Latentna semantička analiza, varijante i primjene

Marasović, Ana

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ana Marasović

LATENTNA SEMANTIČKA ANALIZA, VARIJANTE I PRIMJENE

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Saša Singer

Zagreb, srpanj, 2015

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Sadržaj **iii**

Uvod **1**

1 Model vektorskog prostora **2**
1.1 Reprezentacija dokumenata 2
1.2 Uspoređivanje dokumenata 4

2 Dekompozicija matrice i latentna semantička analiza **6**
2.1 Dekompozcije matrice 6
2.2 Latentna semantička analiza 11
2.3 Dinamička kolekcija 13

3 Probabilistička latentna semantička analiza **17**
3.1 Izvod modela 17
3.2 Usporedba LSA i pLSA 24
3.3 Uspoređivanje dokumenata 25

4 Latentna Dirichletova alokacija **32**
4.1 Dirichletova distribucija 33
4.2 Izvod modela 35

5 Primjeri **41**
5.1 Model vektorskog prostora 41
5.2 Modeliranje tema 43

Bibliografija **48**

Uvod

Svatko od nas, gotovo svakodnevno, pokušava pronaći dokument (stranicu na Internetu), vezan za informaciju koja nas zanima. U tu svrhu smišljamo upit, za koji je najvjerojatnije da ćemo pronaći relevantne dokumente. Koristeći znanje i iskustvo, to činimo mehanički i jako brzo. Međutim, izrada računala sa sposobnošću pronalaska najrelevantnijih dokumenata u kolekciji, je izazovan zadatak za znanstvenike u području strojnog učenja (eng. machine learning) i obrade prirodnog jezika (eng. natural language processing). Njegovo rješavanje dodatno komplicira rapidan rast dostupnih dokumenata na Internetu, koji se osim toga pojavljuju u raznim formatima.

Cilj ovog rada je prezentirati latentne modele za zadatak pronalaska najrelevantnijih dokumenata uz dani upit i poseban naglasak je dan na njihove matematičke izvode. Osnovni model, latentna semantička analiza (LSA) opisan je u drugom poglavlju. Prethodno je bilo potrebno prikazati reprezentaciju kolekcije dokumenata vektorskim prostorom, što smo učinili prateći knjigu [14]. U trećem poglavlju prezentiramo probabilističku latentnu semantičku analizu (pLSA), s kojom rješavamo nedostatke prve metode. U četvrtom poglavlju bavimo se latentnom Dirichletovom alokacijom (LDA), koja služi kao poboljšanje pLSA. Na kraju, u petom poglavlju primjenjujemo opisana razmatranja na mali uzorak članaka s Wikipedije.

Koristim priliku kako bih se zahvalila mentoru, izv. prof. dr. sc. Saši Singeru, na savjetima, temeljitosti i strpljenju tokom pisanja ovog rada.

Poglavlje 1

Model vektorskog prostora

1.1 Reprezentacija dokumenata

Svatko od nas se gotovo svakodnevno koristi nekom tražilicom na Internetu, što znači da gotovo svakodnevno smišljamo relevantan upit (eng. query) za neki traženi pojam. Ljudi, koristeći znanje i iskustvo, to čine rutinski, dok, za računala, pronalazak nekoliko relevantnih stranica za dotični upit nije trivijalan zadatak. Računalo iz nekog skupa dokumenata mora vratiti listu onih koji su u najvećoj vezi s danim upitom. Dakle, trebamo izračunati koliko proizvoljan dokument odgovara danom upitu.

Prirodno je pridružiti veću težinu dokumentima u kojima se riječi iz upita pojavljuju više puta. Pretpostavimo da je upit skup riječi. Ako svakom pojmu koji se javlja u dokumentu pridružimo težinu, koja ovisi o broju pojavljivanja tog pojma u dokumentu, onda za svaku riječi koja se pojavljuje u upitu možemo izračunati koliko je ona povezana s dokumentom, tako da pogledamo njezinu težinu za dani dokument. Nadalje, možemo zbrojiti tako dobivene težine za svaku riječ koja se pojavljuje u upitu i na taj način dobiti vrijednost koja odgovara tome koliko je dokument relevantan za dani upit.

Označimo s $tf_{t,d}$ frekvenciju pojma t u dokumentu d (eng. term frequency). Sljedeći primjer ilustrira na kakve poteškoće nailazimo ako za težinu pojma u dokumentu koristimo samo njegovu frekvenciju u dokumentu. Zamislimo da imamo skup članka koji govore o proizvodima organskog podrijetla. U mnogo takvih članka često će se pojavljivati riječi "organski" i "prirodno", te zato te riječi neće činiti razliku između članka iz te kolekcije. Ako jedan članak govori o kremi organskog podrijetla, a drugi o žitaricama organskog podrijetla, vidimo da su riječi koje razlikuju te članke vezane uz riječi "krema", "žitarice" i sl., dok se riječ "organski" pojavljuje u jednom i drugom, i zato ne sudjeluje u njihovom razlučivanju. Dakle, korištenjem frekvencije pojma u dokumentu smatramo da svi pojmovi u dokumentu jednako sudjeluju u određivanju tematike dokumenta, iako pojmovi koji se učestalo pojavljuju u svim dokumentima nemaju nikakvu moć određivanja.

pojam	cf	df
try	10 422	8 760
insurance	10 440	3 997

Tablica 1.1: Reuters kolekcija

Dakle, moramo reducirati $tf_{t,d}$ težinu učestalih pojmova i to najlakše postižemo tako da težinu podijelimo s faktorom koji raste s frekvencijom pojma u cijeloj kolekciji. Za broj pojavljivanja pojma u cijeloj kolekciji koristimo naziv kolekcijska frekvencija pojma (eng. collection frequency) i označavamo je s cf_t. Sada je problem što u kolekciji može biti malo dokumenata koji spominju dani pojam, ali ako su oni opsežni, kolekcijska frekvencija će biti velika. Zato umjesto kolekcijske frekvencije koristimo dokumentnu frekvenciju pojma (eng. document frequency), u oznaci df_t, i definiramo je kao broj dokumenata u kojim se pojavljuje pojam t.

U Tablici 1.1 uzeli smo dva pojma iz Reuters kolekcije[1] kako bismo vidjeli da riječi koje imaju gotovo jednaku cf_t vrijednost ne moraju imati sličnu vrijednost df_t. Želimo da dokumenti koji sadrže riječ "insurance" dobiju veću važnost za upit koji sadrži riječ "insurance", nego dokumenti koji sadrže riječ "try", kada upit sadrži riječ "try". Zato definiramo inverznu dokumentnu frekvenciju pojma t (eng. inverse document frequency) kao

$$idf_t = \log \frac{n}{df_t},$$

gdje je n broj dokumenata u kolekciji. Inverzna dokumentna frekvencija služi kao spomenti faktor za reduciranje frekvencije $tf_{t,d}$. Ona je velika za pojmove koji se rijetko pojavljuju u kolekciji i obrnuto. Napomenimo da baza logaritma nije bitna za rangiranje, te služi samo kako bi se ublažio utjecaj faktora idf_t.

Sada za svaki pojam t možemo definirati njegovu težinu u dokumentu d kao

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t .$$

Napomena 1.1.1. *Vrijedi*

(1) Težina tf-idf je najveća kad se pojam t pojavljuje u jako malo dokumenata.

(2) Težina tf-idf je manja kad se pojam t pojavljuje par puta u dokumentu, ili kad se pojavljuje u puno dokumenata.

(3) Težina tf-idf je najmanja kad se pojavljuje u svim dokumentima.

[1] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

Konačno, definiramo vrijednost koja mjeri koliko je neki dokument d bitan za dani upit q, kao sumu *tf-idf* težina svih riječi iz upita q za dokument d

$$score(q,d) = \sum_{t \in q} tf\text{-}idf_{t,d} \,. \tag{1.1}$$

Nakon što uvedemo mjeru sličnosti u vektorskom prostoru koji reprezentira kolekciju, važnost dokumenta za dani upit moći ćemo mjeriti tako da kažemo koliko su vektori dokumenta i upita u novonastalom prostoru slični.

1.2 Uspoređivanje dokumenata

Označimo s $\mathcal{T} = \{t_1, ..., t_m\}$ rječnik, na primjer, to može biti unija svih pojmova koji se javljaju u svim dokumentima. Neka je $\mathcal{D} = \{d_1, ..., d_n\}$ skup dokumenata, na primjer, ako pretražujemo Wikipediju, onda je to skup članaka. Tada svakom dokumentu d_i , $i = 1, ..., n$, pridružujemo vektor

$$v_i = \begin{pmatrix} w_{1i} \\ \vdots \\ w_{mi} \end{pmatrix},$$

gdje je $w_{ji} = tf\text{-}idf_{t_j,d_i}$, $j = 1, ..., m$.

Sada možemo konstruirati pojmovno-dokumentnu matricu (eng. term-document matrix) dimenzije $m \times n$ na sljedeći način

$$V = (v_1 \cdots v_n) = \begin{bmatrix} w_{11} & \dots & w_{1j} & \dots & w_{1n} \\ \vdots & & \vdots & & \vdots \\ w_{i1} & \dots & w_{ij} & \dots & w_{in} \\ \vdots & & \vdots & & \vdots \\ w_{m1} & \dots & w_{mj} & \dots & w_{mn} \end{bmatrix}. \tag{1.2}$$

Model u kojem dokument reprezentiramo kao vektor u odgovarajućem vektorskom prostoru zovemo model vektorskog prostora. U literaturi se predstavljanje dokumenta u modelu vektorskog prostora često zove predstavljanje "vrećom riječi" (eng. bag of words represetation).

Reprezentacija dokumenta vektorom omogućuje da dokumente uspoređujemo standardnim mjerama sličnosti (eng. similarity measure) u vektorskom prostoru. Jedna od njih je kosinusna sličnost (eng. cosine similarity) definirana kao

$$sim_{\cos}(d_1, d_2) = \cos(\theta) = \frac{v(d_1) \cdot v(d_2)}{\|v(d_1)\| \cdot \|v(d_2)\|},$$

gdje je $v(d_i)$ vektor dobiven *tf-idf* shemom dokumenta d_i, $i = 1, 2$. Ona mjeri kosinus kuta između vektora $v(d_1)$ i $v(d_2)$. Budući da su elementi vektora dokumenata *tf-idf* vrijednosti, koje su nenegativne, slijedi da je $sim_{\cos}(d_1, d_2) \in [0, 1]$. Što je dobivena vrijednost bliža 1, to su dokumenti d_1 i d_2 sličniji, a što je bliža 0, to su različitiji.

Korisnikov upit možemo shvatiti kao dokument i *tf-idf* shemom reprezentirati vektorom, kojeg ćemo, radi jednostavnosti, opet označiti s q. Zatim možemo rangirati dokumente po važnosti za upit tako da ih rangiramo po tome koliko su slični vektori koji reprezentiraju dokument i upit. Dakle, umjesto vrijednosti dane formulom (1.1), dalje koristimo

$$score(q,d) = sim_{\cos}(q, d)\,.$$

Iako je vektorska reprezentacija jako pogodna za korištenje matematičkih metoda i raznoliku primjenu, postoje mnogi problemi ovakvog pristupa. Metoda se ne može primijeniti na individualne riječi, jer će one u novonastalom prostoru biti ortogonalne. Dalje, za metodu veliki problem predstavljaju sinonimi, riječi jednakog značenja, i višeznačnice, riječi koje imaju više značenja. Na primjer, ako je korisnik u upitu iskoristio riječ ”kompjuter”, a neki dokument učestalo, umjesto ”kompjuter”, koristi riječ ”računalo”, taj dokument neće imati važnost za taj upit koju potencijalno ima. Dakle, radi sinonima metoda će imati loš odziv (eng. recall), omjer relevantnih dokumenata koji su vraćeni, naspram ukupnog broja relevantnih dokumenata u kolekciji. S druge strane, ako korisnik pretražuje o mišu kao računalnom pomagalu, metoda će vratiti i dokumente koji govore o mišu kao životinji. To će rezultirati lošom preciznošću (eng. precision), udjelom relevatnih dokumenata u svim dokumentima koji su vraćeni. Osim toga, veliki problem predstavlja velika dimenzija i rang dobivene pojmovno-dokumentne matrice. U daljnjem radu želimo metode koje se bolje nose s tim poteškoćama.

Poglavlje 2

Dekompozicija matrice i latentna semantička analiza

Kolekcija dokumenata razumne veličine vrlo vjerojatno će rezultirati pojmovno-dokumentnom matricom koja ima desetak tisuća redaka i stupaca, te rang veličine par stotina. Daljnji rad s takvom matricom postaje vremenski i memorijski zahtjevan. Osim toga, želimo metodu koja korisniku vraća relevantne dokumente na temelju toga koliko su tematski slični, a ne koliko zajedničkih riječi dijele. Latentna semantička analiza (LSA) ili latentno semantičko indeksiranje[1] (LSI) uzima u obzir, ne samo pojavljivanje riječi u dokumentu, već i s kojim drugim riječima se taj pojam najčešće pojavljuje u cijeloj kolekciji. Pojam dijeli sličan kontekst s riječima s kojim se učestalo pojavljuje u kolekciji i taj kontekst je latentna (skrivena) struktura koja je uzrokovana velikim izborom riječi za izražavanje. LSA pomoću singularne dekompozicije matrice (SVD) omogućava reprezentaciju dokumenata i upita u prostoru niže dimenzije, u kojem su dokumenti sličnog konteksta bliži jedan drugome i na takav način smanjuje šum dobiven zbog jezične varijabilnosti.

2.1 Dekompozcije matrice

Teorem 2.1.1. *Neka je $A \in \mathbb{C}^{m\times m}$ hermitska matrica. Tada postoji unitarna matrica $U \in \mathbb{C}^{m\times m}$ i realna, dijagonalna matrica $\Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_m) \in \mathbb{R}^{m\times m}$ takva da je*

$$A = U\Lambda U^{-1}\,. \tag{2.1}$$

U tom slučaju, dijagonalni elementi matrice Λ su svojstvene vrijednosti matrice A, a stupci matrice U su svojstveni vektori matrice A. Rastav (2.1) zovemo spektralna dekompozicija matrice A.

[1] Naziv „latentno semantičko indeksiranje“ se koristi kad se LSA primjenjuje na probleme u pretraživanju informacija (eng. information retrieval).

Međutim, ako je m broj riječi u rječniku i n broj dokumenata, $m \neq n$, *tf-idf* shemom dobijemo pravokutnu $m \times n$ pojmovno-dokumentnu matricu koja, očito, ne mora biti simetrična. Dakle, za takvu matricu navedena dekompozicija neće biti korisna. Zato navodimo još jednu dekompoziciju matrice koja je primjenjiva na proizvoljnu, pravokutnu, kompleksnu matricu i, radi toga, jako korisna u matematici i računarstvu.

Teorem 2.1.2. *Neka je $A \in \mathbb{C}^{m\times n}$ ranga r. Tada postoje unitarne matrice $U \in \mathbb{C}^{m\times m}$ i $V \in \mathbb{C}^{n\times n}$ takve da je*

$$U^*AV = \begin{array}{c} \\ r \\ m-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} \Sigma_+ & 0 \\ 0 & 0 \end{pmatrix} \end{array} = \Sigma,$$

odnosno

$$A = U\Sigma V^*, \tag{2.2}$$

gdje je $\Sigma_+ = \mathrm{diag}(\sigma_1, ..., \sigma_r)$, pri čemu su $\sigma_1, \ldots, \sigma_r$ realni i vrijedi $\sigma_1 \geq \cdots \geq \sigma_r > 0$. Skalari $\sigma_1, \ldots, \sigma_{\min\{m,n\}}$ zovu se singularne vrijednosti, a stupci matrica U i V, lijevi i desni singularni vektori matrice A. Rastav matrice A na $A = U\Sigma V^*$ nazivamo singularna dekopozicija matrice A ili kraće, SVD matrice A.

Dokaz. Pogledati u [8]. □

Uzimajući u obzir da je dobivena pojmovno-dokumenta matrica realna, dalje radimo na $\mathbb{R}^{m\times n}$. Tada je hermitska matrica simetrična, unitarna je ortogonalna, a umjesto kompleksnog konjugiranja i transponiranja, potrebno je samo transponiranje.

Ilustraciju singularne dekompozicije matrice možemo vidjeti na Slici (2.1).

Napomena 2.1.3. *Ako jednadžbu* (2.2) *pomnožimo zdesna s matricom A^T dobijemo*

$$AA^T = U\Sigma V^T(U\Sigma V^T)^T = U\Sigma V^T(V^T)^T\Sigma^T U^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma^2 U^T. \tag{2.3}$$

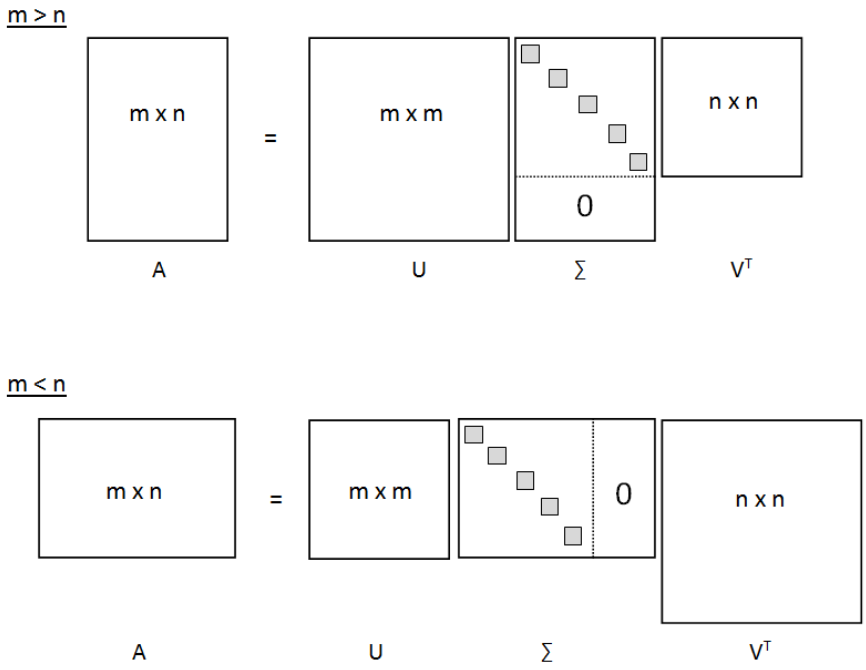
Analogno, množenjem slijeva slijedi

$$A^T A = V\Sigma^2 V^T. \tag{2.4}$$

Dakle, $\sigma_1^2, ..., \sigma_r^2$ su svojstvene vrijednosti matrice AA^T i A^TA različite od nule, a prvih r stupaca matrica U, odnosno, V su pripadni svojstveni vektori.

