2. korak: Uporabimo verjetnosti besed
  $p(s|a,e,m) = \prod_{j=1}^{m} t(s_j|e_{a_j})$
- Končni model:
  $p(s,a|e,m) = p(a|e,m) \cdot p(s|a,e,m) =$
  $\prod_{j=1}^{m} q(a_j|j,l,m)t(s_j|e_{a_j})$

Poravnava

- Ko imamo parametre q in t, lahko zelo enostavno določimo najbolj verjetno poravnavo.
- Za par: $e_1, e_2, ..., e_m$ in $s_1, s_2, ..., s_l$ lahko izračunamo:
  $a_j = arg\ max_{a \in \{0...l\}} q(a|j, l, m) * t(s_j|e_{a_j});\ j = 1...m$

Primer

e = NULL And the program has been implemented
s = In program je bil implementiran

Izračun poravnave za besedo “je”:
$NULL : q(0|3,6,5) \cdot t(je|NULL)$
$And : q(1|3,6,5) \cdot t(je|And)$
$the : q(2|3,6,5) \cdot t(je|the)$
...
$implemented : q(6|3,6,5) \cdot t(je|implemented)$

- q - verjetnost pozicije
- t - verjetnost besede
- vrednosti q in t dobimo na osnovi učnega korpusa

Model

- Učna množica ($\{e^{(1)}, s^{(1)}, a^{(1)}\}$, $\{e^{(2)}, s^{(2)}, a^{(2)}\}$, ...):
  $e^{(100)}$ = And the program has been implemented
  $s^{(100)}$ = In program je bil implementiran
  $a^{(100)} = \{1, 3, 4, 5, 6\}$
- $t_{MLE}(s|e) = \frac{\text{število}(e,s)}{\text{število}(e)}$
- $q_{MLE}(j, i|l, m) = \frac{\text{število}(j|i,m,n)}{\text{število}(i,l,m)}$

Viri

- Michael Collins, Natural Language Processing
  https://class.coursera.org/nlangp-001
  https://archive.org/details/academictorrents_f99e7184fca947ee8f77901679e171fcadbf82e7
- Transfer-based MT
  https://en.wikipedia.org/wiki/Transfer-based_machine_translation

Ekstrakcija informacij in prepoznavanje imenskih entitet

(angl. Information Extraction and Named Entity Recognition)

Ekstrakcija informacij

- Začela se je razvijati ob pojavitvi sistemov za razpoznavanje imenskih entitet (leta 1970).
- Ekstrahirani podatki omogočajo nove načine poizvedovanja, organizacije, analize in predstavitve podatkov (biomedicinska domena).
- Zbiranje informacij iz različnih delov besedila
- Najti in razumeti določene dele besedila
- Cilj ekstrakcije informacij je:
  - Pridobiti strukturirane podatke iz nestrukturiranih ali pol-strukturiranih podatkovnih virov.
  - Postaviti informacije v natančno pomensko obliko, ki omogoča računalniškim algoritmom nadaljnja sklepanja.

Ekstrakcija informacij

- Naloga sistemi za ekstrakcijo informacij je, da ekstrahirajo jasne dejanske informacije (kdo, kaj, komu, kdaj, kje).
- Primer
  - Iz besedila ugotovi, kdo je predavatelj, kaj predava, kje pradava, kdaj predava.
  - predavatej(“Janez Novak”, “Slovenščina”,“Maribor”,“Ponedeljek”,“Poletni semester”)

Preprosta ekstrakcija informacij

- Obstaja v različni programski opremi.
- Odjemalec elektronske pošte, na osnovi vsebine ponuja določene aktivnost. Ko razpozna datum, nam ponudi ustvarjanje dogodka.
- Pri spletnem iskanju, imamo ponujene informacije glede na vsebino povpraševanja (npr. kje se nahaja iskano mesto).
- Večina preprostih ekstrakcij informacij uporablja regularne izraze.

Prepoznavanje imenskih entitet

- Zelo pomembno opravilo pri ekstrakciji informacij je prepoznavanje imenskih entitet.
- Potrebno je **poiskati** in **klasificirati** imena znotraj besedila.
  - S pomočjo leksikalne analize ugotovimo imena.
  - Vsa imena klasificiramo npr.: ime človeka, organizacije, lokacije itd.
- Presenečenja te volitve vsaj pri vrhu niso prinesle sprememb. **SDS** je, kot so napovedovale tudi vse javnomnenjske raziskave, nesporna zmagovalka, saj ji je pripadel vsak četrti glas. V novem, osmem sklicu državnega zbora bo imela 25 poslancev. Drugouvrščena, **Stranka Marjana Šarca**, jih bo imela 13. **SD** in **SMC** sta dosegli skoraj identični rezultat, zastopalo ju bo po deset poslancev, **Levica** jih bo imela devet. Sledijo **Nova Slovenija** s sedmimi poslanci, **Stranka Alenke Bratušek** in **DeSUS** s po petimi ter **Slovenska nacionalna stranka** s štirimi. (vir: 24ur.com, 4. 6. 2018)

Prepoznavanje imenskih entitet

- Uporabnost
  - Indeksiranje in povezovanje imenskih entitet (npr. povezave spletnih strani).
  - Določanje na koga ali kaj se nanaša analiza sentimenta.
  - Relacije pri ekstrakciji informacij so povezane z imeni entitet.
  - Sistemi vprašanj in odgovorov, pogosto uporabljajo poimenovanje entitet.

Ocenjevanje prepoznavanja imenskih entitet

- Uporaba kontingenčne tabele (tp,tn,fp,fn)
- Preciznost (angl. *Precision*)
- Priklic (angl. *Recall*)
- Mera F1.

Primer prepoznavanja imenskih entitet

- Miha (oseba)
- Maribor (mesto)
- Merkator (podjetje)
- Nova **Ljubljanska banka** (podjetje)
  Problem ugotavljanja mej, ki določata imensko entiteto.
  Rezultate je delno pravilen.
  Ni nujno, da gre za imensko entiteto.

Metode za prepoznavanje imenskih entitet

- Ročno zapisani regularni izrazi.
  - Uporabno v primeru lepo strukturiranih spetnih straneh.
  - Ponavadi uporabno za nekatere omejene splošne entitete v nestrukturiranih besedilih (npr. datumi in telefonske številke).
  - Pomagamo si lahko s pomočjo:
    - oblikoslovnega označevanja besedil (angl. part-of-speech tagging),
    - sintaksičnega razpornavanja (identifikacija fraz) in
    - semantične klasifikacije besed (npr. s pomočjo orodja WordNet).

Metode za prepoznavanje imenskih entitet

- Uporaba klasifikatorjev.
  - Generativne in diskriminatorne metode.
  - Klasifikacija besed v dva razreda: “za ekstrakcijo” in “ni za ekstrakcijo”.
  - V določenih preprostih domenah dosegajo zavidljive rezultate.
  - Primer: ugotavljanje spremembe elektronskega naslova.
- Sekvenčni modeli.
  - Učenje:
    - Učni dokumenti,
    - vsak leksikalni simbol ima oznako “imenska entiteta” oz. “ostalo”,
    - načrtovanje značilk in
    - učenje sekvenčnih klasifikatorjev.
  - Testiranje:
    - Testna množica,
    - klasifikacija in
    - ocenjevanje kvalitete klasifikacije.

Sekvenčni modeli

	IO kodiranje	IOB kodiranje
Miha	OSEBA	B-OSEBA
je	DRUGO	DRUGO
Marku	OSEBA	B-OSEBA
pokazal	DRUGO	DRUGO
nov	DRUGO	DRUGO
program	DRUGO	DRUGO
Janeza	OSEBA	**B-OSEBA**
Novaka	OSEBA	**I-OSEBA**
	c+1 oznak	2c+1 oznak
Časovna zahtevnost	manjša	večja
Uporabnejše	DA	NE

B - Začetek imenske entitete

I - Nadaljevanje imenske entitete

Značilke za sekvenčno označevanje

- Besede
  - Trenutna beseda (naučen slovar)
  - Prejšnja/naslednja beseda (kontekst)
- Drugi načini klasifikacije
  - Oblikoslovno označevanje
- Kontekst oznak
  - Prejšnja in naslednja oznaka

Viri

- Dan Jurafsky, Chris Manning, Natural Language Processing `http://web.stanford.edu/~jurafsky/NLPCourseraSlides.html`
- Slavnko Žitnik, Iterativno pridobivanje semantičnih podatkov iz nestrukturiranih besednih virov, doktorska disertacija, 2014.

