

**Eötvös Loránd Tudományegyetem**  
**Társadalomtudományi Kar**  
Mesterképzés

**Látens topikok és látható címkék**  
***Az Author-Topic Model gyakorlati alkalmazása***  
***a korrupció témáiban***

Katona Eszter Rita  
Z7DY7N  
Survey Statisztika MSc.

Témavezető:  
Kmetty Zoltán PhD.

2018. április

*Szeretném megköszönni Varjú Zoltánnak szakdolgozatom megírása során nyújtott segítségét. Dolgozatom elkészítésének minden fázisában rengeteget tanultam tőle, szakmai meglátásai, tanácsai nélkül nem tudtam volna elkészíteni a dolgozatomat. Emellett bizalmával, támogatásával is segítette a munkám.*

*Köszönöm Ilyés Virágnak javaslatait, valamint a tartalmi és formai hibáim javítását.*

*Köszönöm a Precognoxnak és a K-Monitornak, hogy lehetőséget kaptam dolgozatom megírására.*

## Tartalomjegyzék

<b>I. BEVEZETÉS</b>	<b>5</b>
1. 1. Motiváció	5
1. 2. A dolgozat felépítése	6
<b>II. ELMÉLETI KERET</b>	<b>7</b>
1. A topikmodellezés alapfogalmai	10
1. 1. Vekortérmodell	10
1. 2. A Markov-lánc Monte Carlo módszer, és a Gibbs mintavételezés	12
1. 3. Variational Bayes	14
2. Topikmodellek, az Author Model és a látens Dirichlet allokáció	16
3. Az Author-Topic Model	19
4. Futásidő, skálázhatóság	23
<b>III. ESETTANULMÁNY</b>	<b>24</b>
1. Az elemzés során használt eszközök bemutatása	24
1. 1. A fejlesztési környezet	24
1. 2. SQL	25
1. 3. Python	25
1. 4. Gensim	26
1. 5. Az adatvizualizációhoz használt eszközök	27
2. Az adatok előkészítése	28
2. 1. Az adatok forrása: K-Monitor Akták	28
2. 2. Az adatok bemutatása	30
2. 3. Cikkek legyűjtése	36
2. 4. Adatfeldolgozás, adattisztítás	36
2. 4. 1. Named Entity Recognition	37
2. 4. 2. Szótövezés, szófaj szerinti szűrés	39
2. 4. 3. Szignifikáns bigramok	40
2. 4. 4. A tartalomszavak összeállítása	41
3. Az Akták cikkeinek elemzése az Author-Topic Modellel	43
3.1. A korpusz előállítása	43
3.2. A modell tanítása	44
3. 3. Topikok	46
3. 3. 1. Az optimális topikszám keresése	47
3. 3. 2. Topikok elnevezése	51
4. Az előfeldolgozás fázisainak összehasonlítása	55
<b>IV. ÖSSZEGZÉS</b>	<b>57</b>
<b>V. Irodalomjegyzék</b>	<b>59</b>
<b>VI. Melléklet</b>	<b>64</b>
<b>VII. Python kód</b>	
Branch: db-extractor	Hiba! A könyvjelző nincs definiálva.
db-extractor.py	<b>Hiba! A könyvjelző nincs definiálva.</b>
Branch: data-collector	Hiba! A könyvjelző nincs definiálva.
get_articles.py	<b>Hiba! A könyvjelző nincs definiálva.</b>
inspect_articles.py	Hiba! A könyvjelző nincs definiálva.
Branch: process_rawtext	<b>Hiba! A könyvjelző nincs definiálva.</b>
docfreq_wfrec_contentw.py	Hiba! A könyvjelző nincs definiálva.
get_all_entities.py	<b>Hiba! A könyvjelző nincs definiálva.</b>
process_rawtext.py	Hiba! A könyvjelző nincs definiálva.

*Branch: filter\_tags*

filter\_tags.py

filtered\_desc.py

*Branch: model-todos*

prepare\_corpus\_gensim.py

atm\_models.py

atm\_eval.py

atm\_eval2.py

*Branch: topic\_info*

topic\_info.py

sort\_docs.py

termite.py

graph.py

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

***Hiba! A könyvjelző nincs definiálva.***

## I. BEVEZETÉS

### 1. 1. Motiváció

Szakedolgozatomban a K-Monitor cikkgyűjteményéből származó dokumentumokon szeretném bemutatni a szerző-topikmodell (*Author-Topic Model*<sup>1</sup>) használatát, és ezáltal betekintést nyújtani a szöveganalitikai elemzés legfontosabb lépéseibe.