Na lijevoj strani jednadžbe (2.3) nalazi se kvadratna, simetrična, realna matrica u kojoj (i, j)-element predstavlja mjeru preklapanja i-tog i j-tog pojma, dok s desne strane imamo njezinu spektralnu dekompoziciju. Preciznije, neka je pojmovno-dokumentna matrica A dobivena *tf-idf* shemom, oblika (1.2). Njezin (i, j)-element dan je formulom

$$(AA^T)_{i,j} = \sum_{k=1}^{\mathrm{n}} w_{ik} \cdot w_{jk},$$



Slika 2.1: Singularna dekompozicija matrice A

gdje je n i dalje broj dokumenata. Da matrica AA^T predstavlja mjeru preklapanja pojmova, možemo jasnije vidjeti ako, umjesto *tf-idf* sheme, pretpostavimo da je pojmovno-dokumentna matrica A nastala tako da je težina pojma u dokumentu indikator da li se pojam nalazi u dokumentu ili ne. Tada je (i, j)-element matrice AA^T jednak broju dokumenata u kojem se nalaze i i-ti i j-ti element.

Ako matrica AA^T predstavlja mjeru preklapanja pojmova, koristeći vezu matrice AA^T i singularne dekompozicije matrice A iz Napomene 2.1.3, jasno je da korištenjem singularne dekompozicije matrice uzimamo u obzir, ne samo individualnu riječ, nego i s kojim riječima se ona pojavljuje, što zapravo predstavlja kontekst riječi. Međutim, LSA pretpostavlja da riječ ima jedan kontekst i upravo zato se ne nosi dobro s višeznačnicama.

Aproksimacija matricom nižeg ranga

Pojmovno-dokumentnu matricu želimo aproksimirati matricom nižeg ranga, koja će obuhvaćati najvažnije veze među pojmovima, ali kojom ćemo, zbog manjeg ranga, ukloniti šum (eng. noise) dobiven radi efekta različitog izražavanja različitih ljudi.

Kako bismo izmjerili koliko jedna matrica odstupa od druge, želimo poopćiti Euklidsku

udaljenost na matrice. Definiramo Frobeniusovu normu $m \times n$ matrice na sljedeći način

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2},$$

ili, ekvivalentno, koristeći trag matrice

$$\|A\|_F = \sqrt{\mathrm{tr}(X^T X)} = \sqrt{\mathrm{tr}(X X^T)}\,. \tag{2.5}$$

Iz (2.5) slijedi da za ortogonalnu matricu O vrijedi

$$\|OA\|_F = \|A\|_F\,.$$

Dakle, Frobeniusova norma je unitarno invarijantna, pa možemo izbrisati unitarne matrice slijeva i zdesna u njezinom SVD rastavu

$$\|A\|_F = \|U\Sigma V^T\|_F = \|\Sigma V^T\|_F = \|\Sigma\|_F = \sqrt{\sum_{i=1}^{r} \sigma_j^2},$$

gdje je r rang matrice A.

Sljedeća dva teorema nam daju željenu aproksimaciju.

Teorem 2.1.4. *Neka je SVD $m \times n$ matrice A ranga r, $r \leq \min(m,n)$, dana s* (2.2) *i pretpostavimo da je*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_{\min(m,n)} = 0.$$

Neka su $\mathcal{R}(A)$ i $\mathcal{N}(A)$, redom, slika i jezgra matrice A.

(1) Za jezgru i sliku vrijedi

$$\mathcal{N}(A) = \mathrm{span}\{v_{r+1}, \ldots, v_n\}\,,$$
$$\mathcal{R}(A) = \mathrm{span}\{u_1, \ldots, u_r\}\,,$$

gdje je $U = (u_1 \cdots u_m)$ i $V = (v_1 \cdots v_n)$.

(2) Matricu A možemo napisati kao zbroj matrica ranga 1*,*

$$A = \sum_{i=1}^{r} u_i \cdot \sigma_i \cdot v_i^T\,.$$

(3) Za Frobeniusovu i spektralnu normu matrice A vrijedi

$$\|A\|_F^2 = \sigma_1^2 + \cdots + \sigma_r^2 ,$$
$$\|A\|_2^2 = \sigma_1 .$$

Dokaz. Pogledati u [8]. □

Teorem 2.1.5. (Eckart i Young) *Neka je SVD $m \times n$ matrice A ranga r dana s* (2.2). *Za bilo koji $k \in \{1, \ldots, r\}$ definiramo matricu*

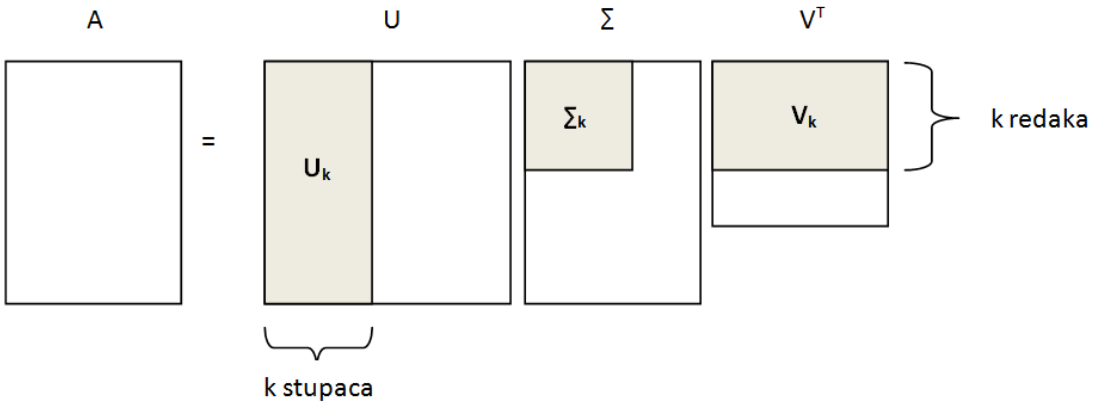
$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T .$$

Vrijedi

$$\min_{rang(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_{\min(m,n)}^2 .$$

Dokaz. Pogledati u [8]. □

Kako dobijemo matricu A_k ilustrirano je na Slici 2.2.



Slika 2.2: Aproksimacija matrice A

Dakle, A_k je najbolja aproksimacija matrice A matricom ranga k. Osim toga, Frobeniusovu normu možemo zamijeniti bilo kojom unitarnom invarijantnom normom [15]. Iz toga slijedi

$$\min_{rang(B)=k} \|A - B\|_2^2 = \|A - A_k\|_2^2 = \sigma_{k+1}^2 .$$

Primijetimo još da smo, aproksimirajući matricu A matricom A_k, zanemarili utjecaj svojstvenih vektora koji su vezani uz male svojstvene vrijednosti. Za pojmovno-dokumentnu matricu A to znači da smo zanemarili smjerove u novonastalom prostoru koji imaju malen utjecaj na cjelokupnu kolekciju, odnosno, obaziremo se samo na osnovne teme koje se pojavljuju u kolekciji.

Preostaje uočiti da, iako znamo najbolju aproksimaciju matrice A matricom ranga k, ne znamo kojim rangom k ćemo obuhvatiti najvažnije informacije i istovremeno maksimalno smanjiti utjecaj šuma, odnosno, ne znamo apriori koji k je optimalan. Koji rang daje optimalnu reprezentaciju proizvoljne baze je otvoreno pitanje i najčešće se nalazi eksperimentalno [2]. Međutim, za jako velike baze, k se najčešće bira između 100 i 300 [13].

Izračunavanje singularne dekompozicije matrice

Pojmovno-dokumentna matrica je najčešće rijetka matrica (eng. sparse matrix), odnosno, većina njezinih elemenata je jednaka nuli. Osim toga, ne-nul elementi se ne pojavljuju u nikakvoj pravilnoj strukturi koja bi se mogla iskoristiti za efikasnije računanje s takvom matricom. Kako bismo izračunali singularnu dekompoziciju takve matrice, potrebno je spremiti samo ne-nul elemente. U tu svrhu su smišljeni posebni formati za spremanje rijetkih matrica, poput Harwell – Boeing formata ili njegove fleksibilnije inačice – Rutherford – Boeing [6].

Metode razvijene za računanje singularne dekopozicije rijetke matrice koriste iterativne metode poput Arnoldijeve [12] , Lanczosove [11], iteracije potprostora [17] i minimizacije traga [18], koje se mogu prilagoditi formatu rijetkih matrica. Implementacije spomenutih metoda su dostupne na `www.netlib.org`.

2.2 Latentna semantička analiza

Sada bismo htjeli iskoristiti aproksimaciju pojmovno-dokumentne matrice matricom nižeg ranga, kako bismo napravili novu reprezentaciju kolekcije dokumenata. Kako bismo usporedili korsnikov upit s dokumentima u novonastaloj reprezentaciji, i upit prikazujemo vektorom u novom prostoru. Dalje koristimo kosinusnu sličnost i rangiramo dokumente po tome koliko su relevantni za dani upit. Ovaj proces zovemo latentna semantička analiza.

Kako bismo implementirali latentnu semantičku analizu prvo je potrebno napraviti pojmovno-dokumentnu matricu kao u Poglavlju 1. Neka je m broj riječi u rječniku i n broj dokumenata, uz prirodnu pretpostavku $m > n$. Neka je $d_i \in \mathbb{R}^m$, $i = 1, \ldots, n$, *tf-idf* reprezentacija i-tog dokumenta. Tada pojmovno-dokumentna matrica A ima sljedeći oblik

$$A = (d_1 \cdots d_n) .$$

Također, i korisnikov upit moramo reprezentirati *tf-idf* shemom vektorom $q \in \mathbb{R}^m$.

Sada bismo htjeli izračunati singularnu dekompozciju matrice $A = U\Sigma V^T$ iz razloga pojašnjenih u prethodnoj sekciji, te iskoristiti Teorem 2.1.5 za izračun aproksimacije A_k matrice A. Zapišimo singularnu dekompoziciju matrice A matrično

$$(d_1 \cdots d_n) = (u_1 \cdots u_m)\Sigma \underbrace{(\tilde{d}_1 \cdots \tilde{d}_n)}_{V^T} .$$

Pretpostavimo da smo konstruirali matrice U_k, Σ_k i V_k kao u Teoremu 2.1.5. Prikaz dokumenta d_i u prostoru razapetom s $\{u_1, \ldots, u_k\}$ glasi

$$d_i = \sum_{j=1}^{k} \underbrace{\sigma_i \cdot (\tilde{d}_i)_j}_{\text{koeficijent}} \cdot \underbrace{u_j}_{\text{vektor baze}} .$$

Singularna vrijednost σ_i predstavalja važnost i-tog vektora baze, u_i. Ako uzmemo u obzir da smo dobili prostor koji predstavlja koncepte, onda σ_i govori koliko je i-ti koncept, kojeg predstavlja vektor u_i, važan u kolekciji. Na primjer, ako imamo kolekciju dokumenata koji govore o organskoj hrani i kozmetici, dva važna vektora baze predstavljaju pojmove "hrana" i "kozmetika". Stupci matrice $\Sigma_k V_k^T$ predstavljaju koeficijente dokumenata u novonastalom prostoru niže dimenzije, tzv. latentnom prostoru.

Kako bismo upit mogli uspoređivati s dokumentima, moramo i njega prikazati u novonastalom latentnom prostoru. Dakle, tražimo koordinate tog vektora u prostoru razapetom stupcima matrice U_k. Ako s $\hat{q}$ označimo projekciju vektora q na taj prostor, onda znamo da vrijedi

$$\hat{q} = U_k U_k^T q .$$

Dakle, vektor $U_k^T q$ predstavlja koeficijente vektora q u latentnom prostoru (razapetom stupcima matrice U_k). Koeficijenti ostalih dokumenata u latentnom prostoru dani su stupcima matrice $\Sigma_k V_k^T$. Kako bismo izbjegli eksplicitno računanje matrice $\Sigma_k V_k^T$ definiramo

$$\tilde{q} = \Sigma_k^{-1} U_k^T q$$

i uspoređujemo upit q s dokumentima tako da uspoređujemo $\tilde{q}$ sa stupcima matrice V_k^T (ili, ekvivalentno, $\tilde{q}^T$ s retcima matrice V_k). Računamo

$$\cos\theta_j = \frac{\tilde{d}_j \cdot \tilde{q}}{\|\tilde{d}_j\|_2 \cdot \|\tilde{q}\|_2} ,$$

za svaki $j = 1, \ldots, n$. Vidimo da se kosinusi mogu izračunati bez eksplicitnog računanja matrice A_k. Norme $\|\tilde{d}_j\|_2$ računaju se samo jednom, a zatim koriste za sve upite. Budući da je vektor q jako rijedak, trošak izračuna $\Sigma_k^{-1} U_k^T q$ nije velik. Dakle, nikad nije potrebno

izračunati potpuni SVD matrice A, dovoljno je samo izračunati k najvećih singularnih vrijednosti i pripadne singularne vektore koji čine U_k, Σ_k i V_k. Konačno, rangiramo kosinuse od najvećeg prema najmanjem i vraćamo prvih par dokumenata koji odgovaraju najvećim kosinusima.

2.3 Dinamička kolekcija

Prirodna pretpostavka je da se kolekcija dokumenata s vremenom mijenja, odnosno, da se u nju dodaju ili iz nje uklanjaju pojmovi i dokumenti. Na primjer, ako Wikipedia predstavlja kolekciju dokumenata (članaka), znamo da se dnevno velike količine novih dokumenata dodaju, a s novim dokumentima dolazi i novi rječnik. Međutim, pojmovno-dokumentna matrica i njezina aproksimacija već su konstruirane za staru kolekciju, na što je utrošena znatna količina vremena. Zato je dobro pitanje mogu li se one jednostavno ažurirati kod promjene kolekcije.

Zamislimo da želimo dodati nove dokumente i pojmove. Jedna od mogućnosti je da ponovno izračunamo pojmovno-dokumentnu matricu i njezinu aproksimaciju za novu kolekciju, ali očito je da je to za veliku kolekciju vremenski i memorijski preskupo. Zato su se nametnule dvije najčešće alternative: dodavanje (eng. folding-in) dokumenata u postojeći latentni prostor i ažuriranje singularne dekompozicije (eng. SVD-updating).

Ideja umetanja je da svaki novi dokument, poput upita, prikažemo u latentnom prostoru vektorom

$$\tilde{d} = d^T U_k \Sigma_k^{-1}$$

i dodamo ga kao novi redak matrice V_k. Analogno, novi pojam t prikažemo vektorom

$$\tilde{t} = t V_k \Sigma_k^{-1}$$

i dodamo kao novi stupac matrice U_k. Ovaj pristup je računarski jednostavan i nezahtjevan, međutim može rezultirati nepreciznom reprezentacijom. Novodobivena matrica V_k više nema ortonormirane stupce, te ako želimo dodati dokument koji je skoro ortogonalan na vektore baze, informacija o ortogonalnosti će se izgubiti prilikom projiciranja. Uočimo da ovako uspoređujemo upit s novim dokumentima samo na temelju toga što stari dokumenti i pojmovi kažu gdje se novi dokumenti nalaze u latentnom prostoru. Dakle, nikad ne koristimo informacije dobivene iz novih dokumenata. Nakon što dodamo puno dokumenata na ovaj način, ipak ćemo morati ponovno izračunati SVD.

Opišimo sada ažuriranje singularne dekompozicije prateći razmatranja u [1] i [2].