Globoko učenje - uvod

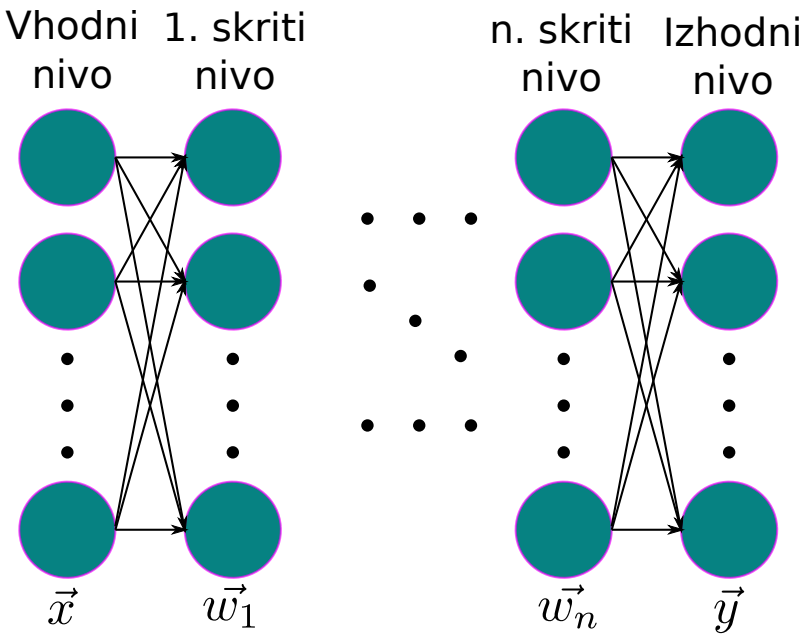
(angl. Deep Learning)

Kaj je globoko učenje

- Je področje strojnega učenja
- Večina metod strojnega učenja deluje dobro.
  - Razlog temu je človeško zasnovana predstavitev in določanje vhodnih značilk.
  - Globoko učenje ima večplastno predstavitev in od tod tudi prihaja njeno ime.
  - Pri izboljševanju metod se ljudje dosti naučijo o določenem problemu in hkrati pomagajo programom, da bolje opravljajo svojo nalogo.
- Na strojno učenje lahko gledamo kot na optimizacijo uteži, da dosežemo najboljšo možno predikcijo.

Kaj je globoko učenje

- Samodejno se poskuša ugotoviti dobre značilke oz. predstavitve (angl. feature learning ali representation learning)
- Algoritmi globokega učenja poskušajo "naučiti" večplastne predstavitve, da dobijo primerne izhode.
- Učenje temelji na vhodnih vzorcih $\vec{\mathbf{x}}$. To so lahko besede, značilke, zvok itd.



Zakaj uporabljati globoko učenje

- Ročno določene značilke so ponavadi preveč specifične, nepopolne in za njihov razvoj in validacijo potrebujemo ogromno časa.
- Določanje značilk s pomočjo strojnega učenja je možno prilagoditi določeni domeni. Ta postopek je relativno hiter.
- Globoko učenje omogoča zelo fleksibilno oz. skoraj univerzalno ogrodje za predstavitev sveta. To vključuje tako vizualni svet kot tudi jezikovne informacije.
- Globoko učenje lahko uporablja nenadzorovano (na osnovi neoznačenega besedila) kot tudi nadzorovano učenje (s pomočjo označenega besedila).

Zakaj uporabljati globoko učenje

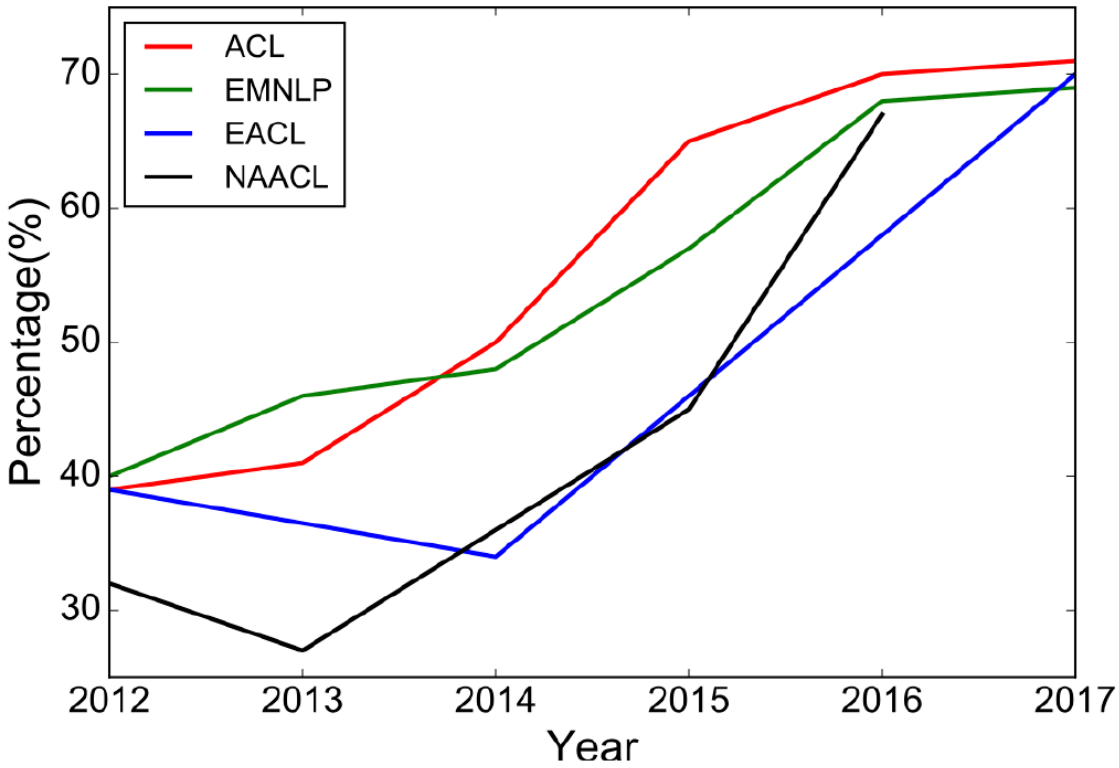
- Približno leta 2010 je globoko učenje začelo premagovati ostale metode strojnega učenja.
- S pomočjo globokega učenja je računalnik premagal svetovnega prvaka v igri go leta 2016.
- Kaj je pripomoglo k razvoju globokega učenja?
  - Velike količine podatkov.
  - Hitrejši in večjedrni procesorji. To vključuje tako centralno procesno enoto kot tudi grafično procesno enoto.
  - Novi modeli, algoritmi in ideje
    - Boljše in bolj fleksibilno učenje vmesnih plasti.
    - Učinkovit združen sistem učenja.
    - Učinkovite metode učenja, ki omogočajo boljši prenos informacij med konteksti kakor tudi med domenami.
  - Izboljšane zmogljivosti pri govoru, vidu in tudi pri jezikovnih tehnologijah.

Globoko učenje in govor

- Prvi preboj globokega učenja se je zgodil pri razpoznavanju govora s pomočjo velikih učnih množic.
  G. E. Dahl, D. Yu, L. Deng in A. Acero, Čontext-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition,"v IEEE Transactions on Audio, Speech, and Language Processing, letn. 20, št. 1, str. 30-42, 2012.
- Na področju računalniškega vida so prvi preboj naredili v naslednjem članku:
  Alex Krizhevsky, Ilya Sutskever in Geoffrey Hinton, "ImageNet Classification with Deep Convolutional Neural Networks". Neural Information Processing Systems, 25, 2012, doi=10.1145/3065386.

Globoko učenje in jezikovne tehnologije

Število objav, ki obravnavajo globoko učenje in jezikovne tehnologije.



Vir: Tom Young, Devamanyu Hazarika, Soujanya Poria in Erik Cambria,

Recent Trends in Deep Learning Based Natural Language Processing, arXiv: 1708.02709

Globoko učenje in jezikovne tehnologije

- Združuje ideje in cilje jezikovnih tehnologij z globokim učenjem.
- Cilj je rešiti probleme jezikovnih tehnologij.
- Globoko učenje je aplicirano na različne
  - **nivoje** jezikovnih tehnologij: govor, besede, sintaksa, semantika itd.
  - **orodja** jezikovnih tehnologij: označevanje besedila, razpoznavanje entitet itd.
  - **aplikacije** jezikovnih tehnologij: strojno prevajanje, analiza sentimenta, sistemi vprašanj in odgovorov.