Dolgozatom módszertani jellegű, bár a korrupció napjainkban aktuális és forró téma, most nem a korrupció vizsgálatára fókuszálok, hanem a modell használhatóságáról és módszertanáról írok<sup>2</sup>.

Dolgozatomat öt célkitűzés mentén építettem fel. Egyrészt szeretném bemutatni, hogy milyen lépésekre van szükség, ha szöveges adatok elemzésébe kezdünk, ehhez kapcsolódó célkitűzésem a modell alkalmazhatóságának vizsgálata különböző mértékben előfeldolgozott szövegeken.

Author-Topic Modelt akkor használhatunk, ha sok dokumentum áll rendelkezésünkre szerzőkkel (metaadatokkal), és szeretnénk felderíteni, hogy az egyes szerzők milyen témákkal járnak együtt. A Pythonból elérhető Gensim csomagban – amit a topikmodellezés során használok – akadnak technikai nehézségek, dolgozatomban ezek áthidalására is szeretnék megoldási alternatívát kínálni. Egyfelől szeretném megmutatni, hogy ha van egy kiinduló adathalmazunk annotálva (azaz címkézve, szerzőkhöz társítva), akkor hogyan tudjuk a meglévő címkéinket egy nagy adathalmazra átvinni<sup>3</sup>. Másfelől a mutató, mellyel a szakirodalom alapján az optimális topikszámot kereshetnénk meg, az Author-Topic Model esetében nem használható. Szeretnék egy lehetséges heurisztikát mutatni, amivel megkereshetjük az optimális topikszámot.

Szeretném megkönnyíteni a topikok értelmezését, és elnevezésüket. Ehhez vizualizációkat készítettem, melyeket a dolgozatban statikus formában mutatok be, de a szakedolgozathoz tartozó github oldalamon<sup>4</sup> interaktív vizualizációként is

---

<sup>1</sup> Dolgozatomban sokszor a kifejezések angol megfelelőit használom, de az első előfordulásuk esetében megadom a magyar fordításukat is.

<sup>2</sup> Ennek megfelelően dolgozatomban nem tértem ki a korrupció irodalmára.

<sup>3</sup> Erre azért van szükség, mert a csomagban nincsen implementálva a topic inference.

<sup>4</sup> eszterkatona.github.io/atm-kmonitor/

elérhetőek, és kipróbálhatóak. Végül, de nem utolsó sorban a céloom, hogy a K-Monitor cikkgyűjteményének címkézését segítsem, standardizáljam és egyszerűsítsem.

## **1. 2. A dolgozat felépítése**

Dolgozatom két nagy részből áll: egy elméleti bevezetőből és egy esettanulmányból.

Az elméleti keretet a gépi tanulás fogalmától indítom, a természetes nyelvfeldolgozás bemutatását követően pedig áttérek a topikmodellek történetére röviden. Így jutunk el oda, hogy bemutathassam, hogy mi az az elméleti, matematikai háttér, amire a topikmodellek épülnek. Definiálom a főbb fogalmakat, melyeket a későbbiekben használni fogok, és a topikmodellezés alapját képezik.

Mivel az általam használt Author-Topic Model, két modellre épül, az szerző modellre (*Author Model*) és a látens Dirichlet allokációra, így fontosnak tartom ezek rövid bemutatását, mielőtt rátérek az általam használt modell ismertetésére. Végül egy rövid fejezetet szánok a modell futási idejének leírására.

A második részt az elemzés során használt eszközök bemutatásával kezdem. Fontosnak tartom, hogy rendezett környezetben dolgozzunk, ezért a fejlesztési környezet bemutatására is kitérek. Ezt követően bemutatom az adataimat, azok legyűjtésének és rendezésének módját, különös hangsúlyt fektetve a címkék kezelésére, mert ez kulcsfontosságú része a modellépítésnek.

Ezt követően bemutatom az adattisztítás folyamatát, a korpusz előkészítésének lépéseit, és a modell tanítási folyamatát.

A szakdolgozatom egy fontos része a topikokkal foglalkozó fejezet, melyben célkitűzéseimnek megfelelően bemutatom a címkeajánlás során használt módszert. A predikciót alkalmazva adok heurisztikát az optimális topikszám megtalálására, és a topikok elnevezésének menetét is itt mutatom be. Végül a különböző előfeldolgozási státuszokon átesett korpuszok eredményeiről írok röviden.

A mellékletben megtalálható a szakdolgozathoz tartozó kód, a nem szervesen kapcsolódó ábrák, illetve a gyakran használt jelölések feloldása.