Ažuriranje singularne dekompozicije

Pretpostavimo da želimo dodati t novih pojmova u rječnik. Označimo s T *tf-idf* reprezentaciju tih t pojmova u odnosu na n postojećih dokumenata. Matricu T, tipa $t \times n$, dodajemo

retcima matrice A_k, tipa $m \times n$, i kao rezultat dobijemo $(m+t) \times n$ matricu

$$\tilde{A}_k = \begin{pmatrix} A_k \\ T \end{pmatrix}.$$

Sada želimo izračunati k najvećih singularnih vrijednosti i pripadne singularne vektore matrice $\tilde{A}_k$. Uočimo da je razlika između ovog načina i računanja potpuno nove singularne dekompozicije u tome da smo matricu A zamijenili matricom A_k. Izvedimo dekompoziciju matrice $\tilde{A}_k$,

$$\begin{aligned}
\tilde{A}_k &= \begin{pmatrix} A_k \\ T \end{pmatrix} = \begin{pmatrix} U_k \Sigma_k V_k^T \\ T \end{pmatrix} = \begin{pmatrix} U_k \Sigma_k V_k^T \\ T V_k V_k^T + T - T V_k V_k^T \end{pmatrix} \\
&= \begin{pmatrix} U_k & 0 \\ 0 & I_t \end{pmatrix} \left[\begin{pmatrix} \Sigma_k \\ T V_k \end{pmatrix} V_k^T + \begin{pmatrix} 0 \\ T(I_n - V_k V_k^T) \end{pmatrix} \right] \\
&= \begin{pmatrix} U_k & 0 \\ 0 & I_t \end{pmatrix} \begin{pmatrix} \Sigma_k & 0 \\ T V_k & I_t \end{pmatrix} \begin{pmatrix} V_k^T \\ T(I_n - V_k V_k^T) \end{pmatrix} \\
&= \begin{pmatrix} U_k & 0 \\ 0 & I_t \end{pmatrix} \begin{pmatrix} \Sigma_k & 0 \\ T V_k & I_t \end{pmatrix} \begin{pmatrix} V_k & (I_n - V_k V_k^T) T^T \end{pmatrix}^T .
\end{aligned} \tag{2.6}$$

Lijeva matrica u zadnjoj nejednakosti je tipa $(m+t) \times (k+t)$, srednja tipa $(k+t) \times (k+t)$ i desna tipa $(k+t) \times n$. Matrice U_k i V_k imaju ortonormirane stupce, pa ih ima i prva matrica u posljednjem redu (2.6). Međutim, posljednja matrica u tom redu nema ortonormirane stupce, jer ih nema ni matrica $\tilde{V}_k = (I - V_k V_k^T) T^T$. Stupci matrice $\tilde{V}_k$ pripadaju ortogonalnom komplementu prostora stupaca matrice V_k (Slika 2.3). Dakle, vektore novih pojmova (retke matrice T) smo projicirali na ortogonalni komplement prostora redaka matrice V_k. Ortogonalnost popravljamo tako da radimo QR faktorizaciju $n \times t$ matrice $\tilde{V}_k$

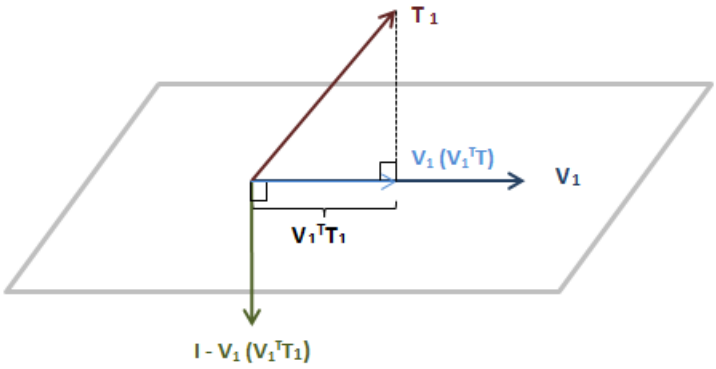
$$\tilde{V}_k \Pi = \hat{V}_k R = \hat{V}_k \begin{pmatrix} R_0 \\ 0 \end{pmatrix},$$

gdje je Π permutacijska matrica reda t, $\hat{V}_k$ matrica reda n s ortonormiranim stupcima, a R_0 gornjetrokutasta matrica reda t, uz prirodnu pretpostavku da je $\min\{n, t\} = t$. Slijedi

$$\begin{pmatrix} V_k & (I_n - V_k V_k^T) T^T \end{pmatrix} = \begin{pmatrix} V_k & \hat{V}_k R \Pi^T \end{pmatrix}.$$

Dakle, jednadžba (2.6) onda glasi

$$\begin{aligned}
\tilde{A}_k &= \begin{pmatrix} U_k & 0 \\ 0 & I_t \end{pmatrix} \begin{pmatrix} \Sigma_k & 0 \\ T V_k & I_t \end{pmatrix} \begin{pmatrix} V_k & \hat{V}_k R \Pi^T \end{pmatrix}^T \\
&= \begin{pmatrix} U_k & 0 \\ 0 & I_t \end{pmatrix} \begin{pmatrix} \Sigma_k & 0 \\ T V_k & \Pi R^T \end{pmatrix} \begin{pmatrix} V_k^T \\ \hat{V}_k^T \end{pmatrix}.
\end{aligned}$$



Slika 2.3: Stupci matrice $\tilde{V}_k$ čine ortogonalni komplement prostora stupaca matrice V_k

Pretpostavimo da SVD $(k+t)\times(k+n)$ matrice u sredini prethodnog retka glasi

$$\begin{pmatrix} \Sigma_k & 0 \\ TV_k & \Pi R^T \end{pmatrix} = \begin{pmatrix} P_k & P_k^{\perp} \end{pmatrix} \begin{pmatrix} \hat{\Sigma}_k & 0 \\ 0 & \hat{\Sigma}_t \end{pmatrix} \begin{pmatrix} Q_k & Q_k^{\perp} \end{pmatrix}^T ,$$

gdje je P_k matrica tipa $(k+t)\times k$, $\hat{\Sigma}_k$ matrica tipa $k\times k$ i Q_k matrica tipa $(k+t)\times k$. Sada slijedi da je najbolja aproksimacija matrice $\tilde{A}_k$ matricom ranga k

$$\tilde{A}_k = \underbrace{\begin{pmatrix} U_k & 0 \\ 0 & I_t \end{pmatrix} P_k}_{U_{\tilde{A}_k}} \underbrace{\hat{\Sigma}_k}_{\Sigma_{\tilde{A}_k}} \underbrace{\begin{pmatrix} (V_k & \hat{V}_k) & Q_k \end{pmatrix}^T}_{V_{\tilde{A}_k}} .$$

Ako želimo dodati dokumente u postojeću kolekciju onda analogan postupak radimo za matricu

$$\tilde{A}_k = \begin{pmatrix} A_k & D \end{pmatrix} ,$$

gdje je D pojmovno-dokumentna matrica kolekcije novih dokumenata.

Na kraju, pretpostavimo da želimo samo ažurirati težine u aproksimaciji $A_k = U_k\Sigma_k V_k^T$ pojmovno-dokumentne matrice A. Želimo izračunati singularnu dekompoziciju matrice

$$\tilde{A}_k = A_k + YZ^T ,$$

u kojoj Y predstavlja indikatorsku matricu težina koje želimo mijenjati, a Z mjeri razliku novih i starih težina.

Slično kao prije, radimo QR faktorizaciju matrica $(I - U_k U_k^T)Y$ i $(I - V_k V_k^T)Z$

$$(I - U_k U_k^T)Y\Pi_Y = Q_Y R_Y\,,$$
$$(I - V_k V_k^T)Z\Pi_Z = Q_Z R_Z\,.$$

Slijedi

$$\tilde{A}_k = \begin{pmatrix} U_k & Q_Y \end{pmatrix} \Bigg[\underbrace{\begin{pmatrix} \Sigma_k & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} U_k^T Y \\ R_Y \Pi_Y^T \end{pmatrix} \begin{pmatrix} Z^T V_k & \Pi_Z R_Z^T \end{pmatrix}}_{\hat{A}_k} \Bigg] \begin{pmatrix} V_k^T \\ Q_Z^T \end{pmatrix}.$$

Napravimo SVD dekompoziciju matrice $\hat{A}_k$

$$\hat{A}_k = \begin{pmatrix} \hat{U}_k & \hat{U}_k^\perp \end{pmatrix} \begin{pmatrix} \hat{\Sigma}_k & 0 \\ 0 & \hat{\Sigma}_j \end{pmatrix} \begin{pmatrix} \hat{V}_k & \hat{V}_k^\perp \end{pmatrix}^T.$$

Konačno, najbolja aproksimacija matrice $\tilde{A}_k$ matricom ranga k je

$$\bar{A}_k = \underbrace{\left(\begin{pmatrix} U_k & Q_Y \end{pmatrix} \hat{U}_k \right)}_{U_{\bar{A}_k}} \underbrace{\hat{\Sigma}_k}_{\Sigma_{\bar{A}_k}} \underbrace{\left(\begin{pmatrix} V_k & Q_Z \end{pmatrix} \hat{V}_k \right)^T}_{V_{\bar{A}_k}}.$$

Poglavlje 3

Probabilistička latentna semantička analiza

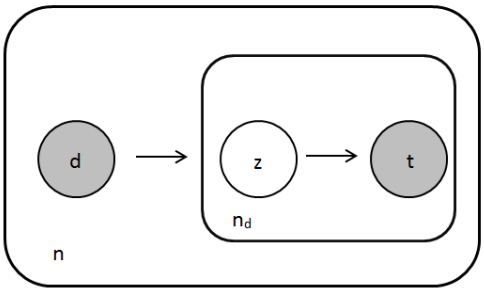
Kao što smo vidjeli u prošlom poglavlju, LSA je metoda koju nije teško implementirati i koristiti, te je upravo zato našla široku primjenu u pretraživanju informacija (eng. information retrieval). Nadalje, s njom smo donekle riješili problem sinonima, ali možemo uočiti da ona još uvijek ima problem s višeznačnicama. Uzrok toga je što je svaki pojam reprezentiran jedinstvenim vektorom u latentnom prostoru. Metoda je jednostavna za shvatiti, ali bi svakako bila intuitivnija kad bi njezini rezultati imali vjerojatnosnu interpretaciju. Osim toga, s obzirom da LSA minimizira razliku aproksimacije i matrice u Frobeniusovoj normi, ona predstavlja problem najmanjih kvadrata čije je rješenje jednako rješenju maksimizacije izglednosti uz pretpostavku da šum uzorka podataka ima Gaussovu distribuciju. Međutim, s obzirom da modeliramo pojavljivanje različitih pojmova u dokumentima, prirodno je pretpostaviti multinomijalnu distribuciju. To možemo postići sljedećom metodom, probabilističkom latentnom semantičkom analizom (pLSA), koja, kao što njezino ime kaže, ima i vjerojatnosnu interpretaciju.

3.1 Izvod modela

Vratimo se na primjer smišljanja najboljeg upita u svrhu pronalaska dokumenta koji govori o informacijama koje nas zanimaju. Osnovni model je model vektorskog prostora, koji ovisi samo o broju pojavljivanja riječi u dokumentu i kroz kolekciju. Međutim, želimo model koji će biti bliži ljudskom načinu razmišljanja. Ako čovjek traži dokument koji je relevantan za neki upit, on neće tražiti riječi koje bi se učestalo mogle pojaviti u dokumentu, već riječi koji su najveći nositelj informacija. Takve riječi tražimo tako da razmislimo koje su osnovne teme ili koncepti dokumenta, a zatim nađemo riječi koje najbolje opisuju te koncepte. Upravo tom idejom su vođeni autori probabilističke latentne semantičke analize.

Neka je $\mathcal{D} = \{d_1, \ldots, d_n\}$ skup dokumenata, $\mathcal{T} = \{t_1, \ldots, t_m\}$ skup pojmova i $\mathcal{Z} = \{z_1, \ldots, z_k\}$ skup latentnih tema (koncepata), gdje broj k mora biti određen apriori. Skup primjera čine svi parovi $(d, t) \in \mathcal{D} \times \mathcal{T}$ u kolekciji, pa je model u potpunosti određen ako izračunamo zajedničke vjerojatnosti $p(d, t)$, za svaki $(d, t) \in \mathcal{D} \times \mathcal{T}$. Uvodimo sljedeće pretpostavke:

- Svaki dokument je skup pojmova, odnosno, zanemarujemo poredak riječi. Kao posljedicu imamo da su svi opaženi primjeri (d, t) nezavisno uzorkovani, odnosno, vrijedi $p(\mathcal{D}, \mathcal{T}) = \prod_{(d,t)} p(d, t)$.
- Pretpostavljamo da su riječi i dokumenti uvjetno nezavisni u odnosu na latentnu varijablu z. To je prirodna pretpostavka, ali se može i pokazati koristeći d-separaciju u grafičkom modelu na Slici 3.1. Dakle, za svaki t i d vrijedi $p(t \mid z, d) = p(t \mid d)$ i $p(t, d \mid z) = p(t \mid z)p(d \mid z)$.



Slika 3.1: Grafički model koji upisuje generativni proces. Broj dokumenata označen je s n, a broj riječi u dokumentu d s n_d. Osjenčani krugovi označavaju opažene varijable, a prazni krugovi latentne varijable.

Za daljnji izvod modela potrebno je definirati kategoričku razdiobu, koja je poopćenje Bernoullijeve razdiobe. Kategorička razdioba je razdioba slučajne varijable X koja opisuje pojavljivanje uspjeha u točno jednoj od k kategorija. Razdioba je dana kao

$$p(X = \boldsymbol{x} \mid \boldsymbol{\mu}) = \prod_{i=1}^{k} \mu_i^{x_i},$$

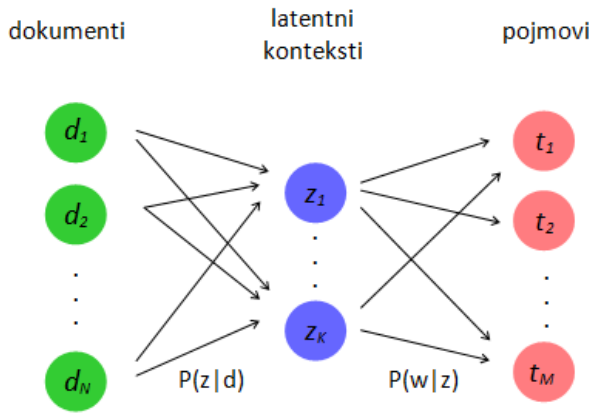
gdje je $\boldsymbol{x} = (x_1, \ldots, x_k)^T$ binarni vektor indikatorskih varijabli i $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)$, $\sum_{i=1}^{k} \mu_i = 1$ i $\mu_i \geq 0$.

Multinomijalna razdioba je razdioba slučajne varijable X definirane kao broj uspjeha u svakoj od k kategorija u n nezavisnih pokusa. Dakle, u slučaju provođenja samo jednog pokusa, multinomijalna i kategorička razdioba su jednake. Zato se često u području prirodne obrade jezika (eng. natural language processing) i pretraživanja informacija (eng. information retrieval), kategorička i multinomijalna razdioba poistovjećuju. Autor originalnoga članka koristi naziv multinomijalna, pa, radi dosljednosti u oznakama, u nastavku pišemo multinomijalna, iako podrazumijevamo kategoričku.

Definiramo generativni model za podudaranje pojmova i dokumenata na sljedeći način:

- odaberi dokument $d \sim \text{Multinomial}(p(d))$,
- odaberi koncept $z \sim \text{Multinomial}(p(z \mid d))$,
- odaberi pojam $t \sim \text{Multinomial}(p(t \mid z))$.

Dakle, vjerojatnosti $p(d)$, $p(z \mid d)$ i $p(t \mid z)$, su parametri modela. Ilustraciju opisanog postupka možemo vidjeti na Slici 3.2. Uočimo da ovako možemo generirati samo već viđene primjere i da zato pLSA nije generativni model u pravom smislu riječi.



Slika 3.2: Ideja pLSA je da svakom dokumentu na temelju cijele kolekcije pridružimo neki koncept, a onda tom konceptu pridružimo riječ za koju je vjerojatno da će ga dobro opisati.

Računamo zajedničku vjerojatnost

$$p(d,t) = \sum_z p(d,t,z) = \sum_z p(z)p(d,t \mid z) = \text{(uvjetna nezavisnost od } t \text{ i } d \text{ u odnosu na } z) =$$
$$= \sum_z \underbrace{p(z)p(d \mid z)}_{p(d)p(z \mid d)} p(t \mid z) = \sum_z p(d)p(z \mid d)p(t \mid z) = p(d) \sum_z p(t \mid z)p(z \mid d)\,.$$

Dobili smo

$$p(d,t) = p(d) \sum_z p(t \mid z)p(z \mid d)\,. \tag{3.1}$$

Dobivena zajednička vjerojatnost je linearna kombinacija k vjerojatnosti $p(t \mid z_i)$ s koeficijentima $p(z_i \mid d)$, $i = 1, \ldots, k$. Dakle, jer smo svaki dokument prikazali kao mješavinu (eng. mixture) unaprijed zadanih latentnih koncepata, slijedi da je pLSA miješani model (eng. mixture model).

Koristeći Bayesovo pravilo pokažimo da je prethodna formula ekvivalentna formuli

$$p(d,t) = \sum_z p(t \mid z)p(d \mid z)p(z)\,. \tag{3.2}$$

Iz zapisa $p(z,d)$ preko uvjetnih vjerojatnosti na dva načina, dobivamo

$$\begin{aligned} p(z \mid d)p(d) &= p(d \mid z)p(z) \\ p(t \mid z)p(z \mid d)p(d) &= p(t \mid z)p(d \mid z)p(z) \\ p(d) \sum_z p(t \mid z)p(z \mid d) &= \sum_z p(t \mid z)p(d \mid z)p(z)\,. \end{aligned}$$

Neka je $\boldsymbol{\theta} = (p(d), p(z \mid d), p(t \mid z))$ vektor parametara modela. Pripadne izglednosti za (3.1) i (3.2) su

$$\mathcal{L}_1(\boldsymbol{\theta} \mid \mathcal{D}, \mathcal{T}) = \log\left(\prod_{d \in \mathcal{D}} \prod_{t \in \mathcal{T}} p(d,t)^{n(d,t)} \right) = \sum_{d \in \mathcal{D}} \sum_{t \in \mathcal{T}} n(d,t) \log\left(p(d) \sum_z p(t \mid z)p(z \mid d) \right),$$
$$\mathcal{L}_2(\boldsymbol{\theta} \mid \mathcal{D}, \mathcal{T}) = \log\left(\prod_{d \in \mathcal{D}} \prod_{t \in \mathcal{T}} p(d,t)^{n(d,t)} \right) = \sum_{d \in \mathcal{D}} \sum_{t \in \mathcal{T}} n(d,t) \log\left(\sum_z p(t \mid z)p(d \mid z)p(z) \right),$$

gdje je $n(d,t)$ broj pojavljivanja pojma t u dokumentu d. Nažalost, maksimizacija ovakve izglednosti nema rješenje u zatvorenoj formi. Problem predstavlja logaritamska funkcija koja djeluje na sumaciju. Rješenje zato moramo pronaći u iterativnim metodama, preciznije u metodi maksimizacije očekivanja.

Općenita formulacija algoritma maksimizacije očekivanja

Algoritam maksimizacije očekivanja (eng. expectation–maximization algorithm) ili EM-algoritam, je iterativni algoritam za maksimizaciju izglednosti koji se često koristi kod modela s latentnim varijablama. Pokažimo kako algoritam radi općenito, pa ćemo ga onda primijeniti na konkretnu mješavinu. Kod modela s latentnim varijablama tražimo parametar θ tako da je izglednost

$$\mathcal{L}(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Z}) = p(\mathcal{X}, \mathcal{Z} \mid \theta) = \sum_z p(\mathcal{X} \mid z, \theta) p(z \mid \theta)$$

maksimalna, uz dani skup primjera $\mathcal{X} = \{x^{(i)}\}_{i=1}^n$ i skup latentnih varijabli $\mathcal{Z} = \{z_i\}_{i=1}^k$. Ovakvu izglednost zovemo potpuna izglednost (eng. complete likelihood).