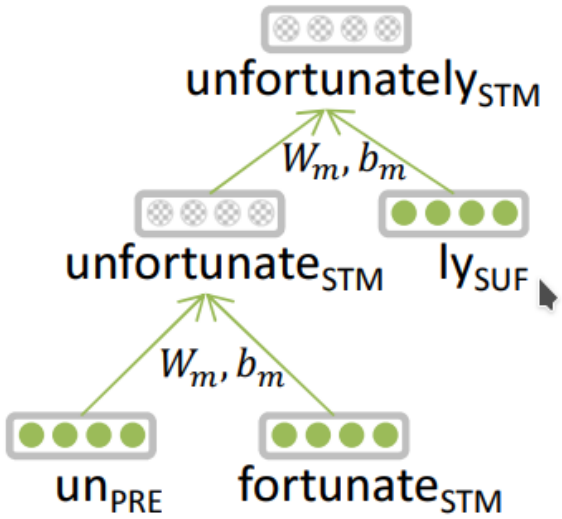
Predstavitev pomena besede - vektor

- V jezikovnih tehnologija se vse začne z besedami.
- Za predstavitev pomena besed uporabimo vektorje.
- Vektorji so velikih dimenzij (minimalno 25 dimenzij).
- Besedo postavimo v n-dimenzionalni prostor.
- Besede s podobnim pomenom so združene v grozde.
- Besede, ki se nahajajo v okolici besede **žaba**:
  Rosnica, Sekulja, Zelena žaba

Predstavitev nivojev jezikovnih tehnologij

- Morfologija
- Tradicionalno: besede so sestavljene iz morfemov oz. iz manjših besednih enot s samostojnim semantičnim pomenom.
- Globoko učenje
  - Vsak morfem je vektor.

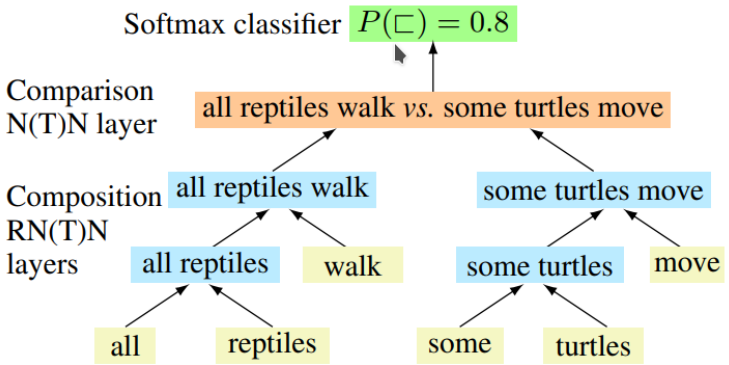
    Thang Luong, Richard Socher in Christopher Manning, Better Word Representations with Recursive Neural Networks for Morphology, Proceedings of the Seventeenth Conference on Computational Natural Language Learning, str. 104-113, 2013
- Nevronske mreže lahko natančno določijo struktura stavkov, ki vključuje tudi razlago.



Thang Luong in ostali, 2013

Predstavitev nivojev jezikovnih tehnologij

- Sintaksa
- Tradicionalen: Lambda račun
  - Natančno načrtovane funkcije.
  - Vhod v funkcijo je neka druga funkcija.
  - Ni notacij o podobnosti in nejasnosti jezika.
- Globoko učenje:
  - Vsaka beseda, vsaka fraza in vsak logični izraz je vektor.
  - Nevronska mreža združuje dva vektorja v enega.



Samuel R. Bowman, Christopher Potts in Christopher D. Manning, Recursive Neural Networks Can Learn Logical Semantics, Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, str. 12-21, 2015

Aplikacije globokega učenja

- Analiza sentimenta
  Lahko se uporabi enak model globokega učenja kot je bil uporabljen za morfologijo, sintakso in logično semantiko
- Sistem vprašanj in odgovorov
  Lahko se uporabi struktura globokega učenja in dejstva so lahko shranjena v vektorjih
- Ustvarjanje odgovorov
  Je aplikacija zmogljivih in splošnih jezikovnih modelov, ki temeljijo na rekurentnih nevronskih mrežah.
- Strojno prevajanje
  Izvorna poved se preslika v vektor, nato se ustvari izhodna poved.

  [Sutskever in ostali 2014, Bahdanau in ostali, 2014, Luong in Manning, 2016]

  Nekateri jeziki v prevajalniku Google.
- **Zaključek:** Vektorji na vseh nivojih predstavitev!

Vir

Christopher Manning in Richard Socher, Natural Language Processing with Deep Learning, CS224N/Ling284

Vektorji besed

(angl. Word Vectors)

Kako predstavimo pomen besed?

- Pomen (SSKJ):
  - Kar beseda vsebuje glede na označevani pojem, predmet.
  - Poudarja bistvene, tipične lastnosti česa, kot jih določa prilastek.
  - Pozitivne lastnosti, značilnosti česa.
  - Izraža nepotrebnost česa.
- Splošen način lingvističnega razmišljanja o pomenu:
  - Označevalec $<=>$ označeno (ideja o zadevi) $=$ denotacija

Kako dobimo uporaben pomen v računalniku?

- Uporaba taksonomij kot je npr. WordNet.
  - Samostalnik
    - mesojedec je hipernim psa
    - pes je hiponim mesojedca
    - volk je kohiponim psa in pes je kohiponim volka
    - stavba je holonim okna
    - okno je meronim stavbe
  - Glagol
    - potovati je hipernim glagola gibati se
    - šepetati je troponim glagola govoriti
    - glagol spati vsebuje glagol smrčati; prvi je pogoj za drugega
    - kohiponim: glagola šepetati in kričati, ki imata skupen hipernim – glagol govoriti
  - Pridevnik
    - hišni prag – pridevnik hišni, ki izvira iz samostalnika hiša
    - deležniki: pojoča deklica

Diskretna predstavitev besed

- Predstavlja dober vir.
- Ima probleme s podrobnostmi. Npr. **strokovnjak** je na seznamu sinonimov besede **dobro**. To drži le v določenih kontekstih.
- Manjkajo novi pomeni besed (nemogoče vzdrževati).
- Subjektivno.
- Potreben je človeški trud za izdelavo in vzdrževanje.
- Ni možno določiti natančne podobnosti.

Diskretna predstavitev besed

- Velika večina metod tradicionalnih jezikovnih tehnologij obravnava besede kot simbole: soba, avto, tek.
- V vektorskem prostoru bi jih tako predstavili z vektorji, katerih ene komponenta ima vrednost 1 in vse ostale komponente vrednost 0. To predstavitev imenujemo "**one hot**" kodiranje.
  - soba $= \{0, 0, 0, 0, 0, 1, 0, 0, 0\}$
  - avto $= \{0, 0, 1, 0, 0, 0, 0, 0, 0\}$
  - tek $= \{0, 0, 0, 1, 0, 0, 0, 0, 0\}$
- Dimenzija vektorjev bi morali biti enaka velikosti slovarja.
- Dva vektorja sta med seboj pravokotna. To pomeni, da je težko določiti podobnost med dvema vektorjema oz. besedama.
  - Lahko bi uporabili WordNet ampak na tak način imamo problem nepopolnosti.
  - **Poskusimo zakodirati podobnost v sam vektor.**

Predstavitev besed s pmočjo konteksta

- Distribucijska semantika: **Pomen besede je podan z besedami, ki se pogosto nahajajo v njihovi bližini.**
- Ena od najuspešnejših idej sodobnih metod jezikovnih tehnologij.
- Ko se beseda pojavi v besedilu, njen konteks predstavlja množica besed v njeni okolici (znotraj fiksne velikosti okna).
- Za izgradnjo predstavitve besede *b* uporabimo več kontekstov.
- Konteksti za predstavitev besede **kolo**:
  - Popoldne so zaplesali **kolo**.
  - 36 prvenstveno **kolo** je bilo ključno za Maribor.
  - Zamenjal je **kolo** avtomobila.
  - Sedel je na **kolo** in se odpeljal.
  - Kupil je novo motorno **kolo**.

Vektorji besed

- Angleški izrazi: word vectors, word embeddings, word representations.
- Zgradili bomo vektor za vsako besedo.
- Vektor bo omogočal izbiro podobnih vektorjev besed, ki se pojavljajo v podobnih kontekstih.
- Besede so porazdeljene po predstavitvi (angl. distributed representation).

Ogrodje za učenje vektorjev besed (word2vec)

- Enostaven in hiter model.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, in Jeff Dean, "Distributed Representations of Words and Phrases and their Compositionality", v Advances in Neural Information Processing Systems 26, str. 3111–3119, 2013.
- Veliki učni korpus.
- Vsaka beseda v slovarju fiksne dolžine je predstavljena s pomočjo vektorja.
- Sprehodimo se skozi besedilo, kjer je osrednja beseda c in besede konteksta so o.
- Uporabi se podobnost vektorjev c in o, da se izračuna verjetnost o za podani c in obratno.
- Nadaljuje se s prilagajanjem vektorjev, tako da se maksimira izračunane verjetnosti.