## II. ELMÉLETI KERET

A természetes nyelvfeldolgozás (*Natural Language Processing, NLP*) a számítógéptudomány (*computer science*), a mesterséges intelligencia (*Artificial Intelligence, AI*) és a nyelvészet alterülete (bővebben a témáról: Liddy, 2001).

Gépi tanulásnak (*Machine Learning, ML*) nevezik azokat az mesterséges intelligencián alapuló algoritmusokat, melyek a tapasztalati úton tanulnak, strukturálatlan vagy félig strukturált adatokban keresnek mintázatot (Mitchell, 1997). Radim Rehurek (2011) Pinkerre (2010) hivatkozva azt írja, hogy a gépi tanulás algoritmusainak sikereit nem a nyelvi értelemben vett „eleganciával” mérhetjük, hanem azzal, hogy mennyire alkalmazhatóak egy adott probléma megoldására. Szerinte még a legfejlettebb NLP eszközök is a „buta, de hasznos” paradigmába tartoznak. A buta kifejezés itt nagyjából arra vonatkozik, hogy az egyes modellek nem megfelelőek az emberi nyelv gazdagságának és annak modellezésére, hogy hogyan éljük meg a nyelvet belsőleg, de ahogy Rehurek is írja – nagyon egyszerű gondolatok vannak egy-két módszer mögött, a matematikai implementációjuk mégsem mondható egyszerűnek.

A gépi tanulás legfontosabb célkitűzése hatékony és robusztus algoritmusok előállítása, melyekkel ismeretlen elemeket is pontosan előrejelezhetünk, vagy kategóriákba sorolhatunk. A modellalkotás során megkülönböztetünk felügyelt (*supervised*), és nem-felügyelt (*unsupervised*) módszereket.

Az általam bemutatott modell nem-felügyelt tanulási módszeren alapul. Nem-felügyelt esetben nem magyarázó modellt építünk. Ebben az esetben egy eloszlás valamilyen tulajdonságára szeretnénk becslést adni. Ilyen módszer a K-means klaszterezés, vagy a főkomponens-elemzés is.

A felügyelt módszerek közé tartoznak többek között a lineáris és logisztikus regresszió, vagy az SVM, azaz a *Support Vector Machines* módszerek. Esetükben a mintaelemek értékei szét vannak osztva "bemenetre" ( $x$ ) és "kimenetre" ( $y$ )<sup>5</sup>. Ilyenkor  $x \rightarrow y$  kapcsolatot keresünk. Ekkor a mintára ismert az elemek  $y$  értéke, de új bemeneti

---

<sup>5</sup> Ha összesen  $N$  esettel dolgozunk, a feature-ök száma pedig  $m$  akkor  $x$  egy  $N \times m$ -es mátrix,  $y$  pedig  $N \times 1$ -es kimeneti vektor.

adatok esetében már nem. Felügyelt esetben tudjuk, hogy milyen struktúrába szeretnénk az adatainkat kódolni, és ezt tanítjuk meg az algoritmusnak.

Például egy e-mail fiók esetében a spam felismerése a feladatunk. Készíthetünk egy automatikus osztályozót a szűrésre, úgy, hogy kész példákat használunk a tanuló algoritmusunk tanítására. Ebben az esetben a tanuló mintánk a felcímkézett levelekből áll, és teszt mintán értékeli ki az algoritmusunk teljesítményét. A teszt adatokkal az algoritmus a tanulás során nem találkozik. A spam-szűrőnk esetében a spam címkét az algoritmusnak kell prediktálnia. Ahhoz, hogy az e predikció pontosságát mérni tudjuk, a tippjeinket összehasonlítjuk a tesztminta címkéivel.

Az NLP kísérletet tesz arra, hogy a jelentés egy minél teljesebb reprezentációját nyerje ki a szövegből. Az NLP nyelvi fogalmakat (szófajokat, azaz *part-of-speech (POS) tageket*: főnév, ige, melléknév) és struktúrákat (például névszói szerkezeteket, vagy függőségi kapcsolatokat) használ. A látens tartalom feltárására különböző reprezentációkat alkalmaz, mint például:

- szavak szójegyzékét (szótárat) a kifejezések jelentéseivel és nyelvtani tulajdonságaival,
- entitások, cselekvések ontológiáját, vagy
- szinonimák tezaurusát.

A szövegbányászat (*textmining*) célja a nem triviális tudás feltárása strukturáltalan szövegekből. Ez alatt sokféle módszert érthetünk, ilyen például az úgynevezett információkinyerés (*information retrieval*), entitás/reláció kinyerés<sup>6</sup> (*entity/relation extraction*), szövegklasszifikáció és klaszterezés (Kao – Poteet, 2007:1). Szövegbányászati módszer a dolgozatomban bemutatott topikmodellezés is.