Želimo naći donju ogradu za potpunu izglednost i zatim, za donju ogradu naći parametar θ koji ju maksimizira. Iz Jensenove nejednakosti za konveksnu funkciju f

$$\lambda f(x) + (1 - \lambda) f(y) \geq f(\lambda x + (1 - \lambda) y)\,, \lambda \in \langle 0, 1 \rangle\,,$$

dobivamo

$$\mathbb{E}[f(x)] = \sum_x p(x) f(x) \geq f\Big(\sum_x p(x) x \Big) = f(\mathbb{E}[x])\,,$$

gdje je $\mathbb{E}[\cdot]$ oznaka za očekivanje. Za konkavnu funkciju f vrijedi

$$\mathbb{E}[f(x)] \leq f(\mathbb{E}[x])\,.$$

Vratimo se na potpunu log-izglednost. Neka je $Q \not\equiv 0$ neka razdioba latentnih varijabli z. Onda je

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Z}) &= \log \sum_z p(\mathcal{X} \mid z, \boldsymbol{\theta}) p(z \mid \boldsymbol{\theta}) \\
&= \log \sum_z p(\mathcal{X} \mid z, \boldsymbol{\theta}) p(z \mid \boldsymbol{\theta}) \frac{Q(z)}{Q(z)} \\
&= \log \mathbb{E}_{z \sim q} \left[\frac{p(\mathcal{X} \mid z, \boldsymbol{\theta}) p(z \mid \boldsymbol{\theta})}{Q(z)} \right] \\
&\geq \mathbb{E}_z \left[\log \frac{p(\mathcal{X} \mid z, \boldsymbol{\theta}) p(z \mid \boldsymbol{\theta})}{Q(z)} \right]. \qquad (3.3)
\end{aligned}$$

Dobili smo donju ogradu za potpunu izglednost, za bilo koju razdiobu Q latentnih varijabli z. Postoji mnogo mogućnosti za razdiobu Q i pitamo se koja je najbolja. Uočimo da ako tražimo parametar $\boldsymbol{\theta}$ takav da gornja nejednakost postane jednakost, da ujedno i maksimiziramo donju ogradu. Znamo da će to biti ispunjeno ako očekivanje djeluje na konstantnu

slučajnu varijablu. Zato zahtijevamo $\frac{p(\mathcal{X},z\,|\,\boldsymbol{\theta})}{Q(z)} = c$, za slučajnu varijablu c koja ne ovisi o ni jednom z. To je ispunjeno uz uvjet $Q(z) \propto p(\mathcal{X}, z \,|\, \boldsymbol{\theta})$. Budući da je Q razdioba latentnih varijabli z, vrijedi $\sum_z Q(z) = 1$, a onda slijedi da je $c = \sum_z p(\mathcal{X}, z \,|\, \theta)$. Konačno,

$$Q(z) = \frac{p(\mathcal{X}, z \,|\, \boldsymbol{\theta})}{\sum\limits_z p(\mathcal{X}, z \,|\, \boldsymbol{\theta})} = \frac{p(\mathcal{X}, z \,|\, \boldsymbol{\theta})}{p(\mathcal{X} \,|\, \boldsymbol{\theta})} = p(z \,|\, \mathcal{X}, \boldsymbol{\theta})\,.$$

U EM-algoritmu, čiji je pseudo-kod dan u Algoritmu 1, u E-koraku za svaki z računamo $Q(z)$ kao posteriornu vjerojatnost od z uz dani $\mathcal{X}$ i $\boldsymbol{\theta}$. To je razdioba Q koja maksimizira donju ogradu u (3.3). U M-koraku za izračunatu razdiobu Q i trenutni $\boldsymbol{\theta}$ tražimo nove parametre $\boldsymbol{\theta}^*$ takve da je izglednost maksimalna.

Algoritam 1: EM-algoritam

initialize $\theta^{(0)}$ proizvoljan, t $\leftarrow 0$
repeat
  E-korak:
  for *za svaki* $x^{(i)} \in \mathcal{X}$ *i svaki* $z^{(j)} \in \mathcal{Z}$ **do**
    definiraj $q_i(z^{(j)}) = p(z^{(j)} \mid x^{(i)}, \theta^{(t)})$
  end
  M-korak:
  $\theta^* = \underset{\theta}{\operatorname{argmax}} \sum\limits_i \sum\limits_j q_i(z^{(j)}) \log \dfrac{p(x^{(i)}, z^{(j)} \mid \theta^{(t)})}{q_i(z^{(j)})}$
  $t \leftarrow t + 1$
  $\theta^{(t)} \leftarrow \theta^*$
until *konvergencija*;

Primjena EM algoritma u pLSA

Neka je uređeni par (d, t) oznaka da se pojam t iskoristio u dokumentu d. Skup primjera $\mathcal{X}$ kod pLSA je skup

$$\{(d_i, t_i) : d_i \in \mathcal{D}, t_i \in \mathcal{T}, i = 1, \ldots, c \leq m \cdot n\} \subseteq \mathcal{D} \times \mathcal{T}\,,$$

gdje je c broj ostvarenih parova (d, t) u kolekciji. Dio tog skupa uzimamo za učenje parametara, a dio za treniranje modela. Radi jednostavnosti, dalje pišemo kao da smo cijelu kolekciju uzeli za treniranje. Skup latentnih varijabli je $\mathcal{Z} = \{z_i : i = 1, \ldots, k\}$. Dalje ćemo raditi s forumulom (3.2) za zajedničku vjerojatnost $p(d, t) = \sum\limits_z p(t|z)p(d|z)p(z)$. Iskoristit ćemo EM-algoritam opisan u prethodnom potpoglavlju. U E-koraku računamo posteriorne

vjerojatnosti

$$q_i(z) = p(z \mid d_i, t_i) = \frac{p(z, d_i, t_i)}{p(d_i, t_i)} = \frac{p(t_i \mid z)p(d_i \mid z)p(z)}{\sum_z p(t_i \mid z)p(d_i \mid z)p(z)},$$

a u M-koraku maksimiziramo log-izglednost po $\boldsymbol{\theta} = (p(d), p(z \mid d), p(t \mid z))$

$$l(\boldsymbol{\theta}) = \sum_{(d_i,t_i)\in\mathcal{X}} \sum_{z\in\mathcal{Z}} p(z \mid d_i, t_i) \log \frac{p(d_i, t_i, z)}{p(z \mid d_i, t_i)} = \sum_{d_i\in\mathcal{D}} \sum_{t_j\in\mathcal{T}} n(d_i, t_j) \sum_{z\in\mathcal{Z}} p(z \mid d_i, t_j) \log \frac{p(d_i, t_j, z)}{p(z \mid d_i, t_j)}.$$

Parametri su $p(z)$, $p(t \mid z)$ i $p(z \mid d)$, pa je maksimiziranje prethodne fukncije ekvivalentno maksimiziranju ciljne funkcije

$$\begin{aligned}\tilde{l}(\boldsymbol{\theta}) &= \sum_{d_i\in\mathcal{D}} \sum_{t_j\in\mathcal{T}} n(d_i, t_j) \sum_{z\in\mathcal{Z}} p(z \mid d_i, t_j) \log p(d_i, t_j, z) \\ &= \sum_{d_i\in\mathcal{D}} \sum_{t_j\in\mathcal{T}} n(d_i, t_j) \sum_{z\in\mathcal{Z}} p(z \mid d_i, t_j)\Big(\log p(t_i \mid z) + \log p(d_i \mid z) + \log p(z)\Big),\end{aligned}$$

uz uvjete

$$\sum_z p(z) = 1, \sum_z p(t \mid z) = 1, \sum_z p(d \mid z) = 1.$$

Kombiniranjem ciljne funkcije i rubnih uvjeta dobivamo sljedeću Lagrangeovu funkciju

$$L(\boldsymbol{\theta}) = \tilde{l}(\boldsymbol{\theta}) + \alpha_1\Big(1 - \sum_z p(z)\Big) + \alpha_2\Big(1 - \sum_z p(t \mid z)\Big) + \alpha_3\Big(1 - \sum_z p(d \mid z)\Big),$$

gdje su α_i, $i = 1, 2, 3$, Lagrangeovi multiplikatori. Deriviranjem slijedi

$$\begin{aligned}\frac{\partial L}{\partial p(z)} &= \sum_{d_i\in\mathcal{D}} \sum_{t_j\in\mathcal{T}} n(d_i, t_j) \frac{p(z \mid d_i, t_j)}{p(z)} - \alpha_1 = 0, \\ \frac{\partial L}{\partial p(t \mid z)} &= \sum_{d_i\in\mathcal{D}} \sum_{t_j\in\mathcal{T}} n(d_i, t_j) \frac{p(z \mid d_i, t_j)}{p(t \mid z)} - \alpha_2 = 0, \\ \frac{\partial L}{\partial p(d \mid z)} &= \sum_{d_i\in\mathcal{D}} \sum_{t_j\in\mathcal{T}} n(d_i, t_j) \frac{p(z \mid d_i, t_j)}{p(d \mid z)} - \alpha_3 = 0.\end{aligned} \tag{3.4}$$

Prvu jednakost u (3.4) pomnožimo s $p(z)$ i zatim djelujemo na nju sumom po svim latentnim varijablama z, pa dobijemo

$$\sum_{d_i\in\mathcal{D}} \sum_{t_j\in\mathcal{T}} \sum_{z_l\in\mathcal{Z}} n(d_i, t_j) p(z_l \mid d_i, t_j) - \alpha_1 \underbrace{\sum_{z_l\in\mathcal{Z}} p(z_l)}_{=1} = 0.$$

Dakle, dobili smo α_1 i u jednadžbi (3.4) je ostala samo jedna nepoznanica, $p(z)$. Slijedi

$$p(z) = \frac{\sum_{d_i \in \mathcal{D}} \sum_{t_j \in \mathcal{T}} n(d_i, t_j) p(z \mid d_i, t_j)}{\sum_{d_i \in \mathcal{D}} \sum_{t_j \in \mathcal{T}} \sum_{z_l \in \mathcal{Z}} n(d_i, t_j) p(z_l \mid d_i, t_j)} = \frac{\sum_{d_i \in \mathcal{D}} \sum_{t_j \in \mathcal{T}} n(d_i, t_j) p(z \mid d_i, t_j)}{\underbrace{\sum_{d_i \in \mathcal{D}} \sum_{t_j \in \mathcal{T}} n(d_i, t_j)}_{c}},$$

gdje je c broj primjera.

Analogno se dobije

$$p(t \mid z) = \frac{\sum_{d_i \in \mathcal{D}} n(d_i, t) p(z \mid d_i, t)}{\sum_{d_i \in \mathcal{D}} \sum_{t_j \in \mathcal{T}} n(d_i, t_j) p(z \mid d_i, t_j)},$$

$$p(d \mid z) = \frac{\sum_{t_j \in \mathcal{D}} n(d, t_j) p(z \mid d, t_j)}{\sum_{d_i \in \mathcal{D}} \sum_{t_j \in \mathcal{T}} n(d_i, t_j) p(z \mid d_i, t_j)}.$$

Kako bi se izbjegla prenaučenost (eng. overfitting), autor originalnog članka [9] predložio je varijaciju EM algoritma, zvanu "tempered EM".

3.2 Usporedba LSA i pLSA

Budući da smo pretpostavili da je svaki kontekst razdioba riječi iz rječnika $\mathcal{T}$, onda razdiobu $p(\cdot \mid z_i)$ možemo prikazati kao točku u $m-1$ dimenzionalnom simpleksu svih razdioba na $\mathcal{T}$. Konveksna ljuska k točaka tog simpleksa čini konveksan skup R. Iz jednadžbe

$$p(t \mid d) = \sum_z \underbrace{p(z \mid d)}_{\text{koeficijenti}} p(t \mid z)$$

vidimo da su sve uvjetne vjerojatnosti $p(\cdot \mid d_i)$, $i = 1, \ldots, n$, konveksne kombinacije k razdioba latentnih koncepata. Koeficijenti konveksne kombinacije $p(z_l \mid d_i)$, $l = 1, \ldots, k$, u potpunosti određuju vjerojatnost $p(\cdot \mid d_i)$, $i = 1, \ldots, n$. Dakle, vjerojatnost $p(\cdot \mid d_i)$, $i = 1, \ldots, n$, je točka područja R. Budući da je dimenzija područja R jednaka $k-1$, koja je standardno puno manja od dimenzije $(m-1)$-simpleksa, i kod pLSA imamo svojevrsnu redukciju dimenzije. Svaki smjer u probabilističkom, latentnom, semantičkom području R odgovara nekom latentnom kontekstu.

Kako bismo vidjeli još neke poveznice s LSA, preformulirat ćemo model za zajedničku vjerojatnost u matričnoj notaciji. Definiramo matrice $U = (p(d_i \mid z_k))_{i,k}$, $V = (p(t_j \mid z_k))_{j,k}$ i $\Sigma = \mathrm{diag}(p(z_k))$. Tada matrica $P = U\Sigma V^T$ sadrži zajedničke vjerojatnosti i odgovara aproksimaciji pojmovno-dokumentne matrice matricom nižeg ranga. Dobiveni elementi matrice P imaju vjerojatnosnu interpretaciju, jer je svaki dokument u reprezentaciji miješani model latentnih koncepata, dok kod LSA to nemamo. Dapače, dobivene vrijednosti kod LSA mogu biti i negativne. Nadalje, broj latentnih koncepata k može se dobiti statističkom teorijom za selekciju modela (end. model selection) i kontrolu složenosti (eng. complexity control), dok određivanje broja k u LSA radimo eksperimentalno. Nadalje, EM-algoritam garantira da će pronaći samo lokalni, ali ne i globalni maksimum izglednosti. Dakle, pitamo se koliko ta činjenica utječe na točnost modela i ako znamo da je lokalni maksimum skoro dobar kao globalni, kolika je vremenska složenost pronalaska rješenja? U mnogim eksperimentima na raznim skupovima podataka utvrđeno je da je prenaučenost modela uzrok toga što s pLSA nismo dobili bar približno globalno rješenje. Dakle, ako izbjegnemo prenaučenost, vrlo vjerojatno ćemo dobiti i globalni maksimum. Prenaučenost se može izbjeći varijacijom EM algoritma – "tempered" EM, koja je opisana u originalnom članku [9] ili regularizacijskim metodama koje su zasnovane na tome da se iterativni postupak zaustavi ako je razlika vrijednosti dobivenih u prošloj i trenutnoj iteraciji manja od neke unaprijed zadane, male vrijednosti (eng. early stopping methods). Također, može se pokazati da izbor inicijalnih vrijednosti ne utječe značajno na izvedbu algoritma, te da pLSA uz najlošiju maksimizaciju izglednosti ima bolje rezultate od LSA. Broj iteracija je najčešće oko 20 do 50, što ne troši više resursa od računanja SVD-a.

3.3 Uspoređivanje dokumenata

Kako bismo mogli koristiti pLSA u primjeni, moramo definirati sličnost dokumenata u tom modelu. Najjednostavniji način je da sličnost definiramo kao euklidsku udaljenost vektora dokumenata u $(k-1)$-dimenzionalnom području R, kojeg smo opisali u prošlom odjeljku. Preciznije $sim(d_1, d_2) = \left\| p(\mathcal{Z} \mid d_1) - p(\mathcal{Z} \mid d_2) \right\|$. Međutim, pokazalo se da ta mjera sličnosti nije pogodna za ovakvu vrstu problema.

Druga mjera sličnosti za dvije razdiobe je KL-divergencija. Za diskretne slučajne varijable P i R, KL-divergencija se definira kao

$$\mathrm{KL}(P, R) = \sum_i \log P(i) \cdot \frac{P(i)}{R(i)}\,. \tag{3.5}$$

U pLSA modelu uspoređujemo dvije posteriorne razdiobe latentnih varijabli, uz dani dokument d_1, odnosno, d_2

$$\mathrm{KL}\Big(p(\cdot \mid d_1), p(\cdot \mid d_1)\Big) = \sum_z p(z \mid d_1) \log \frac{p(z \mid d_1)}{p(z \mid d_2)}\,.$$

Međutim, ona nije simetrična i zato ju je teško interpretirati. Osim toga, ne koristi vjerojatnost koncepta u cijeloj kolekciji, $p(z)$. Ako dva dokumenta d_1 i d_2 imaju veliku vjerojatnost uz koncept z, ta činjenica je još važnija ako je $p(z)$ malen, odnosno, malo dokumenata govori o tom konceptu.

Dakle, treba nam definicija sličnost koja je bazirana na statističkoj teoriji, ali koja ima i svojstva dobro definirane metrike. Jedna takva sličnost je Fisherova jezgra, koju ćemo izvesti za pLSA.

Fisherova jezgra

Pretpostavimo da su podaci $\mathcal{X} = \{x^{(i)}\}_{i=1}^{n}$ generirani parametarskim modelom $p(x \mid \Theta)$, gdje je Θ skup parametara modela. Pretpostavimo da parametre modele tražimo tako da maksimiziramo izglednost

$$\Theta^* = \operatorname*{argmax}_{\Theta} \mathcal{L}(\Theta \mid \mathcal{X}) .$$

Ako su podaci nezavisni i identično distribuirani, vrijedi

$$\Theta^* = \operatorname*{argmax}_{\Theta} \prod_{i=1}^{n} p(x^{(i)} \mid \Theta) = \operatorname*{argmax}_{\Theta} \sum_{i=1}^{n} \log p(x^{(i)} \mid \Theta) .$$

Tada je

$$\sum_{i=1}^{n} \nabla_{\Theta} \log p(x^{(i)} \mid \Theta^*) = 0 ,$$

gdje je ∇_{Θ} oznaka za gradijent. Definiramo sljedeće vrijednosti

- $U_x = \nabla_{\Theta} \log p(x \mid \Theta) \ldots$ Fisherov rezultat (eng. Fisher score) ,
- $I_F = \mathbb{E}[U_x U_x^T] \ldots$ Fisherova informacija ,
- $\Phi(x) = I_F^{-1} U_x \ldots$ smjer najvećeg porasta $p(x \mid \Theta)$.