Ogrodje za učenje vektorjev besed (word2vec)

- Primer okna (velikosti 2) in izračun verjetnosti:
- Centralna besede b_t

Kako	se	**novi**	koronavirus	prenaša
$P(b_{t-2}\|b_t)$	$P(b_{t-1}\|b_t)$	b_t	$P(b_{t+1}\|b_t)$	$P(b_{t+2}\|b_t)$

se	novi	**koronavirus**	prenaša	med
$P(b_{t-2}\|b_t)$	$P(b_{t-1}\|b_t)$	b_t	$P(b_{t+1}\|b_t)$	$P(b_{t+2}\|b_t)$

Ogrodje za učenje vektorjev besed (word2vec)

- za vsak položaj $t = 1, ..., T$ napovemo okoliške besede s pomočjo okna določene velikosti m in podane centralne besede w_j.

$$Verjetnost = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(b_{t+j} | b_t; \theta)$$

- θ - spremeljivke, ki jih je potrebno optimizirati.
- Ocenitvena funkcija (angl. objective, cost, loss function) $J(\theta)$:

$$J(\theta) = -\frac{1}{T} \log(L(\theta)) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log \ P(b_{t+j} | b_t; \theta)$$

- Minimizacija ocenitvene funkcije <=> maksimizacija natančnosti predikcije.

Ogrodje za učenje vektorjev besed (word2vec)

- Želimi minimizirati naslednjo ocenitveno funkcijo:

$$J(\theta) = -\frac{1}{T}\log(L(\theta)) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log\ P(b_{t+j}|b_t;\theta)$$

- Kako izračunati $P(b_{t+j}|b_t;\theta)$?
- Uporabili bomo dva vektorja za vsako besedo.
  - v_b ko je b centralna beseda.
  - u_b ko je b beseda iz konteksta.
- Za centralno besedo c in besedo o iz konteksta dobimo:

$$P(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{b\in s}\exp\left(u_b^T v_c\right)}$$

Ogrodje za učenje vektorjev besed (word2vec)

- Primer okna in računanja verjetnosti $P(b_{t+j}|b_t)$
- koronavirus = korona...
- $P(u_{se}|v_{korona...}) = P(se|korona...; u_{se}, v_{korona}, ..., \theta)$

se novi **korona...** prenaša med

$P(u_{se}|v_{korona...})P(u_{novi}|v_{korona...})$ b_t $P(u_{\text{prenaša}}|v_{korona...})P(u_{med}|v_{korona...})$

Ogrodje za učenje vektorjev besed (word2vec)

$$P(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{b \in s} \exp\left(u_b^T v_c\right)}$$

- Skalarni produkt dveh vektorjev določa podobnost dveh besed: $u_o^T v_c = \sum_{i=1}^{n} u_{o,i} \cdot v_{c,i}$
- Večja vrednost skalarnega produkta pomeni večjo verjetnost.
- Normalizacija skozi celoten slovar: $\sum_{b \in s} exp(u_b^T v_c)$
- To je primer *softmax* funkcije $\mathbb{R}^n \rightarrow (0,1)^n$

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- S pomočjo *softmax* funkcije preslikamo vrednosti x_i v verjetnostno porazdelitev p_i.
  - *max* - povečuje verjetnost za večje x_i.
  - *soft* - nekaj verjetnosti dodeli tudi malim x_i
  - Pogosto uporabljeno pri strojnem učenju.

Učenje modela s pomočjo optimizacije

- Izračun gradienta za vse vektorje.
- θ predstavlja vse parametre v modelu.
- V našem primeru imamo d-dimenzionalne vektorje in V besed.

$$\theta = \begin{bmatrix} v_{kolo} \\ v_{avto} \\ \dots \\ v_{sonce} \\ u_{kolo} \\ u_{avto} \\ \dots \\ u_{sonce} \end{bmatrix} \in \mathbb{R}^{2dV}$$

- Vsak beseda ima dva vektorja.
- Te parametre optimiziramo tako, da se premikamo v smeri gradienta oz. uporabimo algoritem Gradientni spust.

Gradient

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log\ P(b_{t+j}|b_t;\theta)$$

$$\frac{\partial}{\partial v_c}\log\frac{\exp\left(u_o^T v_c\right)}{\sum_{b\in s}\exp\left(u_b^T v_c\right)} =$$

$$\frac{\partial}{\partial v_c}\log\ \exp\left(u_o^T v_c\right) - \frac{\partial}{\partial v_c}\log\sum_{b\in s}\exp\left(u_b^T v_c\right)$$

Gradient

$$\frac{\partial}{\partial v_c} u_o^T v_c = u_o; \frac{\partial}{\partial (v_c)_1} u_o^T v_c = u_{o_1} v_{c_1} + u_{o_2} v_{c_2} + \ldots + u_{o_d} v_{c_d} = u_{o_1}$$

$$\frac{\partial}{\partial v_c} \log \sum_{b \in s} \exp\left(u_b^T v_c\right) = \frac{1}{\sum_{b \in s} \exp\left(u_b^T v_c\right)} \cdot \frac{\partial}{\partial v_c} \sum_{x \in s} \exp\left(u_x^T v_c\right) =$$

$$\frac{1}{\sum_{b \in s} \exp\left(u_b^T v_c\right)} \cdot \sum_{x \in s} \frac{\partial}{\partial v_c} \exp\left(u_x^T v_c\right) =$$

$$\frac{1}{\sum_{b \in s} \exp\left(u_b^T v_c\right)} \cdot \sum_{x \in s} \left(\exp\left(u_x^T v_c\right) \cdot \frac{\partial}{\partial v_c} u_x^T v_c\right) =$$

$$\frac{1}{\sum_{b \in s} \exp\left(u_b^T v_c\right)} \cdot \sum_{x \in s} \left(\exp\left(u_x^T v_c\right) \cdot u_x\right)$$

Pravilo verige: $y = f(u); \quad u = g(x); \quad y = f(g(x)); \rightarrow \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}u} \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}f(u)}{\mathrm{d}u} \frac{\mathrm{d}g(x)}{\mathrm{d}x}$

$\log'(x) = \frac{1}{x}; \quad \exp'(x) = \exp(x); f(x) = x \rightarrow f(x)' = 1$

Gradient

$$\frac{\partial}{\partial v_c} \log(P(o|c)) = u_o - \frac{\sum_{x \in s} \left(\exp\left(u_x^T v_c \right) \cdot u_x \right)}{\sum_{b \in s} \exp\left(u_b^T v_c \right)} =$$

$$u_o - \sum_{x \in s} \left(\frac{\exp\left(u_x^T v_c \right)}{\sum_{b \in s} \exp\left(u_b^T v_c \right)} \cdot u_x \right) =$$

$$u_o - \sum_{x \in s} \left(p(b|c) \cdot u_x \right)$$

- $\frac{\partial}{\partial v_c} \log(P(o|c))$ - smer v večdimenzionalnem prostoru
- u_o - opazovana predstavitev
- $\sum_{x \in s} p(b|c) \cdot u_x$ - model oz. pričakovana predstavitev

Gradientni spust

- Matrična notacija: $\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$
- Notacija za določen parameter: $\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$
- α - velikost koraka oz. stopnja učenja.
- Problem: $J(\theta)$ je funkcija vseh oken v korpusu in je časovno zelo zahtevna.
- Rešitev: Uporaba algoritma Stohastični gradientni spust. Iterativno vzorčimo okna in jih posodabljamo po vsaki epohi.
- Zaradi strojne opreme, ki se lahko izvaja paralelno, vzorčimo okna velikosti 32, 64 itd.

Stohastnični gradientni spust

- Iterativno računamo gradiente za vsako okno.
- V vsakem oknu imamo največ 2m+1 besed.
- $\nabla_\theta J_t(\theta)$ je zelo redek.

$$\nabla_\theta J_t(\theta) = \begin{bmatrix} 0 \\ \dots \\ \nabla_{b_{kolo}} \\ \dots \\ 0 \\ \nabla_{b_{avto}} \\ \dots \\ \nabla_{b_{sonce}} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Stohastnični gradientni spust

- Posodabljanje vektorjev besed, ki se dejansko pojavljajo.
- Rešitev: uporaba operatorjev za posodabljanje redkih matrik ali uporaba preslikave vektorjev besed (angl. hash).
- V primeru ogromne količine besed in distribuiranega sistema, si ne moremo privoščiti razpošiljanje ogromnih posodobitev!