A Natural Language Processing talán legelterjedtebb felhasználási területei a társadalomtudományokban a szentimentelemzés, a véleményelemzés (*opinion mining*) valamint a topik identifikáció, a topikmodellek. Dolgozatomban a topikmodellek egyik típusát, a szerző-topikmodellt (Author-Topic Model, ATM) fogom használni.

---

<sup>6</sup> Ilyen például III. 2. 4. 1. fejezetben bemutatott *Named Entity Recognition*.



A topikmodellek fejlődése az utóbbi években gyorsan alakult. A topikmodellek a látens szemantikus indexelésből (*latent semantic indexing, LSI*) alakultak ki (Deerwester et al., 1990), bár az LSI még nem valószínűségi modell. Hoffmann (2001) az LSI-t továbbgondolva alakította ki a valószínűségi látens szemantikus elemzést (*probabilistic latent semantic analysis, PLSA*), ezt követően Blei, Ng és Jordan (2003) írt először a PLSA kiterjesztésével létrehozott látens Dirichlet allokációról (*Latent Dirichlet Allocation, LDA*). Az LDA-nak számos kiterjesztése létezik, a fejezet végén írok részletesebben ezekről.

## 1. A topikmodellezés alapfogalmai

A valószínűségi topikmodellek segítséget nyújthatnak a nagyméretű szöveges korpuszok megértéséhez (Blei, 2011), és alkalmasak az adataink, korpuszunk felderítésére, az adatokban található struktúrák feltárására. A topikidentifikáció az adatokban rejlő témák reprezentációjára alkalmas, segít megtalálni a látens topikokat, melyek a korpusz főbb témáit képezik.

Blei és Lafferty (2009) szerint a topikok megjelenítik a korpusz összefoglalását, amit kézzel lehetetlen lenne megfigyelni. A topikmodellek olyan kapcsolatokat tárhatnak fel a dokumentumok között és a dokumentumokon belül, melyek nem nyilvánvalóak, melyekre nem számítunk a priori.

### 1.1. Vektortérmodell

A topikmodelleket dokumentumok látens csoportokba sorolására használjuk, amely során dokumentumokat hasonlítunk össze. Erre a vektortérmodell (*Vector Space Modelben, VSM*) ad lehetőséget, ami a dokumentumokat egy sokdimenziós vektortérben értelmezi.

Rehurek (2011) tanulmánya alapján a vektortérmodellben minden egyes dokumentumot tulajdonságok (*feature-ök*) reprezentálnak.

Egy feature reprezentálható egy kérdés és válasz párként, például következőképpen:

1. Hányszor jelenik meg egy adott szó a dokumentumban? Nullaszor.

2. Hány paragrafust tartalmaz a dokumentum? Kettőt.

A kérdést általában egy egész szám (pl. itt 1 és 2) képviseli, így a dokumentum reprezentációja olyan párok sorozata lesz, mint (1, 0.0), (2, 2.0), ... . Ha minden kérdést ismerünk, felírhatjuk a válaszokat impliciten: (0.0, 2.0,).

Ez a válaszsorozat nagydimenziós (ebben az esetben 2-dimenziós) vektorként értelmezhető. Olyan kérdést fogalmazzunk meg, melyre a válaszként egy valós számot kapunk (vagy legalábbis a válasz legyen átalakítható azzá). Fontos, hogy a kérdéseinket jól választottuk meg, és minden dokumentumhoz ugyanazok a kérdések tartoznak. Ha ez teljesül, akkor összehasonlíthatunk egymással két vektort – ami két dokumentumot reprezentál –, ha a számok a két vektorban hasonlóak,

azt mondhatjuk, hogy az eredeti dokumentumoknak a vizsgált kérdések szempontjából hasonlónak kell lenniük.

A modell tehát a két vektortér között végez átalakítást, egy transzformációként értelmezhető a dokumentum két különböző reprezentációja között.

A vektortérmodellel különböző hasonlósági mutatókat hozhatunk létre, vizsgálhatjuk

- két dokumentum hasonlóságát,
- két szó hasonlóságát, vagy
- szókapcsolatok hasonlóságát.

A különböző hasonlóságfüggvények bemenete két vektor-reprezentáció, és a kimenetük egyetlen szám, ami a hasonlóságot méri.