Sada možemo definirati jezgrenu funkciju

$$K(x, y) = \Phi(x)^T I_F^{-1} \Phi(Y) . \tag{3.6}$$

Hofmann je u [10] pokazao da Fisherova jezgra primijenjena na pLSA glasi

$$K^H(d, q) = \sum_{z} \frac{p(z \mid d) p(z \mid q)}{p(z)} + \sum_{t} \frac{n(d, t)}{|d|} \frac{n(q, t)}{|q|} \sum_{z} \frac{p(z \mid d, t) p(z \mid q, t)}{p(t \mid z)} . \tag{3.7}$$

Međutim, nije opravdao normalizaciju s $|d|$ i $|q|$, te je Fisherovu informacijsku matricu aproksimirao s identitetom, iako teorijska i eksperimentalna razmatranja nisu u skladu s

tim. U [16], autori su izveli bolju Fisherovu jezgru za pLSA, koju ćemo izvesti u nastavku. Dakle, sada je cilj pronaći izraz za (3.6), koji nema nedostatke koje ima (3.7). Krenimo načinom na koji je Hofmann izveo Fisherovu jezgru. On je parametrizirao parametre modela s njihovim drugim korijenom (eng. square root re-parametrisation)

$$\begin{aligned}
\varphi(z) = 2\sqrt{p(z)} \quad &\Rightarrow \quad p(z) = \frac{1}{4}\varphi^2(z) \quad &&\Rightarrow \quad \frac{\partial p(z)}{\partial \varphi(z)} = \frac{1}{2}\varphi(z) = \sqrt{p(z)}\,, \\
\varphi(d \mid z) = 2\sqrt{p(d \mid z)} \quad &\Rightarrow \quad p(d \mid z) = \frac{1}{4}\varphi^2(d \mid z) \quad &&\Rightarrow \quad \frac{\partial p(d \mid z)}{\partial \varphi(d \mid z)} = \frac{1}{2}\varphi(d \mid z) = \sqrt{p(d \mid z)}\,, \\
\varphi(t \mid z) = 2\sqrt{p(t \mid z)} \quad &\Rightarrow \quad p(t \mid z) = \frac{1}{4}\varphi^2(t \mid z) \quad &&\Rightarrow \quad \frac{\partial p(t \mid z)}{\partial \varphi(t \mid z)} = \frac{1}{2}\varphi(t \mid z) = \sqrt{p(t \mid z)}\,.
\end{aligned}$$

Kako bismo raspisali (3.6), moramo izračunati Fisherov rezultat uz skup parametara $\Theta = (p(z), p(d \mid z), p(t \mid z))$,

$$U_d(\Theta) = \nabla_d \mathcal{L}(\Theta \mid \mathcal{D}) = \nabla_d \sum_{t \in \mathcal{T}} n(d,t) \log\Big(\sum_z p(z)p(t \mid z)p(d \mid z)\Big)\,.$$

Označimo, radi jednostavnosti, $L_d = \mathcal{L}(\Theta \mid \mathcal{D})$. Računamo

$$\begin{aligned}
\frac{\partial L_d}{\partial \varphi(z)} &= \frac{\partial L_d}{\partial p(z)}\frac{\partial p(z)}{\partial \varphi(z)} = \sqrt{p(z)} \sum_{t \in \mathcal{T}} n(d,t) \frac{p(d \mid z)p(t \mid z)}{\sum_z p(z)p(d \mid z)p(t \mid z)}\,, \\
\frac{\partial L_d}{\partial \varphi(t \mid z)} &= \frac{\partial L_d}{\partial p(t \mid z)}\frac{\partial p(t \mid z)}{\partial \varphi(t \mid z)} = \sqrt{p(t \mid z)} n(d,t) \frac{p(d \mid z)p(z)}{\sum_z p(z)p(d \mid z)p(t \mid z)}\,.
\end{aligned} \tag{3.8}$$

Definiramo

$$\alpha_i = \alpha(z_i) = \sum_d \left(\frac{\partial L_d}{\partial \varphi(z_i)}\right)^2, i = 1, \ldots, k, j = 1, \ldots, m\,,$$

$$\gamma_{i,j} = \gamma(t_j, z_i) = \sum_d \left(\frac{\partial L_d}{\partial \varphi(t_j \mid z_i)}\right)^2, i = 1, \ldots, k, j = 1, \ldots, m\,,$$

i očekivanje, koje je Hofmann aproksimirao identitetom, procjenjujemo s

$$\mathbb{E}[\nabla L_d \nabla L_d^T] \sim \sum_d \nabla L_d \nabla L_d^T\,.$$

Definiramo Fisherovu informacijsku matricu

$$I_F(\varphi) = \begin{pmatrix} \alpha_1 & & & & & & \\ & \ddots & & & & & \\ & & \alpha_k & & & & \\ & & & \gamma_{11} & & & \\ & & & & \ddots & & \\ & & & & & \gamma_{mk} \end{pmatrix}.$$

Izračunali smo sve što se pojavljuje u definiciji Fisherove jezgrene funkcije i ona sada glasi

$$K^N(d,q) = \underbrace{\sum_z \frac{\partial L_d}{\partial \varphi(z)} \alpha_z^{-1} \frac{\partial L_q}{\partial \varphi(z)}}_{K_z(d,q) = \text{doprinos od z}} + \underbrace{\sum_t \sum_z \frac{\partial L_d}{\partial \varphi(t \mid z)} \gamma_{t,z}^{-1} \frac{\partial L_q}{\partial \varphi(t \mid z)}}_{K_t(d,q) = \text{doprinos od t}}. \tag{3.9}$$

Jezgrenu funkciju smo označili s K^N, jer ju je prvi izveo Nyffenegger. Koristeći (3.2) i (3.8), raspišimo $K_z(d,q)$ i $K_t(d,q)$,

$$K_z(d,q) = \sum_z \left\{ \sqrt{p(z)} \sum_t n(d,t) \frac{p(d \mid z)p(t \mid z)}{p(d,t)} \cdot \sqrt{p(z)} \sum_t n(q,t) \frac{p(q \mid z)p(t \mid z)}{p(q,t)} \cdot \right.$$
$$\left. \cdot \left(p(z) \sum_d \sum_t \left(n(d,t) \frac{p(d \mid z)p(t \mid z)}{p(d,t)} \right)^2 \right)^{-1} \right\},$$

$$K_t(d,q) = \sum_z \sum_t \left\{ \sqrt{p(t \mid z)} n(d,t) \frac{p(d \mid z)p(z)}{p(d,t)} \cdot \sqrt{p(t \mid z)} n(q,t) \frac{p(q \mid z)p(z)}{p(q,t)} \cdot \right.$$
$$\left. \cdot \left(p(t \mid z) \sum_d \left(n(d,t) \frac{p(d \mid z)p(z)}{p(d,t)} \right)^2 \right)^{-1} \right\}.$$

Pojednostavimo te forumule da dobijemo kompaktniji zapis relacije (3.9), iz kojeg možemo više toga uočiti. Počnimo s $K_z(d,q)$

$$\begin{aligned} K_z(d,q) &= \sum_z \left\{ p(z) \sum_t n(d,t) \frac{p(d \mid z)p(t \mid z)}{p(d,t)} \cdot \sum_t n(q,t) \frac{p(q \mid z)p(t \mid z)}{p(q,t)} \cdot \alpha_z^{-1} \right\} \\ &= \sum_z \left\{ \underbrace{p(z)p(d \mid z)p(q \mid z)}_{I} \underbrace{\sum_t n(d,t) \frac{p(t \mid z)}{p(d,t)}}_{J} \cdot \sum_t n(q,t) \frac{p(t \mid z)}{p(q,t)} \cdot \alpha_z^{-1} \right\}. \end{aligned}$$

Bayesovim pravilom možemo dobiti da je

$$I = p(z)p(d \mid z)p(q \mid z) = \frac{p(d,z)p(q,z)}{p(z)}\,.$$

Želimo aproksimirati J. Definiramo razdiobu $\hat{p}$ s $\hat{p}(d,t) = \dfrac{n(d,t)}{m}$, $d \in \mathcal{D}$, $t \in \mathcal{T}$. Ako neki parametri $\boldsymbol{\theta}$ maksimiziraju log-izglednost

$$l_{\log}(\boldsymbol{\theta}) = \sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} n(d,t) \log p(d,t;\boldsymbol{\theta})\,,$$

onda iz definicije KL-divergencije (3.5), slijedi da ti parametri minimiziraju KL-divergenciju između p i $\hat{p}$,

$$\begin{aligned}\mathrm{KL}(p,\hat{p}) &= \sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} \hat{p}(d,t) \log \frac{\hat{p}(d,t)}{p(d,t \mid \boldsymbol{\theta})} \\ &= \sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} \frac{n(d,t)}{m} \log \frac{n(d,t)}{m} - \sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} \frac{n(d,t)}{m} \log p(d,t)\,.\end{aligned}$$

Ako tražimo parametre koji minimiziraju KL-divergenciju između p i $\hat{p}$, onda želimo da su p i $\hat{p}$ što sličnije. Dakle, tražimo procjene za $p(d,t)$ koje aproksimiraju $\hat{p}(d,t)$ po svim d i t, odnosno,

$$\sum_t \frac{\hat{p}(d,t)}{p(d,t)} \approx 1\,, \text{ za svaki } d \in \mathcal{D}\,.$$

Budući da je $\displaystyle\sum_t p(t \mid z) = 1$, vrijedi

$$\sum_t \hat{p}(d,t) \frac{p(t \mid z)}{p(d,t)} \approx 1\,.$$

Konačno, imamo da je $J \approx c$, gdje je $c = \displaystyle\sum_{d,t} n(d,t)$ broj opažanja. Slijedi

$$K_z(d,q) = c^2 \sum_z \frac{p(d,z)p(q,z)}{p(z)} \cdot \alpha_z^{-1}\,.$$

Preostaje raspisati što je α_z^{-1}. Raspisujemo direktno

$$\begin{aligned}\alpha_z &= \sum_d \left(\frac{\partial L_d}{\partial \varphi(z)}\right)^2 = \sum_d \sum_z p(z) \cdot \Big(\sum_t n(d,t) \cdot \frac{p(d \mid z)p(t \mid z)}{p(d,t)} \Big)^2 \\ &= \sum_d \sum_z p(z)p(d \mid z)^2 \cdot \Big(\underbrace{\sum_t n(d,t) \cdot \frac{p(t \mid z)}{p(d,t)}}_{c^2} \Big)^2 \\ &= c^2 \sum_d \sum_z p(z)p(d \mid z)^2 = c^2 \sum_d \frac{p^2(d,z)}{p(z)}\,.\end{aligned}$$

I tim smo pojednostavili $K_z(d,q)$. Preostaje napraviti sličan postupak za $K_t(d,q)$. Krenimo raspisivati direktno

$$\begin{aligned}K_t(d,q) &= \sum_t \sum_z \left\{ \sqrt{p(t \mid z)} \cdot n(d,t) \cdot \frac{p(d \mid z)p(z)}{p(d,t)} \cdot \sqrt{p(t \mid z)} \cdot n(q,t) \cdot \frac{p(q \mid z)p(z)}{p(q,t)} \cdot \gamma_{t,z}^{-1} \right\} \\ &= \sum_t n(d,t) \cdot n(q,t) \sum_z \left\{ \underbrace{\frac{p(t \mid z)p(d \mid z)p(z)}{p(d,t)}}_{I} \cdot \frac{p(t \mid z)p(q \mid z)p(z)}{p(q,t)} \cdot \frac{1}{p(t \mid z)} \cdot \gamma_{t,z}^{-1} \right\}.\end{aligned}$$

Definiramo

$$I = \frac{p(t \mid d,z)p(d \mid z)p(z)}{p(d,t)} = \frac{p(z,d,t)}{p(d,t)} = p(z \mid d,t)\,,$$

te uočimo da je $n(\delta,t) = c \cdot \hat{p}(\delta,t)$. Slijedi

$$K_t(d,q) = c^2 \sum_t \hat{p}(d,t) \cdot \hat{p}(q,t) \sum_z \left\{ \frac{p(z \mid d,t) \cdot p(z \mid q,t)}{p(t \mid z)} \cdot \gamma_{t,z}^{-1} \right\}.$$

Računamo $\gamma_{t,z}$ direktnim raspisivanjem

$$\gamma_{t,z} = \sum_d p(d) \cdot \left(\frac{\partial L_d}{\partial \varphi(t \mid z)}\right)^2 = \sum_d p(t \mid d) \cdot \left(n(d,t) \cdot \frac{p(d \mid z)p(z)}{p(d,t)}\right)^2 = \sum_d \left(n(d,t) \cdot \frac{p(z \mid d,t)}{\sqrt{p(t \mid z)}}\right)^2.$$

Sada možemo zapisati jezgru (3.9) u jednostavnijem obliku

$$\begin{aligned}
K^N(d,q) &= K_z(d,q) + K_t(d,q) \\
&= c^2 \sum_z \frac{p(d,z)p(q,z)}{p(z)} \cdot \left[c^2 \sum_d \frac{p^2(d,z)}{p(z)} \right]^{-1} + \dots \\
&+ c^2 \sum_t \hat{p}(d,t) \cdot \hat{p}(q,t) \sum_z \left\{ \frac{p(z \mid d,t) \cdot p(z \mid q,t)}{p(t \mid z)} \cdot \left[\sum_d \left(n(d,t) \cdot \frac{p(z \mid d,t)}{\sqrt{p(t \mid z)}} \right)^2 \right]^{-1} \right\} \\
&= \sum_z \frac{p(d,z)p(q,z)}{p(z)} \cdot \left[\sum_d \frac{p^2(d,z)}{p(z)} \right]^{-1} + \dots \\
&+ c^2 \sum_t \hat{p}(d,t) \cdot \hat{p}(q,t) \sum_z \left\{ \frac{p(z \mid d,t) \cdot p(z \mid q,t)}{p(t \mid z)} \cdot \left[\sum_d \left(n^2(d,t) \cdot \frac{p^2(z \mid d,t)}{p(t \mid z)} \right) \right]^{-1} \right\}.
\end{aligned}$$

Vidimo da zbrajamo faktore koji se pojavljuju i u Hofmannovoj jezgri (3.7), ali pomnožene s nekim koeficijentima koji predstavljaju doprinos Fisherove informacijske matrice. Osim toga, Nyffenegger je, umjesto $p(d \mid z)$ i $\hat{p}(d \mid z)$, koristio $p(d,z)$ i $\hat{p}(d,z)$, te je na taj način izbjegao normaliziranje s $|d|$, odnosno $|q|$, koje Hofmann u svom radu nije opravdao.

Istaknimo neka svojstva probabilističke latentne semantičke analize. To je model u kojem svaki dokument prikazujemo kao mješavinu unaprijed zadanih latentnih koncepata. Budući da za dokumente nemamo oznake koji latentni koncepti im pripadaju, već to učimo na skupu podataka za učenje, riječ je o algoritmu nenadziranog strojnog učenja. Osim toga, zato što više koncepata može pripadati nekom dokumentu, radi se i o "mekom" grupiranju (eng. soft clustering). Nadalje, jer pokušavamo pronaći skrivene tematske strukture u kolekciji dokumenata, pLSA pripada i u skupinu modela tema (eng. topic model) i modela s latentnim varijablama (eng. latent variable model).

Kod pLSA, kao kod LSA, možemo provesti "umetanje" (eng. folding-in) dokumenata koji nisu bili dio originalne kolekcije. Za pLSA, Hofmann je to napravio u [9], tako da je za novi dokument q opet koristio EM-algoritam za procjenu $p(q \mid z)$, ali uz fiksirane $p(t \mid z)$ i $p(z)$, koji su naučeni na danoj kolekciji. Ovakav pristup ima mnoge probleme o kojima se više može pročitati u [19].

Poglavlje 4

Latentna Dirichletova alokacija

Algoritmi modela tema (eng. topic model algorithms) su statističke metode, koje analizirajući riječi originalnoga teksta pronalaze teme koje se pojavljuju u njemu. Jedna takva metoda je probabilistička latentna semantička analiza (pLSA), koju smo prikazali u prethodnom poglavlju. Međutim, uočili smo da pLSA nije pravi generativni algoritam, jer vjerojatnosti $p(z \mid d)$ shvaća kao parametre modela i zato s njom ne možemo generirati neviđene dokumente (prenaučili smo model). Želimo pravi generativni proces i to ćemo dobiti sljedećom metodom, latentnom Dirichletovom alokacijom (eng. latent Dirichlet allocation), koju označavamo s LDA. Ta metoda tretira spomenute vjerojatnosti kao distribuciju koja ovisi o nekom parametru. U nastavku ćemo dati glavnu ideju LDA, a zatim, prateći [3], njezin formalni izvod.

Temu definiramo kao distribuciju riječi iz fisknog rječnika. Dakle, svakoj temi pridodajemo vjerojatnosti za svaku riječ iz rječnika. Na primjer, ako je tema "genetika", visoku vjerojatnost će imati riječi "geni" ili "dnk". Uočimo da neke riječi, kao višeznačnice, mogu imati visoke vjerojatnosti u različitim temama. Pretpostavljamo da opaženi podaci, koje riječi se pojavljuju u kojem dokumentu, dolaze iz nekog generativnog, vjerojatnostnog procesa. Pokušaj realiziranja jednog takvog procesa je opisan sljedećim koracima:

1. Prije samog generiranja dokumenata, za svaku temu napravimo njezinu distribuciju po riječima iz rječnika;
2. Za svaki dokument:
   (a) Napravimo distribuciju distribucija tema iz prvog koraka;
   (b) Nasumično odaberemo temu iz distribucije distribucija tema;
   (c) Nasumično odaberemo riječ iz distribucije teme odabrane u koraku (b).

Uočimo da smo zanemarili poredak riječi. Kad bismo zaista generirali dokument na ovakav način, dobili bismo nešto što nije čitljivo. Međutim, ako nam je cilj samo pronaći teme koje

se javljaju u dokumentu, pogledom na neuređenu hrpu riječi, možemo dobiti predodžbu o kojim temama dokument govori. Važno je uočiti da svaki dokument u kolekciji dijeli jednak skup tema, ali s drugačijim proporcijama.

Ovaj proces bismo htjeli realizirati, ali sve što imamo su opaženi dokumenti. Zato nam je cilj pronaći latentnu, tematsku strukturu - teme koje su generirale dokumente, distribuciju distribucija tema za svaki dokument, te čak i za svaku riječ u dokumentu, koja tema ju je generirala. Za distribuciju distribucija tema nam se prirodno nameće Dirichletova distribucija. Zato ćemo prvo dati njezin kratki pregled.