Ogrodje za učenje vektorjev besed (word2vec)

- Dva tipa modelov
  - Skip-grams (SG)
    Napove besede konteksta (neodvisno od položaja) na osnovi centralne besede. **Model ki smo ga predstavili.**
  - Continuous Bag of Words (CBOW)
    Napove centralno besedo glede na (vrečo besed) besede konteksta.
- Učinkovitost učenja.
  - Negativno vzorčennje.
  - Naïve softmax - je enostavna in časovno zahtevna metodo učenja.

Skip-gram model z negativnim vzorčenjem

- Normalizacija je časovno zahtevna.
  $P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{b \in s} exp(u_b^T v_c)}$
- Uporaba binarne logistične regresije za pravi par centralne besede in besede v njenem kontekstnem oknu v primerjavi z več šumnimi pari kjer centralno besedo povežemo z naključno besedo.
- Članek: "Distributed Representations of Words and Phrases and their Compositionality" (Mikolov et al. 2013)
  - Ocenitvena funkcija: $J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J_t(\theta)$

$$J(\theta) = log\ \sigma(b_o^T\ b_c) + \sum_{i=1}^{k} E_{j \sim P(b)}[log\ \sigma(-b_j^T\ b_c)] \qquad (1)$$

  - Sigmoidna funkcija $\sigma(x) = \frac{1}{1+e^{-x}}$

Skip-gram model z negativnim vzorčenjem

- Notacija v našem kontekstu:
  $J_{negativno\_vzorčenje}(\mathbf{o}, \mathbf{b_c}, \mathbf{U}) =$
  $-log(\sigma(b_o^T\ b_c)) - \sum_{k=1}^{K} log(\sigma(-b_k^T\ b_c))$
- Vzamemo k negativnih vzorcev (uporaba verjetnosti besed)
- Maksimiramo verjetnost za realne besede konteksta.
- Minimiziramo verjetnosti za naključno izbrane besede.
- Verjetnost vzorčenja besed: $P(b) = \frac{U(b)^{\frac{3}{4}}}{Z}$
  - $U(b)$ - porazdelitev unigramov
  - Z - normalizacija
  - $\frac{3}{4}$ - manj frekventne besede bodo pogosteje izbrane.

Matrika sopojavljanja - *Co-occurrence matrix*

Nenavadno je iti skozi celoten korpus večkrat. Zakaj preprosto ne uporabimo statistike o tem, katere besede se pojavljajo ena blizu druge?

- Okno velikosti 1 (ponavadi se uporablja velikost od 5 do 10)
- Simetrična

```
Globoko učenje me zanima.  Jezikovne tehnologije so zanimive.  Globoko
učenje in jezikvone tehnologije so uporabne.
```

števec	globoko	učenje	me	zanima	jezikovne	tehnologije	so	zanimive	in	uporabne
globoko	0	2	0	0	0	0	0	0	0	0
učenje	2	0	1	0	0	0	0	0	1	0
me	0	1	0	1	0	0	0	0	0	0
zanima	0	0	1	0	0	0	0	0	0	0
jezikovne	0	0	0	0	0	2	0	0	1	0
tehnologije	0	0	0	0	2	0	2	0	0	0
so	0	0	0	0	0	2	0	1	0	1
zanimive	0	0	0	0	0	0	1	0	0	0
in	0	1	0	0	1	0	0	0	0	0
uporabne	0	0	0	0	0	0	1	0	0	0

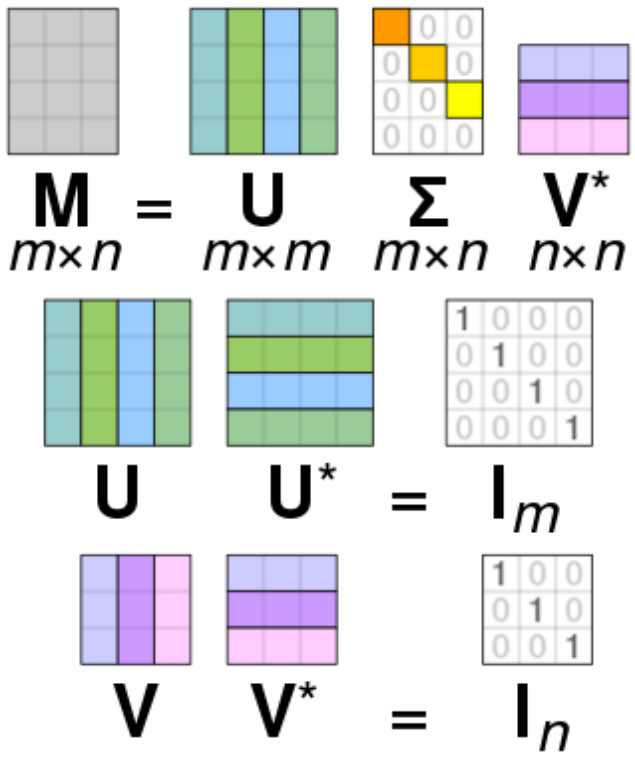
Matrika sopojavljanja - gradnja

- Uporaba manjšega okna
  - Podobno word2vec
  - Uporablja okno v okolici besede (lokalnost)
  - Zajema sintaksične in semantične informacije
- Okno je velikosti odstavka ali celotnega dokumenta
  - Podaja splošno tematiko
  - Analiza sentimenta

Matrika sopojavljanja - značilnosti

- Enostavno določanje vektorjev
  - Z večanjem slovrja se večja dimenzija vektorjev
  - Velika dimenzija zahteva dosti pomnilnika (čeprav so redki).
  - Medeli klsifikacije morajo obravnavati redkost (manjša robustnost).
- Manj dimenzionalni vektorji
  - Uporabnejši
  - Shranimo pomembnejše informacije v vektroje manjših dimenzij - gosti vektorji (angl. dense vectors)
  - Ponavadi imajo od 25 do 1000 dimenzij (podobno kot word2vec)
  - Dekompozicija singularnih vrednosti (angl. singular value decomposition)

Dekompozicija singularnih vrednosti



Vir: wikimedia.org

Dekompozicija singularnih vrednosti

- Na preprostih števcih ne daje dobre rezultate.
- Skaliranje števcev lahko pomaga.
  - Problem: Besede, ki nimajo semantičnega pomena (npr. the, he, has) so zelo frekventne (velik vpliv sintakse).
  - Rešitev
    - Uporaba funkcije *log* nad števci
    - Uporaba manj frekventnih besed
    - Ignoriranje besed, ki nimajo semantičnega pomena
- V skaliranih vektorjih se pojavijo vzorci semantike.
  - Drive $\rightarrow$ Driver
  - Teach $\rightarrow$ Teacher

Neposredna predikcija in predikcija na osnovi štetja

- Na osnovi štetja (dekompozicijska metoda)
  - Hitro učenje.
  - Učinkovita uporaba statistik.
  - Uporabno za zajemanje podobnosti besed.
  - Nesorazmeren pomen dodan velikim številom.
- Neposredna predikcija (skip gram)
  - Skalirajo se z velikostjo korpusa.
  - Neučinkovita uporaba statistik.
  - Omogočajo izboljšave drugih nalog jezikovnih tehnologij.
  - Lahko zajamejo kompleksne vzorce med besedami in ne samo njihovo podobnost.

Kodiranje pomena v razlikah vektorjev

Razmerje verjetnosti sopojavljanja lahko kodira pomen komponent

	x = trdno	x = plinasto	x = voda	x = naključno
$P(x\|led)$	velika	mala	velika	male
$P(x\|para)$	mala	velika	velika	mala
$\frac{P(x\|led)}{P(x\|para)}$	velika	mala	∼1	∼1

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Kodiranje pomena v razlikah vektorjev

Razmerje verjetnosti sopojavljanja lahko kodira pomen komponent

	x = trdno	x = plinasto	x = voda	x = naključno
$P(x\|led)$	$1{,}9\text{x}10^{-4}$	$6{,}6\text{x}10^{-5}$	$3{,}0\text{x}10^{-3}$	$1{,}7\text{x}10^{-5}$
$P(x\|para)$	$2{,}2\text{x}10^{-5}$	$7{,}8\text{x}10^{-4}$	$2{,}2\text{x}10^{-3}$	$1{,}8\text{x}10^{-5}$
$\frac{P(x\|led)}{P(x\|para)}$	8,9	$8{,}5\text{x}10^{-2}$	1,36	0,96

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Kodiranje pomena v razlikah vektorjev

- Kako določimo razmerja verjetnosti sopojavljanja za linearne pomenske komponente v prostoru vektorjev besed?
- Bilinearni model: $w_i \cdot w_j = logP(i|j)$
- Razlika vektorjev: $w_x \cdot (w_a - w_b) = log\frac{P(x|a)}{P(x|b)}$

Združitev obeh pristopov

GloVe (Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.)