A dokumentumainkat a vektortérmodellben szó-dokumentum mátrix (*term-document matrix*) formájában ábrázolhatjuk. A szó-dokumentum mátrix Tikk et al. (2007) alapján a dokumentumaink gyűjteménye, ahol a mátrix sorvektorai a korpusz unikális szavait tartalmazzák. Ezek az egyedi kifejezések alkotják a szótárat. A mátrix oszlopvektorai az egyes dokumentumok. A mátrix ritka és a mérete nagyon nagy, ennek kezelésére matematikai megoldást az alkalmazott topikmodell jelent, mert dimenziócsökkentő eljárásként működnek, másrészt a III. 2. 4 fejezetben bemutatott előfeldolgozási módszerek nyelvtechnológiai megoldásokkal kezelik a problémát (Balogh, 2015).

Egy másik fontos reprezentáció a vektortérmodellben, mellyel a dokumentumainkat megjeleníthetjük, a szózsák (*bag-of-words*) reprezentáció, amelyben Rehurek (2011) tanulmánya alapján minden megfigyelést egyetlen vektor, azaz egy térbeli pont jelöl. A bag-of-words egy olyan *multiset* halmaz, ami megengedő a duplikációkkal szemben. Egy dokumentumot reprezentálhatunk a dokumentum szavaival (ezek a „szózsákban” a *tokenek*). A bag-of-words reprezentációban két dokumentumot hasonlónak tekintünk, ha az azokat felépítő szavak hasonlóak.

A „Lenni vagy nem lenni” multiset reprezentációja: {lenni, vagy, nem, lenni}. A multisetben szereplő szavak sorrendje tetszőleges, így az előző dokumentumot a következő vektor reprezentálja:  $\langle 2, 1, 1 \rangle$ . A vektor elemei a tokenek gyakoriságát mutatják, tehát 2 = lenni, 1 =nem, 1 =vagy.

Már ebből is látható, hogy a bag-of-words reprezentáció információvesztéssel jár, hiszen nem veszi figyelembe a szintaktikai szerkezeteket, a szórendet. Szerencsére ez számunkra nem jelent gondot, a topikmodellünk esetében erre nincs is szükségünk, mert a topikmodellek a szavak együttes előfordulására koncentrálva reprezentálják a dokumentumokat.

## 1. 2. A Markov-lánc Monte Carlo módszer, és a Gibbs mintavételezés

A következő szakaszt Rosen-Zvi – Chemudugunta – Griffith – Smyth – Steyvers (2008) és Radim Rehurek (2011) tanulmányai alapján állítottam össze.

Az Author-Topic Model két folytonos véletlen változót ( $\Theta$  és  $\Phi$ ) tartalmaz. A hierarchikus Bayes-modellben szereplő folytonos véletlen változók posterior eloszlásának becslésére az elmúlt években a variációs következtetéstől (*variational inference*, VI) kezdve a Markov-lánc Monte Carlo (*Markov-Chain Monte Carlo*, MCMC) módszerekig többféle eljárást dolgoztak ki. A Bayes-i modellezés, kiterjeszti a maximum likelihood becslés (MLE) és a maximum a posteriori módszer (MAP) pontbecsléseit, és a teljes posterior eloszlást modellezi (Gelman, 2014).

A Gibbs-mintavételezés elkerüli azokat a nehézségeket, amikkel a teljes együttes valószínűségi minták kezelése járna. Ehhez minták láncolatát hozza létre, úgy, hogy a változók mintavételezése feltételesen független minden más változótól. Azt mondhatjuk, hogy a létrehozott lánc konvergál az együttes valószínűséghez (Robert–Casella, 2004).

Célunk a  $p(\Theta, \Phi | D^{train}, \alpha, \beta)$  posterior eloszlás becslése.

A következtetési sémánk a következő megfigyelésen alapul:

$$p(\Theta, \Phi | D^{train}, \alpha, \beta) = \sum_{z,x} p(\Theta, \Phi | z, x, D^{train}, \alpha, \beta) P(z, x | D^{train}, \alpha, \beta)$$

Maga a folyamat két lépésből áll:

- először a  $P(z, x | D^{train}, \alpha, \beta)$  -re adunk empirikus, minta-alapú becslést a Gibbs mintavétel alkalmazásával.

- Utána az adott  $\mathbf{x}$ -nek<sup>7</sup> és  $\mathbf{z}$ -nek<sup>8</sup> megfelelő minta esetén a  $p(\Theta, \Phi | \mathbf{z}, \mathbf{x}, D^{train}, \alpha, \beta)$  közvetlenül kiszámítható azáltal, hogy a Dirichlet-eloszlás konjugáltja<sup>9</sup> a polinomiálisnak.