4.1 Dirichletova distribucija

Funkcija gustoće k-dimenzionalne Dirichletove slučajne varijable $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ s parametrima $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$, $\alpha_i > 0$, $i = 1, \ldots, k$, koja se nalazi u $k-1$ dimenzionalnom simpkesu $S = \{x_i \in \mathbb{R}^k : x_i \geq 0, \sum_{i=1}^{k} x_i = 1\}$ glasi

$$f(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1},$$

gdje je $B(\boldsymbol{\alpha})$ beta funkcija definirana preko gama funkcije $\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx$, na sljedeći način

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)}. \tag{4.1}$$

Najveći nedostatak procjenitelja najveće izglednosti je što je dobiven koristeći samo znanje dobiveno iz uzorka. Na taj način lako prenaučimo model. Kad bismo imali neko apriorno znanje o distribuciji parametara razdiobe, bilo bi ga korisno ukomponirati u procjenitelj. Na primjer, ako bacamo simetrični novčić, znamo da ćemo u velikom broju bacanja imati otprilike podjednak broj pisma i glave. Međutim, ako novčić bacimo par puta, može se dogoditi da ni jednom ne padne pismo. Takav pokus modeliramo Bernoullijevom slučajnom varijablom X i dobiveni procjenitelj najveće izglednosti za ovaj pokus je $\mu_{ML} = p(X = \text{'pismo'}) = 0$. Takvom procjenom ne možemo biti zadovoljni, jer nam iskustvo govori da to nije realno. U tu svrhu bayesovski procjenitelj kombinira znanje iz uzroka s apriornim znanjem o mogućim parametrima θ. Kako parametar θ ne znamo (jer inače ne bismo tražili njegovu procjenu), tretiramo ga kao slučajnu varijablu i definiramo njegovu distribuciju $p(\theta)$. Na primjer, za bacanje simetričnog novčića, visoku vjerojatnost bi dali $\mu = \frac{1}{2}$, a malu $\mu = 0$. Koristeći Bayesovo pravilo, kombiniramo apriorno znanje sa

znanjem dobivenim iz uzorka

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{\int p(\mathcal{D} \mid \hat{\theta})p(\hat{\theta})d\hat{\theta}}.$$

Definiramo Bayesovski procjenitelj

$$\hat{\theta}_{\text{Bayes}} = \mathbb{E}[\theta \mid \mathcal{D}] = \int \theta p(\theta \mid \mathcal{D})d\theta.$$

Ovisno o umnošku $p(\mathcal{D} \mid \theta)p(\theta)$, takav procjenitelj može biti teško izračunljiv. Međutim, ako $p(\theta)$ i $p(\theta \mid \mathcal{D})$ dolaze iz iste familije distribucija, taj račun postaje jednostavan. Za familiju $\mathcal{F}$ apriornih distribucija $p(\theta)$ kažemo da je konjugirana za izglednost $p(\mathcal{D} \mid \theta)$ ako je $p(\theta \mid D) \in \mathcal{F}$. Tada $p(\theta)$ zovemo konjugiranom apriornom distribucijom (eng. conjugate prior) za izglednost $p(\mathcal{D} \mid \theta)$.

Dirichletova distribucija je konjugirana apriorna distribucija multinomijalne distribucije. Ako neki primjer ima multinomijalnu distribuciju i apriorna distribucija njezinog parametra je Dirichletova, onda će i posteriorna distribucija tog parametra biti Dirchletova. Zapišimo to matematičkom notacijom

$$\begin{aligned}
\boldsymbol{\alpha} &= (\alpha_1, \ldots, \alpha_k), \\
\boldsymbol{p} \mid \boldsymbol{\alpha} &= (p_1, \ldots, p_k) \sim \text{Dirichlet}(k, \boldsymbol{\alpha}), \\
\mathcal{X} \mid p &= (x_1, \ldots, x_k) \sim \text{Multinomial}(k, \boldsymbol{p}), \\
\boldsymbol{c} &= (c_1, \ldots, c_k), \\
\boldsymbol{p} \mid \mathcal{X}, \boldsymbol{\alpha} &\sim \text{Dirichlet}(k, \boldsymbol{c} + \boldsymbol{\alpha}),
\end{aligned}$$

gdje je c_i broj primjera koji se nalaze u kategoriji i, za $i = 1, \ldots, k$. Uočimo da nam $\boldsymbol{\alpha}$ onda služi kao apriorni parametar, tzv. hiperparametar (eng. hyperparameter).

Izvedimo još jedno svojstvo Dirichletove distribucije, koje će nam kasnije biti korisno

$$p(\boldsymbol{p} \mid \mathcal{X}, \boldsymbol{\alpha}) = \text{Dirichlet}(\boldsymbol{p}; \boldsymbol{c} + \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{c} + \boldsymbol{\alpha})} \prod_{i=1}^{k} p_i^{c_i + \alpha_i - 1}.$$

Budući da je $p(\boldsymbol{p} \mid \mathcal{X}, \boldsymbol{\alpha})$ distribucija, integrirajući po svim $\boldsymbol{p}$ moramo dobiti 1. Slijedi

$$1 = \int \frac{1}{B(\boldsymbol{c} + \boldsymbol{\alpha})} \prod_{i=1}^{k} p_i^{c_i + \alpha_i - 1} d\boldsymbol{p} = \frac{1}{B(\boldsymbol{c} + \boldsymbol{\alpha})} \int \prod_{i=1}^{k} p_i^{c_i + \alpha_i - 1} d\boldsymbol{p}.$$

Dobili smo

$$\frac{1}{B(\boldsymbol{c} + \boldsymbol{\alpha})} \int \prod_{i=1}^{k} p_i^{c_i + \alpha_i - 1} d\boldsymbol{p} = B(\boldsymbol{c} + \boldsymbol{\alpha}). \tag{4.2}$$

4.2 Izvod modela

Koristit ćemo sljedeću reprezentaciju opaženih varijabli – riječi u dokumentima,

$$\mathcal{T} = \begin{pmatrix} \{t_1, \ldots, t_{n_{d_1}}\}, \\ \{t_{n_{d_1}+1}, \ldots, t_{n_{d_1}+n_{d_2}}\}, \\ \vdots \\ \{t_{\sum_{j=1}^{n-1} n_{d_j}+1}, \ldots, t_{\sum_{j=1}^{n} n_{d_j}}\} \end{pmatrix} = \begin{pmatrix} \{\text{riječi u 1. dokumentu}\}, \\ \{\text{riječi u 2. dokumentu}\}, \\ \vdots \\ \{\text{riječi u } n\text{-tom dokumentu}\} \end{pmatrix},$$

gdje je n, standardno, broj dokumenata, a n_{d_i} broj riječi u i-tom dokumentu, $i = 1, \ldots, n$. Skup latentnih varijabli, slično, reprezentiramo s

$$\mathcal{Z} = \begin{pmatrix} \{z_1, \ldots, z_{n_{d_1}}\}, \\ \{z_{n_{d_1}+1}, \ldots, z_{n_{d_1}+n_{d_2}}\}, \\ \vdots \\ \{z_{\sum_{j=1}^{n-1} n_{d_j}+1}, \ldots, z_{\sum_{j=1}^{n} n_{d_j}}\} \end{pmatrix} = \begin{pmatrix} \{\text{teme pridružene riječima u 1. dokumentu}\}, \\ \{\text{teme pridružene riječima u 2. dokumentu}\}, \\ \vdots \\ \{\text{teme pridružene riječima u } n\text{-tom dokumentu}\} \end{pmatrix}.$$

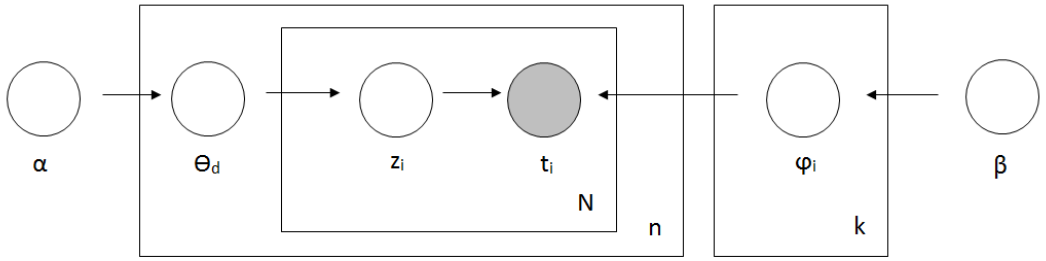
Dakle, $z_i \in \mathcal{Z}$ odgovara temi koja je pridružena riječi $t_i \in \mathcal{T}$. Koristimo sljedeće oznake

m ... broj riječi u kolekciji
n ... broj dokumenata u kolekciji
k ... broj različitih tema u kolekciji
$\hat{m}$ ... broj različitih riječi u kolekciji
$\mathcal{T} = \{t_i\}_{i=1}^m$ ... skup opaženih varijabli
$\mathcal{Z} = \{z_i\}_{i=1}^m$ ... skup latentnih varijabli
$\mathcal{T}^{\neg i} = \mathcal{T} \setminus \{t_i\}$ ... skup opaženih varijabli bez i-te riječi
$\mathcal{Z}^{\neg i} = \mathcal{Z} \setminus \{z_i\}$ ... skup latentnih varijabli bez i-te teme
$\alpha_i \in \mathbb{R}_+^k$ ... vektor apriornih težina za teme u dokumentu, za $i = 1, \ldots, n$
$\beta_i \in \mathbb{R}_+^{\hat{m}}$ ... vektor apriornih težina za riječi u temama, za $i = 1, \ldots, k$
$\Omega_{d,i}$ ... broj riječi iz d kojima je pridodana i-ta tema
Ω_d ... d-ti redak matrice $\Omega = (\Omega_{d,i})$
$\Psi_{i,v}$ ... broj riječi iz kolekcije kojima je pridodana i-ta tema
Ψ_i ... i-ti redak matrice $\Psi = (\Psi_{i,v})$
$\Omega_{d,j}^{\neg i}$ ... broj riječi, izuzev t_i, iz d kojima je pridodana j-ta tema
$\Psi_{j,v}^{\neg i}$ ... broj riječi iz kolekcije, izuzev t_i, kojima je pridodana j-ta tema
$\theta = \{\theta_{d,i}\}$ ... $\theta_{d,i} = p(z = i \mid d), \theta_d = p(z \mid d)$
$\varphi = \{\varphi_{i,v}\}$ ... $\varphi_{i,v} = p(t = v \mid z = i), \varphi_i = p(t \mid z = i)$

Standardno je uzeti jedan vektor α za sve dokumente i zahtijevati da su komponente tog vektora malene, npr. 0.001. Na taj način dobijemo rijetku distribuciju (njezina masa je koncentrirana u par točaka) i, implicitno, dvije ili tri glavne teme po dokumentu. To isto napravimo i za vektore apriornih težina za riječi u temama, β, odnosno, preferiramo par riječi za temu. Generativni proces za svaki dokument sada možemo zapisati na sljedeći način:

$$\begin{aligned}
\varphi_i \mid \beta &\sim \text{Dirichlet}(\beta), 1 \leq i \leq k\,,\\
\theta_d \mid \alpha &\sim \text{Dirichlet}(\alpha), 1 \leq d \leq n\,,\\
z_i \mid \theta_d &\sim \text{Multinomial}(\theta_d), 1 \leq z \leq n_d\,,\\
t_i \mid \varphi_{z_i} &\sim \text{Multinomial}(\varphi_{z_i}), 1 \leq t \leq n_d\,.
\end{aligned}$$

Pripadni grafički model ilustriran je Slikom 4.1.



Slika 4.1: Grafički model koji opisuje generativni proces u LDA. Unutarnji pravokutnik označava ponavljanje postupka N, $N = \sum_{j=1}^{n} n_{d_j}$, puta.

Tražimo latentnu, tematsku strukturu, odnosno $\mathcal{Z}, \theta$ i φ, uz dana opažanja $\mathcal{T}$. Dakle, trebamo preokreniti generativni proces i naučiti posterirorne distribucije latetnih varijabli u modelu s danim opaženim podacima. Taj proces je u literaturi na engleskom jeziku poznat pod imenom "posterior inference". U LDA to znači da trebamo izračunati

$$p(\mathcal{Z}, \theta, \varphi \mid \mathcal{T}, \alpha, \beta) = \frac{p(\mathcal{T}, \mathcal{Z}, \theta, \varphi \mid \alpha, \beta)}{p(\mathcal{T} \mid \alpha, \beta)}\,.$$

U nazivniku se nalazi marginalna distribucija opažanja, koju u teoriji računamo tako da prosumiramo zajedničke vjerojatnosti skupa opaženih primjera, uz svaku od mogućih latentnih, tematskih struktura. Međutim, broj takvih struktura je jako velik i zato se marginalna

distribucija mora aproksimirati. U tu svrhu postoje dvije osnovne kategorije algoritama, algoritmi bazirani na uzorkovanju (eng. sampling-based methods) i varijacijski algoritmi (eng. variational algorithms). Koji je pristup bolji, ovisi o modelu tema koji se koristi. U nastavku ćemo opisati jedan algoritam baziran na uzorkovanju, tzv. Gibbsovo uzorkovanje (eng. Gibbs sampling).

Gibbsovo uzorkovanje za LDA

Gibbsovo uzorkovanje je algoritam iz klase MCMC (eng. Markov Chain Monte Carlo) metoda za uzorkovanje. Koristimo ga kako bismo uzorkovali posteriornu distribuciju $p(\mathcal{Z}|\mathcal{T})$, uz dana opažanja $\mathcal{T}$. Kada nađemo vrijednosti latentnih varijabli iz skupa $\mathcal{Z}$, dobit ćemo i θ i φ.

Gibbsovim uzorkovanjem pronalazimo vrijednosti za $p(\mathcal{Z}|\mathcal{T})$ tako da iterativno računamo $p(z_i|\mathcal{Z}^{\neg i}, \mathcal{T})$, za $\mathcal{Z}^{\neg i} = \{z_j : j \neq i\}$. Gibbsovo uzorkovanje ne zahtjeva egzaktan izračun, već je dovoljno pronaći funkciju $f(\cdot)$ takvu da vrijedi

$$f(z_i \mid \mathcal{Z}^{\neg i}, \mathcal{T}) \propto p(z_i \mid \mathcal{Z}^{\neg i}, \mathcal{T}) .$$

U nastavku ćemo izvesti takvu funkciju f.

Prvo zapišimo zajedničku vjerojatnost u ovisnosti o α i β, a φ i θ ćemo prointegrirati,

$$\begin{aligned} p(\mathcal{Z}, \mathcal{T} \mid \alpha, \beta) &= p(\mathcal{T} \mid \mathcal{Z}, \beta) p(\mathcal{Z} \mid \alpha) , \\ p(\mathcal{T} \mid \mathcal{Z}, \beta) &= \int p(\mathcal{T} \mid \mathcal{Z}, \varphi) p(\varphi \mid \beta) d\varphi . \end{aligned} \tag{4.3}$$

Znamo da je $p(\varphi \mid \beta)$ Dirichletova distribucija, a $p(\mathcal{T} \mid \mathcal{Z}, \varphi)$ multinomijalna razdioba, pa iz njihovih definicija slijedi

$$p(\varphi \mid \beta) = \prod_{i=1}^{k} p(\varphi_i \mid \beta) = \prod_{i=1}^{k} \frac{1}{B(\beta)} \prod_{v=1}^{\hat{m}} \varphi_{i,v}^{\beta_v - 1} ,$$

$$p(\mathcal{T} \mid \mathcal{Z}, \varphi) = \prod_{j=1}^{m} \varphi_{z_j, t_j} = \prod_{i=1}^{k} \prod_{v=1}^{\hat{m}} \varphi_{i,v}^{\Psi_{i,v}} .$$

Uvrstimo posljednje dvije relacije u (4.3), pa slijedi

$$\begin{aligned} p(\mathcal{T} \mid \mathcal{Z}, \beta) &= \int \left(\prod_{i=1}^{k} \prod_{v=1}^{\hat{m}} \varphi_{i,v}^{\Psi_{i,v}} \right) \cdot \left(\prod_{i=1}^{k} \frac{1}{B(\beta)} \prod_{v=1}^{\hat{m}} \varphi_{i,v}^{\beta_v - 1} \right) d\varphi_i \\ &= \int \prod_{i=1}^{k} \frac{1}{B(\beta)} \prod_{v=1}^{\hat{m}} \varphi_{i,v}^{\Psi_{i,v} + \beta_v - 1} d\varphi_i \\ &= \prod_{i=1}^{k} \left(\int \frac{1}{B(\beta)} \prod_{v=1}^{\hat{m}} \varphi_{i,v}^{\Psi_{i,v} + \beta_v - 1} d\varphi_i \right). \end{aligned}$$

Kada iskoristimo jednakost (4.2) dobijemo

$$p(\mathcal{T} \mid \mathcal{Z}, \beta) = \prod_{i=1}^{k} \frac{B(\Psi_i + \beta)}{B(\beta)}.$$

Slično, izvedemo i formulu za $p(\mathcal{Z} \mid \alpha)$,

$$p(\mathcal{Z} \mid \alpha) = \int p(\mathcal{Z} \mid \theta) p(\theta \mid \alpha) d\theta = \prod_{d=1}^{n} \left(\int \frac{1}{B(\alpha)} \prod_{i=1}^{k} \theta_{d,i}^{\Omega_{d,i} + \alpha_i - 1} d\theta_d \right) = \prod_{d=1}^{n} \frac{B(\Omega_d + \alpha)}{B(\alpha)}.$$

Konačno, zajednička vjerojatnost postaje

$$p(\mathcal{Z}, \mathcal{T} \mid \alpha, \beta) = p(\mathcal{T} \mid \mathcal{Z}, \beta) p(\mathcal{Z} \mid \alpha) = \prod_{i=1}^{k} \frac{B(\Psi_i + \beta)}{B(\beta)} \cdot \prod_{d=1}^{n} \frac{B(\Omega_d + \alpha)}{B(\alpha)}. \tag{4.4}$$

Sada je sve spremno da izvedemo funkciju f, proporcionalnu s

$$p(z_i = j \mid \mathcal{Z}^{\neg i}, \mathcal{T}^{\neg i}, \alpha, \beta) = \frac{p(\overbrace{z_i = j, \mathcal{Z}^{\neg i}}^{=\mathcal{Z}}, \mathcal{T} \mid \alpha, \beta)}{p(\mathcal{Z}^{\neg i}, \mathcal{T} \mid \alpha, \beta)}.$$

Budući da samo z_i ovisi o t_i, nazivnik možemo rastaviti kao

$$p(\mathcal{Z}^{\neg i}, \mathcal{T} \mid \alpha, \beta) = p(\mathcal{Z}^{\neg i}, \mathcal{T}^{\neg i} \mid \alpha, \beta) p(\mathcal{Z}^{\neg i}, t_i \mid \alpha, \beta),$$

pa slijedi

$$p(z_i = j \mid \mathcal{Z}^{\neg i}, \mathcal{T}^{\neg i}, \alpha, \beta) \propto \frac{p(\mathcal{Z}, \mathcal{T} \mid \alpha, \beta)}{p(\mathcal{Z}^{\neg i}, \mathcal{T}^{\neg i} \mid \alpha, \beta)}.$$

Kada uvrstimo jednakost (4.4), uz $z_i = j$, dobijemo

$$p(z_i = j \mid \mathcal{Z}^{\neg i}, \mathcal{T}^{\neg i}, \alpha, \beta) \propto \frac{B(\Psi_j + \beta)}{B(\Psi_j^{\neg i} + \beta)} \cdot \frac{B(\Omega_d + \alpha)}{B(\Omega_d^{\neg i} + \alpha)}, \tag{4.5}$$

gdje d označava dokument koji sadrži riječ t_i. Prethodnu formulu možemo pojednostaviti ako iskoristimo reprezentaciju beta funkcije pomoću gama funkcije, (4.1). U nastavku, izvodimo takvo pojednostavljenje za (4.5).