Model log-bilinear: $w_i \cdot w_j = logP(i|j)$

Razlike vektorjev: $w_x \cdot (w_a - w_b) = log\frac{P(x|a)}{P(x|b)}$

Funkcija izgube: $J = \sum_{i,j=1}^{V} f(X_{ij})(w_u^T \tilde{w}_j + b_i + \tilde{b}_j - logX_{ij})^2$

- Hitro učenje
- Možnost velikih korpisov
- Dobri rezultati tudi v primeru malih korpusov in malih dimenzij vektorjev

Kako dobri so vektorji besd?

- Notranje ovrednotenje
  - Ovrednotenje s pomočjo določene notranje naloge
  - Hiter izračun
  - Pomaga razumeti sistem
  - Ne vemo kako uporabni so vektorji, dokler jih ne preizkusimo na realni nalogi
- Zunanje ovrednotenje
  - Ovrednotenje na realni nalogi
  - Časovno zahtevno
  - Ne vemo ali je mogoče problem v sistemu vektorjev ali v katerem drugem podsistemu

Notranje ovrednotenje

- Kako dobro kosinusna razdalja po seštevanju vektorjev zajema semantična in sintaksična vpršanja.

  $d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$

  `moški <=> ženska :: krelj <=> kraljica`

- Kaj pa če relacije niso linearne?

Klasifikacija razlik med vektorji besed

- Kompleksen model
- Uporabimo nevronsko mrežo
- “Učimo” parametre nevronske mreže in “položaje” vektorjev
- Uporabimo `embedding layer`

Viri

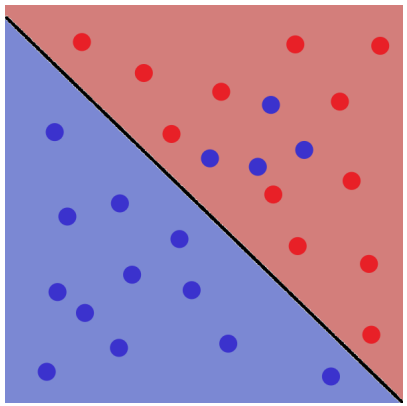
- Christopher Manning in Richard Socher, Natural Language Processing with Deep Learning, CS224N/Ling284
- Rohde in ostali, An Improved Model of Semantic Similarity Base on Lexical Co-Occurrence, 2005
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

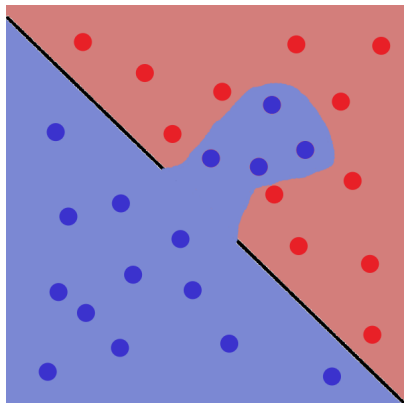
Nevronske mreže

(angl. Neural Networks)

Klasifikator *softmax* in nevronske mreže

- Klasifikatorji *softmax*: $P(y|x) = \frac{\exp(W_y.x)}{\sum_{c=1}^{C} \exp(W_c.x)}$
  - Naučeni parametri W
  - Klasifikator določi linearno odločitveno mejo
- Nevronske mreže
  - V fazi učenja se določajo W in predstavitev oz. porazdelitev besed.
  - Besede so predstavljene z *one-hot* vektorji, ki se preslikajo v vmesni sloj vektorskega prostora. V ta namen se uporabi *embedding* nivo.
  - Uporabimo globoke nevronske mreže, ki naše podatke oz. vektorje preoblikuje več krat. To omogoči ne-linearno klasifikacijo.





Klasifikator *softmax*

Vsebuje tri korake:

- Za vsak razred y izračunamo skalarni produkt: $W_y.x = \sum_{i=1}^{d} W_{yi}x_i = f_y$
- Uporabimo funkcijo *softmax*, da dobimo normalizirano verjetnost: $P(y|x) = \frac{\exp(f_y)}{\sum_{c=1}^{C} \exp(f_c)}$
- Izberemo y z največjo verjetnostjo.

Za vsak učni primer (x,y) si želomo maksimirati verjetnost pravilnega razreda y oz. minimizirati negativno *log* verjetnost:

$$-\log P(y|x) = -\log\left(\frac{\exp(f_y)}{\sum_{c=1}^{C} \exp(f_c)}\right)$$

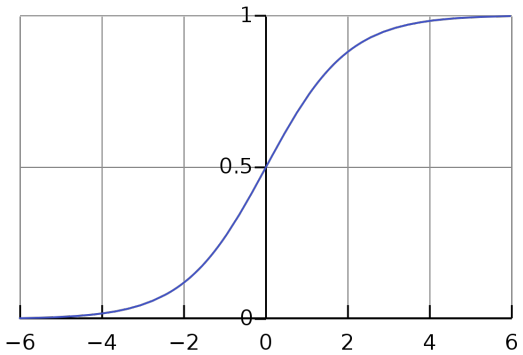
Nevron

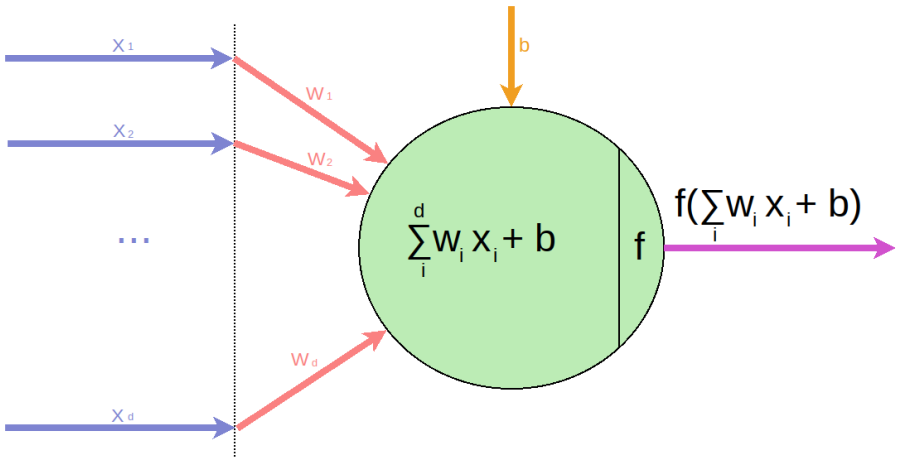
Delovanje binarne logistične regresije je podobno delovanju nevrona.

- f - nelinearna aktivacijska funkcija

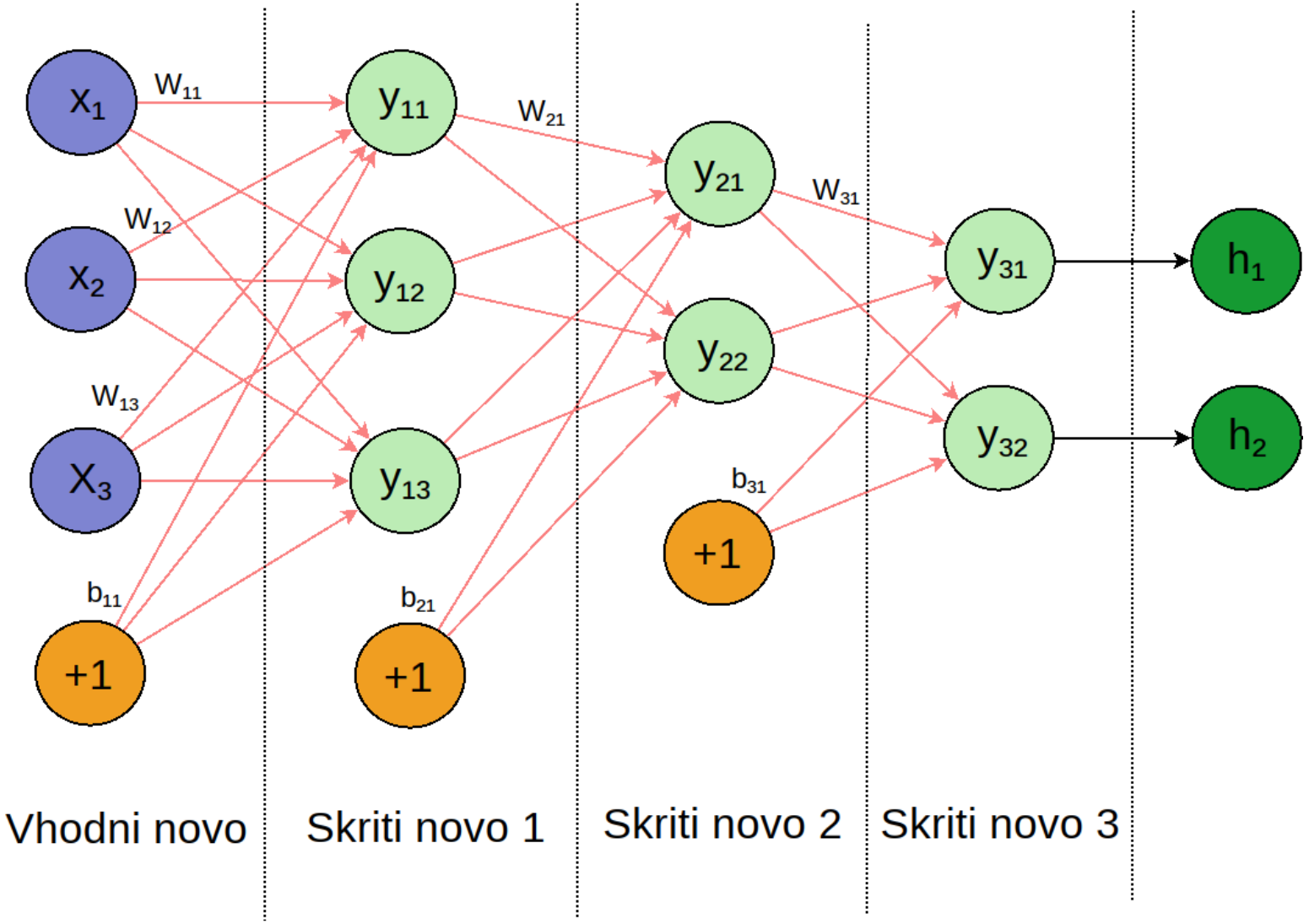
  Sigmoidna: $f(z) = \frac{1}{1+e^{-z}}$