A Gibbs mintavétel az MCMC egy olyan formája, amelyben úgy alakítjuk ki a Markov-láncot, hogy az meghatározott stacionárius eloszlású legyen (jelen tanulmányban erre területi korlátok miatt nem térünk ki, bővebben Gilks et al. (1996) ír róla).

Jelöljük egy dokumentum  $i$ -edik szavát  $w_{di}$ -vel, a szerző hozzárendelést a  $w_{di}$  szóhoz  $x_{di}$ -vel, és  $z_{di}$ -vel pedig a topikhozzárendelést  $w_{di}$  szóhoz.

A Gibbs-mintavétel segítségével a  $P(\mathbf{z}, \mathbf{x} | D^{train}, \alpha, \beta)$  közös eloszlásból származó mintát hozhatunk létre úgy, hogy

- minden egyedi  $w_{di}$  szóhoz mintavételezünk egy  $x_{di}$  szerző hozzárendelést és egy  $z_{di}$  téma-hozzárendelést, azzal a feltétellel, hogy a szerzők és a topikok minden egyes szóra rögzítettek.
- A folyamatot megismételjük minden szóra.

Egyetlen Gibbs mintavételi iteráció a szerző és a témakörök mintavételezésének sorozatos végrehajtását tartalmazza a korpusz minden egyes szavához.

Minden Gibbs mintavételi iterációban minden  $d_i$ -edik szóhoz meghatározzuk a topik és szerző hozzárendelést az alábbi egyenletben szereplő együttes feltételes eloszlásból:

$$P(x_{di} = a, z_{di} = t | w_{di} = w, \mathbf{z}_{-di}, \mathbf{x}_{di}, w_{di}, A, \alpha, \beta) \propto \frac{C_{wt, -di}^{WT} + \beta}{\sum_{w'} C_{w't, -di}^{WT} + W\beta} \frac{C_{ta, -di}^{TA} + \alpha}{\sum_{t'} C_{t'a, -di}^{TA} + T\alpha}$$

A Gibbs mintavételezés nehézsége abban rejlik, hogy nem tudjuk, mikor hajtottunk végre elegendő iterációt, vagyis hogy mikor konvergált megfelelően a Markov-láncunk. A gyakorlatban eleinte egy rögzített mennyiségű iterációt hajtottunk végre, és az előre meghatározott számú iteráció után (ami a Gibbs mintavételezés *burn-in* ideje, azaz beégési ideje) kezdjük csak el az  $\mathbf{x}^s$  és  $\mathbf{z}^s$  mintákat rögzíteni. A burn-in célja, tehát az, hogy a mintavétel közelítse a stacionárius eloszlást, ami nem más, mint a  $P(\mathbf{z}, \mathbf{x} | D^{train}, \alpha, \beta)$  posterior eloszlás.

<sup>7</sup> Szerző hozzárendelés

<sup>8</sup> Topikhozzárendelés

<sup>9</sup> Azaz, ha a polinomiális paraméterek prior eloszlása Dirichlet, akkor a posterior eloszlás is Dirichlet.

### 1. 3. Variational Bayes

A Python Gensim csomagját használtam az elemzésem során. A csomag részletes bemutatására a III. 1. 4. fejezetben kerül sor. A Gensimben a Gibbs mintavétel helyett a *Variational Bayes (VB)* egy típusa, a *blocking Variational Bayes* lett implementálva.

Az MCMC alkalmazása akkor lehet jó döntés, ha biztosak vagyunk benne, hogy a modellünk megfelelően illeszkedik, vagy ha kisebb adathalmazon dolgozunk, tehát, ha gond nélkül szánunk hosszabb időt a rendkívül időigényes modell futtatására. Ezzel szemben a variációs következtetést effektíven alkalmazhatjuk nagy adathalmazon (Blei et al, 2017). A következőkben Mortensen (2017) disszertációja alapján mutatom be a Variational Bayes előnyét a Gibbs mintavétellel szemben.

Bár a VB módszerrel sosem érjük el a valós értéket, a Gibbs mintavételezéssel megfelelő számú iteráció után azt mondhatjuk, hogy közelítjük a valós posteriort.