Prvo, istaknimo svojstvo $\Psi_{j,v}$, odnosno, $\Omega_{d,j}$ koje će biti korisno. Vrijedi

$$\begin{aligned} \Psi_{j,v} &= \begin{cases} \Psi_{j,v}^{\neg i} + 1 & v = t_i \wedge z_i = j\,, \\ \Psi_{j,v}^{\neg i} & \text{inače}\,, \end{cases} \\ \Omega_{d,j} &= \begin{cases} \Omega_{d,j}^{\neg i} + 1 & d = d_i \wedge z_i = j\,, \\ \Omega_{d,j}^{\neg i} & \text{inače}\,. \end{cases} \end{aligned} \tag{4.6}$$

Uvrstimo u jednakost (4.5) definiciju beta funkcije (4.1),

$$\begin{aligned} p(z_i = j \mid \mathcal{Z}^{\neg i}, t_i = \hat{t}, \mathcal{T}^{\neg i}, \alpha, \beta) = {} & \frac{\prod_{t=1}^{\hat{m}} \Gamma(\Psi_{j,t} + \beta_t)}{\Gamma\Big(\sum_{t=1}^{\hat{m}} (\Psi_{j,t} + \beta_t)\Big)} \cdot \frac{1}{\dfrac{\prod_{t=1}^{\hat{m}} \Gamma(\Psi_{j,t}^{\neg i} + \beta_t)}{\Gamma\Big(\sum_{t=1}^{\hat{m}} (\Psi_{j,t}^{\neg i} + \beta_t)\Big)}} \\ & \cdot \frac{\prod_{z=1}^{k} \Gamma(\Omega_{d,z} + \alpha_z)}{\Gamma\Big(\sum_{z=1}^{k} (\Omega_{d,z} + \alpha_z)\Big)} \cdot \frac{1}{\dfrac{\prod_{z=1}^{k} \Gamma(\Omega_{d,z}^{\neg i} + \alpha_z)}{\Gamma\Big(\sum_{z=1}^{k} (\Omega_{d,z}^{\neg i} + \alpha_z)\Big)}}\,. \end{aligned} \tag{4.7}$$

Koristeći formule (4.6) i činjenicu $\Gamma(x+1) = x \cdot \Gamma(x)$, prvi faktor u (4.7) možemo raspisati

$$\begin{aligned} \frac{\prod_{t=1}^{\hat{m}} \Gamma(\Psi_{j,t} + \beta_t)}{\Gamma\Big(\sum_{t=1}^{\hat{m}} (\Psi_{j,t} + \beta_t)\Big)} &= \frac{\prod_{\substack{t=1 \\ t \neq \hat{t}}}^{\hat{m}} \Gamma(\Psi_{j,t} + \beta_t) \cdot \Gamma(\Psi_{j,\hat{t}} + \beta_{\hat{t}})}{\Gamma\Big(\sum_{\substack{t=1 \\ t \neq \hat{t}}}^{\hat{m}} (\Psi_{j,t} + \beta_t) + \Psi_{j,\hat{t}} + \beta_{\hat{t}}\Big)} = \frac{\prod_{\substack{t=1 \\ t \neq \hat{t}}}^{\hat{m}} \Gamma(\Psi_{j,t}^{\neg i} + \beta_t) \cdot \Gamma(\Psi_{j,\hat{t}}^{\neg i} + 1 + \beta_{\hat{t}})}{\Gamma\Big(\sum_{\substack{t=1 \\ t \neq \hat{t}}}^{\hat{m}} (\Psi_{j,t}^{\neg i} + \beta_t) + \Psi_{j,\hat{t}}^{\neg i} + 1 + \beta_{\hat{t}}\Big)} \\ &= \frac{\prod_{\substack{t=1 \\ t \neq \hat{t}}}^{\hat{m}} \Gamma(\Psi_{j,t}^{\neg i} + \beta_t) \cdot \Gamma(\Psi_{j,\hat{t}}^{\neg i} + \beta_{\hat{t}}) \cdot (\Psi_{j,\hat{t}}^{\neg i} + \beta_{\hat{t}})}{\Gamma\Big(\sum_{t=1}^{\hat{m}} (\Psi_{j,t}^{\neg i} + \beta_t) + 1\Big)} \\ &= \frac{\prod_{t=1}^{\hat{m}} \Gamma(\Psi_{j,t}^{\neg i} + \beta_t) \cdot (\Psi_{j,\hat{t}} + \beta_{\hat{t}} - 1)}{\Gamma\Big(\sum_{t=1}^{\hat{m}} (\Psi_{j,t}^{\neg i} + \beta_t)\Big) \cdot \Big(\sum_{t=1}^{\hat{m}} (\Psi_{j,t}^{\neg i} + \beta_t)\Big)} \\ &= \frac{\prod_{t=1}^{\hat{m}} \Gamma(\Psi_{j,t}^{\neg i} + \beta_t)}{\Gamma\Big(\sum_{t=1}^{\hat{m}} (\Psi_{j,t}^{\neg i} + \beta_t)\Big)} \cdot \frac{\Psi_{j,\hat{t}} + \beta_{\hat{t}} - 1}{\sum_{t=1}^{\hat{m}} (\Psi_{j,t} + \beta_t) - 1}\,. \end{aligned}$$

Uočimo da je prvi faktor u zadnjoj jednakosti jednak recipročnoj vrijednosti drugog faktora iz (4.7). Slično se raspiše i treći faktor u (4.7), pa konačno imamo

$$p(z_i = j \mid \mathcal{Z}^{\neg i}, t_i = \hat{t}, \mathcal{T}^{\neg i}, \alpha, \beta) = \frac{\Psi_{j,\hat{t}} + \beta_{\hat{t}} - 1}{\sum_{t=1}^{\hat{m}} (\Psi_{j,t} + \beta_t) - 1} \cdot \frac{\Omega_{d,j} + \alpha_j - 1}{\sum_{z=1}^{k} (\Omega_{d,z} + \alpha_z) - 1}\,.$$

Budući da nazivnik drugog faktora ne ovisi o z_i, a tražimo $f(z_i) \propto p(z_i)$, za $z_i = j$ vrijedi

$$f(z_i) = p(z_i = j \mid \mathcal{Z}^{\neg i}, t_i = v, \mathcal{T}^{\neg i}, \alpha, \beta) \propto \frac{\Psi_{j,v} + \beta_{t_i} - 1}{\sum_{v=1}^{\hat{m}} (\Psi_{j,v} + \beta_v) - 1} (\Omega_{d,j} + \alpha_j - 1) .$$

Dobili smo traženu funkciju f s kojom možemo uzorkovati posteriorne vjerojatnosti latentnih varijabli.

Iz definicija $\theta_{j,t}$ i $\varphi_{d,j}$ slijedi,

$$\theta_{j,t} = \frac{\Psi_{j,t} + \beta_t}{\sum_{\hat{v}=1}^{\hat{m}} (\Psi_{j,\hat{v}} + \beta_{\hat{v}})} ,$$
$$\varphi_{d,j} = \frac{\Omega_{d,j} + \alpha_j}{\sum_{z=1}^{k} (\Omega_{d,z} + \alpha_z)} .$$

Time smo u potpunosti definirali traženu latentnu, tematsku strukturu.

Kao mjeru sličnosti dvaju dokumenata možemo opet iskoristiti Fisherove jezgre ili simetričnu verziju KL-divergencije

$$\mathrm{KL}_\alpha(d, q) = \alpha \cdot \mathrm{KL}(p, \alpha p + (1 - \alpha) q) + (1 - \alpha) \cdot \mathrm{KL}(q, \alpha q + (1 - \alpha) p) .$$

Za $\alpha = \frac{1}{2}$ dobijemo tzv. Jensen-Shannonovu divergenciju.

Poglavlje 5

Primjeri

U ovom poglavlju primijenit ćemo opisane modele na vrlo jednostavne, ilustrativne primjere. Koristit ćemo programski jezik Python, koji je danas jedan od najčešće korištenih programskih jezika u području strojnog učenja (eng. machine learning) i obrade prirodnog jezika (eng. natural language processing). Razlog tome je razvoj mnogobrojnih biblioteka i ekstenzija koje olakšavaju rad u tim područjima. Na primjer, za rad s opisanim latentnim modelima koristan je programski alat Gensim, koji koristi NumPy, SciPy i Cython, te tako rezultira jako brzim i efikasnim algoritmima. Gensim ima odlično napisanu dokumentaciju s puno primjera koja se može naći na adresi `https://radimrehurek.com/gensim/tutorial.html`.

Prvo ćemo pokazati osnovne pojmove iz Poglavalja 1 na primjeru iz [5]. Zatim ćemo reprezentirati modeliranje tema, koristeći Gensim, na malom dijelu Wikipedije. Veličina cijele Wikipedije ne predstavalja problem za Gensim. Međutim, ne želimo prikazati njegovu efikasnost na velikoj bazi podataka, već samo demonstrirati modele. Koristit ćemo samo LSA i LDA, jer se pLSA, uz parametre $\alpha, \eta = 1$, može dobiti iz LDA [7]. Međutim, treba imati na umu da LDA nije uvijek superiornija metoda od pLSA.

5.1 Model vektorskog prostora

Uzet ćemo skup dokumenata kojeg su uzeli autori latentne semantičke analize u [5]. Pomoću funkcije `nltk.FreqDist` i bibiloteke pandas, dobit ćemo pojmovno-dokumentnu matricu u kojoj umjesto, *tf-idf*, težina imamo broj pojavljivanja riječi u dokumentu (Slika 5.1). Funkcijom `TfidfVectorizer` iz klase `sklearn.feature_extraction.text` u biblioteci sklearn, možemo dobiti pojmovnu-dokumentnu matricu iz Poglavlja 1.

	a	abc	and	applications	binary		trees	unordered	user	well	widths
1	0	1	0	1	0		0	0	0	0	0
2	1	0	0	0	0		0	0	1	0	0
3	0	0	0	0	0		0	0	1	0	0
4	0	0	1	0	0		0	0	0	0	0
5	0	0	0	0	0	...	0	0	1	0	0
6	0	0	0	0	1		1	1	0	0	0
7	0	0	0	0	0		1	0	0	0	0
8	0	0	1	0	0		1	0	0	1	1
9	1	0	0	0	0		0	0	0	0	0

Slika 5.1: Pojmovno-dokumentna matrica u kojoj za svaku riječ piše broj njezinog pojavljivanja u svakom od devet dokumenata u kolekciji

Kolekciju dokumenata, gdje je svaki dokument jedna rečenica, u Pythonu-u definiramo na ovaj način

```
In [1]: documents = ["Human machine interface for lab abc computer applications",
                    "A survey of user opinion of computer system response time",
                    "The EPS user interface management system",
                    "System and human system engineering testing of EPS",
                    "Relation of user perceived response time to error measurement",
                    "The generation of random binary unordered trees",
                    "The intersection graph of paths in trees",
                    "Graph minors IV Widths of trees and well quasi ordering",
                    "Graph minors A survey"]
```

Uočimo da u kolekciji imamo dvije grupe dokumenata, jedni govore o interakciji korisnika i računala, dok drugi govore o grafovima.

Funkcijom `cosine_similarity` iz bibiloteke sklearn izračunamo matricu sličnosti dokumenata (eng. similarity matrix). Rezultat vidimo na Slici 5.2. Udaljenost dvaju dokumenata definiramo s

$$\text{dist}(d_1, d_2) = 1 - sim_{\cos}(d_1, d_2)\,.$$

Broj značajki je jednak broju riječi u kolekciji, dakle, broj značajki je 42. Želimo prikazati vektore (dokumente) iz prostora dimenzije 42, kao točke u dvodimenzionalnom prostoru. Algoritam koji to omogućava zove se multidimenzionalno skaliranje (eng. multidimensional scaling) i njegova implementacija se nalazi u biblioteci sklearn. Dobivenu vizualizaciju možemo viditi na Slici 5.3. Kao što je očekivano, vidimo dvije grupe koje su linearno odvojive.

	1	2	3	4	5	6	7	8	9
1	1	0.1088699516	0.132747163	0.1041744609	0	0	0	0	0
2	0.1088699516	1	0.2187061977	0.2576567856	0.3799957488	0.0953230213	0.0997138829	0.0790608848	0.2074740264
3	0.132747163	0.2187061977	1	0.3476856191	0.0996847177	0.1159461178	0.1212869404	0	0
4	0.1041744609	0.2576567856	0.3476856191	1	0.0392096898	0.0456059005	0.0477066437	0.1376520006	0
5	0	0.3799957488	0.0996847177	0.0392096898	1	0.04344755	0.0454488733	0.0360353848	0
6	0	0.0953230213	0.1159461178	0.0456059005	0.04344755	1	0.2637994683	0.1255372072	0
7	0	0.0997138829	0.1212869404	0.0477066437	0.0454488733	0.2637994683	1	0.2187951985	0.1739630516
8	0	0.0790608848	0	0.1376520006	0.0360353848	0.1255372072	0.2187951985	1	0.3203850936
9	0	0.2074740264	0	0	0	0	0.1739630516	0.3203850936	1

Slika 5.2: Matrica sličnosti za dokumente u kolekciji

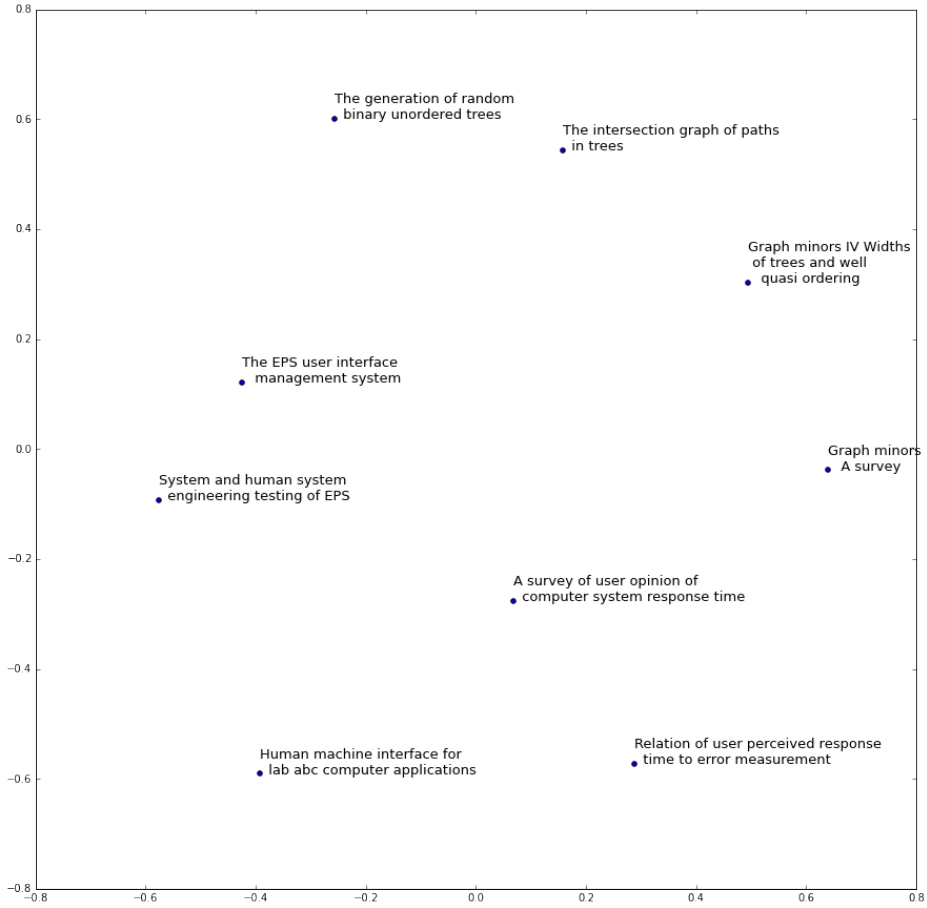
5.2 Modeliranje tema

Predprocesiranje

Koristeći funkcije iz klase `gensim.corpora.wikicorpus` možemo izravno učitati tekst članaka, bez raspakiravanja baze. Prije primjene modela moramo očistiti podatke, odnosno, moramo napraviti predprocesiranje podataka (eng. data preprocessing). Učitavajući tekst iz baze, odmah zahtijevamo da riječ nije tzv. stop-riječ (eng. stop-word). To su riječi koje se učestalo pojavljuju u jeziku ili služe tome da rečenice budu gramatički korektne. U klasi `gensim.parsing.preprocessing` se nalazi skup stop-riječi za engleski jezik. Prvih deset su: ”all”, ”six”, ”just”, ”less”, ”being”, ”indeed”, ”over”, ”move”, ”anyway”, ”four”. Osim toga, koristeći `gensim.utils.simple_preprocess` velika slova pretvaramo u mala i korjenujemo (eng. stem) – svodimo riječi na njihov osnovni oblik. Kasnije je uočeno da korjenovanje pomoću te funkcije nije zadovoljavajuće, pa koristimo klasu `nltk.stem.snowball` iz biblioteke nltk. Većina algoritama radi s numeričkim vrijednostima, pa svakoj riječi moramo pridružiti jedinstveni broj. To preslikavanje zovemo rječnik (eng. dictionary) i definirano je klasom `gensim.corpora.dictionary.Dictionary`. Funkcijom `filter_extremes` iz te klase odstranimo riječi koje se pojavljuju u premalo ili previše dokumenata. Sada smo spremni napraviti reprezentaciju kolekcije ”vrećom riječi” (eng. bag of words represetation). U Gensim-u to činimo pomoću funkcije `doc2bow` koja se isto nalazi u klasi `gensim.corpora.dictionary.Dictionary`.