- w_i - uteži
- b - pristranskost
- x_i - vhod





Nevronska mreža



Matrična notacija

- Izhodne vrednosti nevronov
  - $y_{11} = f(W_{11}x_1 + W_{12}x_2 + W_{12}x_3 + b_{11})$
  - $y_{12} = f(W_{21}x_1 + W_{22}x_2 + W_{22}x_3 + b_{12})$
  - itd.
- Matrična notacija
  - $\mathbf{z} = \mathbf{W}\mathbf{x} + b$
  - $\mathbf{y} = f(\mathbf{z})$
- Aktivacijska funkcija se aplicira na vsak element vektorja
  - $\mathbf{y} = f(\{z_1, z_2, z_3\}) = \{f(z_1), f(z_2), f(z_3)\}$

Gradient - ponovitev

- Funkcija z enim vhodom in enim izhodom $f(x) = x^3$
  - Gradient je enak odvodu $f'(x) = \frac{df}{dx} = 3x^2$
- Funkcija z n vhodi in enim izhodom $f(\mathbf{x}) = f(x_1, x_2, ..., x_n)$
  - Gradient je vektor parcialnih odvodov $\frac{\partial f}{\partial \mathbf{x}} = \{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\}$
- Funkcija z n vhodi in m izhodi

  $\mathbf{f}(\mathbf{x}) = \{f_1(x_1, x_2, ..., x_n), f_2(x_1, x_2, ..., x_n), ..., f_m(x_1, x_2, ..., x_n)\}$
  - Gradient je Jakobijeva matrika, ki vsebuje $m \cdot n$ parcialnih odvodov

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \qquad \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)_{i,j} = \frac{\partial f_i}{\partial x_j}$$

Pravilo verige - ponovitev

- Sestavljena funkcija z eno spremenljivko (množenje odvodov)
  - $z = 3y; \quad y = x^2$
  - $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = (3)(2x) = 6x$
- Funkcije z več spremenljivkami (množenje Jakobijevih matrik)
  - $\mathbf{h} = f(\mathbf{z}); \quad \mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$
  - $\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \frac{\partial \mathbf{h}}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \ldots$
  - Jakobijeva matrika za elementarno aktivacijsko funkcijo

$$\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}}\right)_{i,j} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) = \begin{cases} f'(z_i) & \text{če } i == j \\ 0 & \textbf{drugače} \end{cases}$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \begin{bmatrix} f'(z_1) & & 0 \\ & \ddots & \\ 0 & & f'(z_n) \end{bmatrix}$$

- Jakobijeve matrike

$$\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) = \mathbf{W}; \quad \frac{\partial}{\partial \mathbf{b}}\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) = \mathbf{I}; \quad \frac{\partial}{\partial \mathbf{u}}\left(\mathbf{u}^T\mathbf{h}\right) = \mathbf{h}^T$$

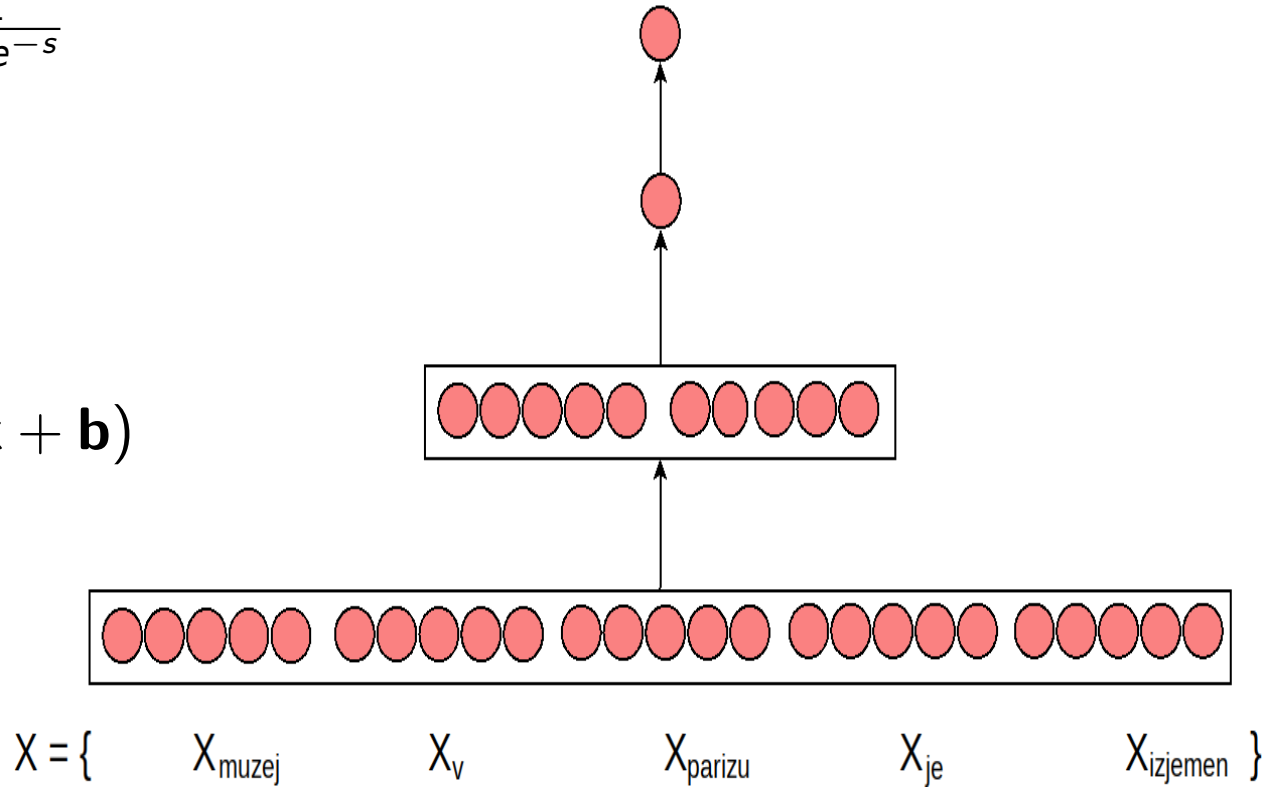
Nevronska mreža

$$\sigma(s) = \frac{1}{1+e^{-s}}$$

$$s = \mathbf{u}^{\mathbf{T}}\mathbf{h}$$

$$\mathbf{h} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

x - vhod



Gradient

Zanima nas gradient izgube (J_t), vendar bomo zaradi poenostavitve izračunali gradient rezultata s oz. $\frac{\partial s}{\partial \mathbf{b}}$.

$$\sigma(s) = \frac{1}{1+e^{-s}}; \quad s = \mathbf{u}^\mathbf{T}\mathbf{h}; \quad \mathbf{h} = f(z); \quad \mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

- Uporabimo pravilo verige: $\frac{\partial s}{\partial \mathbf{b}} = \frac{\partial s}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{b}}$
- Uporabimo Jakobijeve matrike:

$$\frac{\partial s}{\partial \mathbf{h}} = \frac{\partial}{\partial \mathbf{h}} (\mathbf{u}^\mathbf{T}\mathbf{h}) = \mathbf{u}^\mathbf{T}$$
$$\frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \frac{\partial}{\partial \mathbf{z}} (f(z)) = \text{diag}(f'(\mathbf{z}))$$
$$\frac{\partial \mathbf{z}}{\partial \mathbf{b}} = \frac{\partial}{\partial \mathbf{b}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \mathbf{I}$$
$$\frac{\partial s}{\partial b} = \mathbf{u}^\mathbf{T} \text{ diag}(f'(\mathbf{z})) \mathbf{I} = \mathbf{u}^\mathbf{T} \odot f'(\mathbf{z})$$

Gradient

Izračunajmo $\frac{\partial \mathbf{s}}{\partial \mathbf{W}}$.