A Variational Bayes esetén a  $q(\Theta, \beta, \mathbf{z}, \mathbf{x})$  faktorizált variációs eloszlás közelíti a posterior eloszlást:

$$\begin{aligned} q(\Theta, \beta, \mathbf{z}, \mathbf{x}) &= q(\Theta | \gamma) q(\beta | \lambda) q(\mathbf{z} | \Phi) q(\mathbf{x} | \mu) \\ &= \prod_a q(\Theta_a | \gamma_a) \prod_t q(\beta_t | \lambda_t) \prod_{d,i} q(x_{di} | \mu_{di}) \prod_{d,i} q(z_{di} | \Phi_d) \\ &= \prod_a \text{Dir}(\Theta_a | \gamma_a) \prod_t \text{Dir}(\beta_t | \lambda_t) \prod_{d,i} \text{Mult}(x_{di} | \mu_{di}) \prod_{d,i} \text{Mult}(z_{di} | \Phi_d) \end{aligned}$$

A  $\gamma$ ,  $\lambda$ ,  $\mu$  és  $\Phi$  változó paraméterek bevezetésével azt feltételezzük, hogy a modell paraméterei függetlenek. Ez a (valóságnak nem feltétlenül megfelelő) feltevés teszi lehetővé a megoldásunk közelítését.

Viszont definiálhatunk egy olyan variációs eloszlást, ahol  $q(x_{di} = a, z_{di} = t | \Phi_{di}) = \Phi_{diat}$ :

$$\begin{aligned}
q(\Theta, \beta, \mathbf{z}, \mathbf{x}) &= q(\Theta | \gamma) q(\beta | \lambda) q(\mathbf{x}, \mathbf{z} | \Phi) \\
&= \prod_a q(\Theta_a | \gamma_a) \prod_t q(\beta_t | \lambda_t) \prod_{d,i} q(x_{di}, z_{di} | i) \\
&= \prod_a \text{Dir}(\Theta_a | \gamma_a) \prod_t \text{Dir}(\beta_t | \lambda_t) \prod_{d,i} \text{Mult}(x_{di}, z_{di} | \Phi_{di})
\end{aligned}$$

Ennél a közelítésnél, amit blocking Variational Bayes-nek nevezünk, azt feltételezzük, hogy  $\mathbf{x}$  és  $\mathbf{z}$  nem függetlenek.

Számítási tulajdonságait nézve a blocking VB jobb algoritmus, mint a standard Bayes-módszer.

A standard algoritmus a következő lépéseket hajtja végre:

1. Inicializálja a paramétereket.
2. Amíg a *lower bound* nem konvergál, minden  $d$ ,  $v$ ,  $a$  és  $t$  kombináció esetén újraértékeljük (*updateljük*) a  $\Phi$ ,  $\gamma$  és  $\lambda$  változókat.

A lokális ( $\Phi$  és  $\gamma$ ) változók frissítését M-lépésnek, a globális ( $\lambda$ ) változók frissítését E-lépésnek nevezzük. Az algoritmus nagy hátránya, hogy az összes változó tárolását a memóriára bízta, és a  $\Phi$  reprezentációja egy nagyon nagy  $D \times V_d \times A_d \times T$ , ritka mátrix lehet.

Ezzel szemben a blocking VB előnyei közé sorolható, hogy elkerülhető a látens változók variációs megfelelőjének, azaz  $\Phi$  tárolása, valamint lehetővé teszi a variációs változók vektorizálását. Mivel kevesebb feltételezéssel él a posterior alakjáról, elméletileg a blocking VB algoritmusnak javítania kell a konvergenciát.

Az, hogy nem tároljuk  $\Phi$ -t a memóriában, az algoritmus memória komplexitását lecsökkenti a következőre:

$$O(V_d A_d T + AT + KV + \sum_d A_d + \sum_a D_a)$$

a következőre:

$$O(AT + KV + \sum_d A_d + \sum_a D_a)$$

Az ATM memória komplexitását a későbbiekben részletesen bemutatom, a modell futási idejéről a II. 4. fejezetben írok részletesen.

## 2. Topikmodellek, az Author Model és a látens Dirichlet allokáció

A dolgozatomban felhasznált Author-Topic Model két generatív modellre épül: a szerző-modellre (Author Model), illetve a látens Dirichlet allokációra (LDA).

A generatív folyamat lényege Shalev-Schwartz és Ben-David (2014) alapján a  $P(X,Y)$  együttes valószínűségének megismerése (szemben a disztributív modellekkel, melyek a  $P(Y|X)$  valószínűséget keresik). A generatív modellek lehetővé teszik a modell paramétereinek közötti feltételes függőség leírását. A generatív modell azt feltételezi, hogy az adatokat egy adott, néhány paraméter által meghatározott eloszlás hozza létre, és különböző algoritmusokkal közelíti ezt az eloszlást. Ez teszi lehetővé az adatok generálását (és nem csupán osztályozását).

A topikmodellek esetében például a generatív folyamat során kiválasztjuk a dokumentumokhoz tartozó topikokat, és a dokumentum szavait a topikok függvényében határozzuk meg (Balogh 2015).