Primjena modela

Na dijelu baze treniramo LDA i LSA model pomoću klasa `gensim.models.LdaModel` i `gensim.models.LsiModel`. Za LSA smo prethodno napravili transformaciju u *tf-idf* shemu funkcijom `gensim.models.TfidfModel`. Teme koje su modeli pronašli vizualizi-

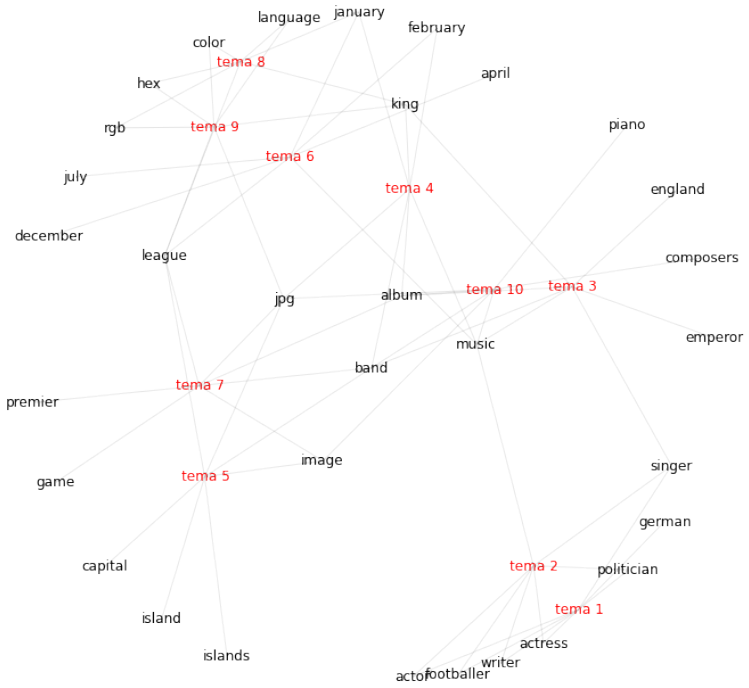


Slika 5.3: Vizualizacija dokumenata u dvodimenzionalnom prostoru

ramo grafom pomoću bibloteke networkx (Slika 5.5 i 5.4). Spomenute klase za treniranje modela omogućavaju trivijalno transformiranje novih dokumenata u novonastale prostore.

Evaluacija modela

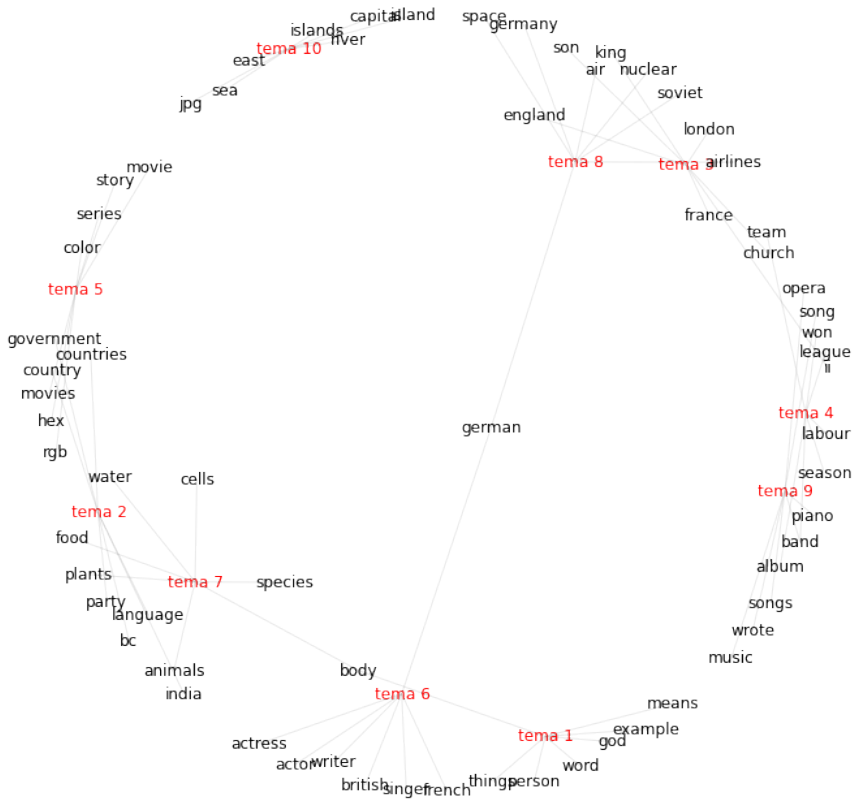
Jedan mogući način evaluacije predložen je u [4] pod nazivom umetanje riječi (eng. word intrusion). Evaluiranje se provodi tako da se od najvažanijih riječi, što su u LDA one s najvećom vjerojatnošću, nasumično odabere jedna i zamijeni s riječju koja se ne nalazi u toj temi, odnosno, čija je vjerojatnost jednaka nula. Zatim se teme sa zamijenjenim riječima prezentiraju osobi koja mora pogoditi koja riječ je zamijenjena.



Slika 5.4: Teme pronađene na dijelu Wikipedije pomoću LSA

Dakle, nakon što smo dobili teme s uzorka članaka s Wikipedije i zamijenili jednu riječ iz svake teme, tražimo odgovor na sljedeće pitanje:

```
Da li možete uočiti riječ koja je promijenjena?
1: person example bc means god word body numbers human theory
2: language government countries country bc roman party px empire languages
3: king england church france ii son movies roman st death
4: labour album league team season food band player football released
5: rgb hex movie color series story movies light film album
6: actor german french british actress writer islands politician footballer president
7: water species animals food plants body hex jpg chemical common
8: air airlines soviet germany space german nuclear means military car
9: music wrote band song songs opera piano rock play london
10: island river islands east color sea jpg largest population region
```



Slika 5.5: Teme pronađene na dijelu Wikipedije pomoću LDA

Riječi koje su zaista promijenjene spremljene su u posebnu listu. Ispišimo ju,

```
Stvarne zamjene su:
[(0, (u'things', u'bc')), (1, (u'india', u'roman')),
(2, (u'london', u'movies')),(3, (u'won', u'food')),
(4, (u'red', u'album')), (5, (u'singer', u'islands')),
(6, (u'cells', u'hex')), (7, (u'plane', u'means')),
(8, (u'popular', u'london')),(9, (u'capital', u'color'))]
```

Međutim, ovaj način je zasnovan na odokativnoj procjeni čovjeka. Tvorci Gensim alata iz tog su razloga predložili drugi način evaluacije. Oni su sve dokumente iz skupa za testiranje podijelili na dva dijela i svaki zasebno transformirali u LDA, odnosno, LSA prostor. Zatim su računali koliko su slični u tom prostoru. Srednja vrijednost tako dobivenih sličnosti mora biti što veća. Osim toga, uspoređivali su dijelove dokumenta s proizvoljnim dijelovima ostalih dokumenata. Srednja vrijednost tako dobivenih sličnosti mora biti što manja. Dobivene srednje vrijednosti sličnosti među dijelovima istog dokumenta i srednje vrijednosti sličnosti proizvoljnih parova dijelova, jednake su redom:

```
LDA
0.819554506799
0.256792083816
LSI
0.84162555348
0.441589808629
```

Srednja vrijednost sličnosti dijelova istog dokumenta je neznatno veća kod LSA, ali je srednja vrijednost sličnosti proizvoljnih parova za LSA značajnije veća nego kod LDA.

Pretraživanje najrelevantnijih dokumenata za dani upit

Problem pronalaska najrelevantnijih dokumenata za neki upit sveli smo na problem pronalaska najsličnijih dokumenata u novim prostorima. Gensim omogućava lako rješavanje tog problema. Potrebno je samo transformirati upit u novonastali prostor i iskoristiti klasu `gensim.similarities.Similarity`. Za sljedeći upit,

```
In [52]: query = "April is the fourth month of the year, and comes between March \
         and May. It has 30 days. April begins on the same day of week as July in \
         all years and also January in leap years."
```

tražili smo pet najsličnijih dokumenata u LSA prostoru. Ispišimo naslove tih članka i par riječi koje se nalaze u njima

```
December [u'december', u'twelfth', u'month', u'year', u'gregorian', u'calendar']
February [u'february', u'second', u'month', u'year', u'coming', u'january']
Leap year [u'leap', u'year', u'comes', u'years', u'year', u'extra']
November [u'november', u'eleventh', u'penultimate', u'second', u'month', u'year']
New Year's Day [u'new', u'year', u'day', u'holiday', u'countries', u'created']
```

Uočimo da smo dobili dokumente koji su vezani za dani upit, te smo zadovoljni kako je LSA riješila problem pronalaska najrelevantnijih dokumenata za ovaj upit.

Bibliografija

[1] M. W. Berry, Z. Drmač i E. R. Jessup, *Matrices, vector spaces, and information retrieval*, SIAM Review **41** (1999), br. 2, str. 335–362.

[2] M. W. Berry, S. T. Dumais i G. W. O'Brien, *Using linear algebra for intelligent information retrieval*, SIAM Review **37** (1995), br. 4, str. 573–595.

[3] D. M. Blei, A. Y. Ng i M. I. Jordan, *Latent dirichlet allocation*, the Journal of machine Learning research **3** (2003), str. 993–1022.

[4] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish i D. M. Blei, *Reading Tea Leaves: How Humans Interpret Topic Models*, Neural Information Processing Systems, 2009.

[5] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas i R. Harshman, *Indexing by latent semantic analysis*, JAsIs **41** (1990), br. 6, str. 391–407.

[6] I. S. Duff, R. G. Grimes i J. G. Lewis, *The Rutherford–Boeing sparse matrix collection*, Teh. izv., Council for the Central Laboratory of the Research Councils, Chilton, England, 1997.

[7] M. Girolami i A. Kabán, *On an equivalence between PLSI and LDA*, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM, 2003, str. 433–434.

[8] G. H. Golub i C. V. Loan, *Matrix Computations*, Johns Hopkins University Press, 1989.

[9] T. Hofmann, *Probabilistic latent semantic indexing*, Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 1999, str. 50–57.

[10] ———, *Learning the similarity of documents: An information-geometric approach to document retrieval and categorization*, Advances in Neural Information Processing Systems **12** (2000), str. 914–920.

[11] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards **45** (1950), str. 255–182.

[12] R. B. Lehoucq i D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM Journal on Matrix Analysis and Applications **17** (1996), br. 4, str. 789–821.

[13] T. Letsche i M. Berry, *Large-scale information retrieval with latent semantic indexing*, Information sciences **100** (1997), br. 1, str. 105–137.

[14] C. D. Manning, P. Raghavan i H. Schütze, *Introduction to information retrieval*, sv. 1, Cambridge university press, Cambridge, 2008.

[15] L. Mirsky, *Symmetric gauge functions and unitarily invariant norms*, The Quarterly Journal of Mathematics **11** (1960), br. 1, str. 50–59.

[16] M. Nyffenegger, J. C. Chappelier i E. Gaussier, *Revisiting Fisher Kernels for Document Similarities*, Machine Learning: ECML 2006 (Proc. of 17th European Conference on Machine Learning), Lecture Notes in Computer Science, sv. 4212, Springer, 2006, str. 727–734.

[17] H. Rutishauser, *Simultaneous iteration method for symmetric matrices*, Numerische Mathematik **16** (1970), br. 3, str. 205–223.

[18] A. H. Sameh i J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM Journal on Numerical Analysis **19** (1982), br. 6, str. 1243–1259.

[19] M. Welling, C. Chemudugunta i N. Sutter, *Deterministic Latent Variable Models and Their Pitfalls.*, In: SIAM conference of Data Mining SDM 2008, SIAM, 2008, str. 196–207.

Sažetak

U današnje vrijeme sve više težimo tome da omogućimo da računalo izvršava zadatke, koje čovjek čini rutinski, jednako brzo i efikasno. Jedan od takvih zadataka je i pronalazak par dokumenata iz kolekcije koji su najrelevantniji za korisnikov upit.

Prvi korak u rješavanju tog problema je reprezentacija kolekcije dokumenata pojmovno-dokumentnom matricom, čiji elementi predstavljaju *tf-idf* težine riječi u dokumentu. Na taj način smo svaki dokument prikazali vektorom u prostoru pojmova. Ako i upit prikažemo vektorom, onda za usporedbu upita i dokumenta iz kolekcije, možemo iskoristiti standardne mjere sličnosti, poput kosinusne. U takvom prostoru, sinonimi će biti ortogonalni, a višeznačnice će biti predstavljene jednim vektorom, neovisno o kontekstu u kojem se riječ nalazi.

Motivirani tom činjenicom i velikom dimenzijom pojmovno-dokumentne matrice, odlučili smo ju aproksimirati matricom nižeg ranga. Aproksimaciju je omogućila singularna dekompozicija matrice (SVD). Pokazali smo da aproksimacijom uzimamo u obzir kontekst u kojem se riječ nalazi. Kako bismo korisnikov upit mogli usporediti s vektorima dokumenata u novonastalom prostoru i njega transformiramo. Pokazali smo kako u slučaju dinamičke kolekcije možemo dodati nove dokumente i pojmove u već postojeći latentni prostor.

Iako je opisana metoda, koju kraće zovemo LSA, donekle riješila problem sinonima, preostao je problem s višeznačnicama. Osim toga, LSA pretpostavlja da šum uzorka podataka (dobiven zbog jezične varijabilnosti) ima Gaussovu distribuciju, što nije prirodna pretpostavka. Sljedećom metodom, pLSA, pretpostavili smo da svaki dokument dolazi iz nekog generativnog, vjerojatnosnog procasa čije parametre tražimo maksimizacijom izglednosti. Svaki dokument je mješavina latentnih koncepta i tražimo posteriorne vjerojatnosti tih koncepta uz dana opažanja. Međutim, pLSA ih shvaća kao parametar modela, što dovodi do prenaučenosti.

Zato smo prezentirali još jedan model, LDA, koji te vjerojatnosti tretira kao distribuciju koja ovisi o nekom parametru. Kao i pLSA, i LDA reprezentira dokumente kao mješavinu latentnih tema, ali teme su sada distribucije riječi iz rječnika. Zato je bilo potrebno definirati neku distribuciju distribucija, gdje se prirodno nametnula Dircheltova distribucija.

Na kraju smo ukratko prikazali modeliranje tema na kolekciji članaka iz Wikipedije.

Summary

Nowadays, more and more important is to make a computer that performs tasks that man does routinely, as fast and efficiently. One of these tasks is finding a few documents from the given collection, that are most relevant for user's query.

The first step in solving this problem is representing the collection of documents as a term-document matrix, whose elements are *tf-idf* weights of words in the document. In this way, we represent each document as a vector in the space of terms. If the query is represented as a vector as well, standard similarity measures, such as a cosine similarity, can be used for comparison of the query and documents. In such space, synonyms will be orthogonal and polysemies will be presented with one vector, regardless of the context of the word.

Motivated by this fact, and a large dimension of the term-document matrix, a lower rank approximation of the matrix is done. The approximation is gained using a singular value decomposition (SVD) of the matrix. We have shown that the approximation takes into account the context of the words. The query needs to be transformed into a new space as well, so it can be compared with vectors in this lower dimensional space. We showed how can we add new documents and terms in the case of a dynamic collection.

While this method, solves the problem of synonyms to some extent, the problem with polysemies remains unsolved. In addition, LSA assumes that the data noise (gained from language variability) has a Gaussian distribution, which is not a natural assumption. The following method, pLSA, assumes that each document comes from a generative, probabilistic process, whose parameters we seek with maximization of likelihood. Each document is a mixture of latent concepts and we look for posterior probabilities of these concepts when observations are given. However, pLSA assumes these probabilities are parameters of model which leads to over-fitting of the model.

Therefore, we present another model, LDA, that treats these probabilities as a distribution that depends on some parameter. Documents are, again, represented as a mixture of latent topics, but these topics are a distribution of words from the dictionary. Therefore, it is necessary to define a distribution of distributions and a natural choice is the Dirichelt distribution.

Finally, we have briefly presented a topic modeling of the collection of articles from

Wikipedia.

Životopis

Rođena sam 5. 8. 1991. u Splitu. Odrasla sam u Omišu, gdje sam stekla početno obrazovanje u Osnovnoj školi Josipa Pupačića. Po završetku osnovne škole upisala sam Treću gimnaziju u Splitu, gdje sam se zainteresirala za matematiku. Maturirala sam 2010. godine i zatim sam upisala preddiplomski studij na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu. Godine 2013. završila sam prediplomski studij i upisala diplomski studij Primijenjena matematika na istom fakultetu.

Dvije godine tokom studiranja bila sam demonstratorica u praktikumu na fakultetu, gdje sam pomagala studentima i održavala red. Na ljeto 2014. godine odradila sam ljetnu, studentsku praksu u hrvatskoj tvrtki Poslovna Inteligencija, gdje sam se bavila rudarenjem podataka. Zadnje tri godine učim ruski i njemački jezik u školi stranih jezika Sputnik u Zagrebu.