- Pravilo verige: $\frac{\partial s}{\partial \mathbf{w}} = \frac{\partial s}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{w}}$
- Uporabimo že izračunane vrednosti:

  $\frac{\partial s}{\partial \mathbf{b}} = \frac{\partial s}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{b}} = \delta \frac{\partial \mathbf{z}}{\partial \mathbf{b}}$

  $\frac{\partial s}{\partial \mathbf{W}} = \frac{\partial s}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{W}} = \delta \frac{\partial \mathbf{z}}{\partial \mathbf{W}}$
- Signal napake: $\delta = \frac{\partial s}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \mathbf{u}^{\mathbf{T}} \odot f'(\mathbf{z})$
- Gradient: $\frac{\partial s}{\partial \mathbf{W}} = \delta \frac{\partial \mathbf{z}}{\partial \mathbf{W}} = \delta \frac{\partial}{\partial \mathbf{W}} (\mathbf{W}\mathbf{x} + \mathbf{b}) = \delta^T \mathbf{x}^T$
- Transponirana vektorja: $[n \times m] = [n \times 1][1 \times m]$

Vir

Christopher Manning in Richard Socher, Natural Language Processing with Deep Learning, CS224N/Ling284

Primeri nalog

Drugi kolokvij

- Poglavja
  - Analiza sentimenta
  - Diskriminatorni klasifikatorji v jezikovnih tehnologijah
  - Strojno prevajanje
  - Ekstrakcija informacij in prepoznavanje imenskih entitet
  - Globoko učenje v jezikovnih tehnologijah
  - Vektorji besed

Primeri nalog

Glede na podane značilke iz kratkih komentarjev filmov, ki so tudi označeni z žanrom kateremu pripadajo, določite kateremu žanru pripdajo podane 6. značilke.

1. zabava, par, ljubezen: **komedija**
2. hitro, besno: **akcija**
3. par, leti, hitro, zabavno: **komedija**
4. besno, streljanje, streljanje: **akcija**
5. leti, hitro, streljaj: **akcija**
6. hitro, par, streljaj, leti: **?**

Za izračun najverjetnejšega razreda uporabite klasifikator Naïve Bayes in uporabite glajenje add-1.

Primeri nalog

$P(\text{značilke}|\text{razred}) = P(\text{razreda}) \prod_{x \in \text{značilke}} P(x)$

$P(\text{značilke}|\text{komedija}) = P(\text{komedija}) \cdot P(\text{hitro}|\text{komedija}) \cdot$
$P(\text{par}|\text{komedija}) \cdot P(\text{streljaj}|\text{komedija}) \cdot$
$P(\text{leti}|\text{komedija}) = \frac{2}{5} \cdot \frac{1+1}{7+9} \cdot \frac{2+1}{7+9} \cdot \frac{0+1}{7+9} \cdot \frac{1+1}{7+9} =$
$\frac{2}{5} \cdot \frac{2}{16} \cdot \frac{3}{16} \cdot \frac{1}{16} \cdot \frac{2}{16} = 7,32e-05$

$P(\text{značilke}|\text{akcija}) =$
$P(\text{akcija}) \cdot P(\text{hitro}|\text{akcija}) \cdot P(\text{par}|\text{akcija}) \cdot P(\text{streljaj}|\text{akcija}) \cdot$
$P(\text{leti}|\text{akcija}) = \frac{3}{5} \cdot \frac{2+1}{8+9} \cdot \frac{0+1}{8+9} \cdot \frac{1+1}{8+9} \cdot \frac{1+1}{8+9} =$
$\frac{3}{5} \cdot \frac{3}{17} \cdot \frac{1}{17} \cdot \frac{2}{17} \cdot \frac{2}{17} = 8,62e-05$

6. hitro, par, streljaj, leti: **akcija** $(8,62e-05 > 7,32e-05)$

Primeri nalog

S pomočjo binariziranega Multinominal Naïve Bayes-a in glajenja add-1 določite razred podane povedi na osnovi podatkov v tabeli.

dok.	"dober"	"slabo"	"odlični"	razred
d1	6	0	6	poz.
d2	0	2	4	poz.
d3	2	6	0	neg.
d4	2	10	4	neg.
d5	0	4	0	neg.

Poved: Dober, dober zaplet, odlični igralci, se pa slabo konča.

Primeri nalog

$$P(P|poz.) =$$
$$P(poz.) \cdot P(dober|poz.) \cdot P(odlični|poz.) \cdot P(slabo|poz.) =$$
$$\frac{2}{5} \cdot \frac{1+1}{18+3} \cdot \frac{1+1}{18+3} \cdot \frac{1+1}{18+3} = \frac{2}{5} \cdot \frac{2}{21} \cdot \frac{2}{21} \cdot \frac{2}{21} = 0,00034$$
$$P(P|neg.) =$$
$$P(neg.) \cdot P(dober|neg.) \cdot P(odlični|neg.) \cdot P(slabo|neg.) =$$
$$\frac{3}{5} \cdot \frac{1+1}{28+3} \cdot \frac{1+1}{28+3} \cdot \frac{1+1}{28+3} = \frac{3}{5} \cdot \frac{2}{31} \cdot \frac{2}{31} \cdot \frac{2}{31} = 0,00016$$

Poved: Dober, dober zaplet, odlični igralci, se pa slabo konča.
poz. $\mathbf{(}0,00034 > 0,00016\mathbf{)}$

Primeri nalog

Na osnovi podanih podatkov in modela IBM 1, izračunajte katera poravnava je verjetnejša.

$a_1 = \{2, 3, 4, 5\}$
$a_2 = \{1, 3, 4, 5\}$

e = the coronavirus has been defeated
s = koronavirus je bil premagan

$l = 5,\ m = 4$

Primeri nalog

$t(in|the) = 0,13 \quad t(in|coronavirus) = 0,12$
$t(in|has) = 0,31 \quad t(in|been) = 0,32 \quad t(in|defeated) = 0,12$
$t(koronavirus|the) = 0,11$
$t(koronavirus|coronavirus) = 0,9 \quad t(koronavirus|has) = 0,15$
$t(koronavirus|been) = 0,13 \quad t(koronavirus|defeated) = 0,14$
$t(je|the) = 0,21 \quad t(je|coronavirus) = 0,13$
$t(je|has) = 0,75 \quad t(je|been) = 0,3 \quad t(je|defeated) = 0,14$
$t(bil|the) = 0,2 \quad t(bil|coronavirus) = 0,13$
$t(bil|has) = 0,63 \quad t(bil|been) = 0,82 \quad t(bil|defeated) = 0,24$
$t(premagan|the) = 0,12$
$t(premagan|coronavirus) = 0,13 \quad t(premagan|has) = 0,14$
$t(premagan|been) = 0,11 \quad t(premagan|defeated) = 0,89$

Primeri nalog

e = the coronavirus has been defeated
s = koronavirus je bil premagan

$l = 5, \ m = 4$
$a_1 = \{2, 3, 4, 5\}; \quad a_2 = \{1, 3, 4, 5\}$

$t(koronavirus|coronavirus) = 0,9$
$t(koronavirus|the) = 0,11; \quad t(je|has) = 0,75;$
$t(bil|been) = 0,82; \quad t(premagan|defeated) = 0,89$

$P(s|a_1, e, m) = t(koronavirus|coronavirus) \cdot t(je|has) \cdot t(bil|been) \cdot$
$t(premagan|defeated) = 0.49$
$P(s|a_2, e, m) = t(koronavirus|the) \cdot t(je|has) \cdot t(bil|been) \cdot$
$t(premagan|defeated) = 0.060$

Verjetnejša je poravnava a_1.