Egy lehetséges topikmodell az Author-Topic Model egyik alappillére adó LDA.

A módszert Balogh Kitti (2015) szakedolgozata alapján szeretném bemutatni.

Az LDA egy generatív valószínűségi modell, alapvetően szöveges korpuszokon alkalmazzák. A modell azt feltételezi, hogy a szavak erős szemantikai információkkal rendelkeznek, és a hasonló témákkal foglalkozó dokumentumok hasonló szavak csoportjait használják. A látens témákat tehát a korpusz dokumentumaiban gyakran együtt szereplő szavak csoportjainak azonosításával fedezik fel.

Előzetesen csak a témák számát határozzák meg, és csak egy megfigyelt változó van: a dokumentumok szavai. Ezekhez az inputokhoz az LDA visszaadja a dokumentumok topikjainak megoszlását, valamint az egyes topikok szóeloszlását.

Az alábbi (1.) ábrán öt dokumentumot látunk. A színek jelölik a topikokat.

Az első és a második dokumentumban az első topik érvényesül, és az étkezés témája köré csoportosul (narancs, reggeli, alma). A harmadik dokumentum egyértelműen a második topikba tartozik, és láthatóan állatokkal kapcsolatos főneveket, és melléknevet tartalmaz. Az utolsó két dokumentumban keverednek a topikok, illetve egy harmadik topik is felfedezhető.





1. ábra: LDA vizualizációja  
(készítette: Szűcs Krisztina)

Alapvetően az LDA sem felügyelt módszer, csak a dokumentumokban szereplő szavakat modellezzük, a cél az, hogy olyan topikokat kapjunk, amik maximalizálják a gyűjtés likelihoodját, posterior valószínűségét.

Az LDA-nak több kiterjesztése létezik, ezek közé tartoznak például a korrelált topikmodellek (*correlated topic models*), amik a topikok közötti interakciót modellezik (Blei – Lafferty 2007), a dinamikus topikmodellek (*dynamic topic models*) pedig a topikok időbeli változását vizsgálják (Blei – Lafferty 2006), a szintaktikai topikmodellek (*syntactic topic models*) leírják a szöveges dokumentumban szereplő szavak szintaktikai korlátait (Byord-Graber – Blei 2009). Ahogy Roberts, Stewart, Tingley és Airolti (2013) tanulmányukban írják, a társadalomtudományokban is egyre közkedveltebb eszköz látens nyelvi, politikai és pszichológiai változók mérésére. Szerintük a hagyományos kérdőíves módszert alkalmazó kutatóknak gyakran nehéz dönteniük arról, hogy alkalmazzanak-e nyílt kérdéseket, mert ezek elemzése költséges lehet. Azonban a topikmodellek csökkenthetik az elemzés költségeit, ennek elősegítésére létrehozták (az LDA kiterjesztéseként) a *Structural Topic Modelt*. A következő fejezetben tárgyalt Author-Topic Modell is az LDA kiterjesztésének tekinthető.

Az Author-Topic Model másik alappillére, az Author Model egy egyszerű generatív modell, ahol minden szerzőhöz egy adott szóeloszlás tartozik.

Az Author Model egy egyszerű módszer a szerzők érdeklődésének modellezésére. Tegyük fel, hogy a szerzők egy csoportja  $a_d$ , úgy dönt, hogy megír  $d$  dokumentumot. A dokumentum minden egyes szavához véletlenszerűen kiválasztunk egy szerzőt és egy szót a valószínűségi eloszlás alapján, amely a szerzőre jellemző. Ez a szerzőmodell azonban nem tartalmaz olyan információt a dokumentum tartalmáról, amely meghaladja a dokumentumban szereplő szavakat és a dokumentum szerzőit (Rosen-Zvi – Griffiths – Steyvers – Smyth 2004a).

### 3. Az Author-Topic Model

Az Author-Topic Model egy generatív valószínűségi modell, ami szerzőségi információval egészíti ki az LDA-t. A modellben a szerzők foglalják el a dokumentumok helyét. Minden szerzőhöz (ahogy eddig a dokumentumokhoz) a topikok polinomiális eloszlása tartozik, és minden topikhoz a szavak polinomiális eloszlása tartozik. A modell egyszerre vizsgálja a dokumentum tartalmát, valamint a szerzők által leggyakrabban érintett témákat.

A következőkben Rosen-Zvi – Chemudugunta – Griffiths – Smyth – Steyvers (2008) tanulmánya alapján mutatom be részletesebben a modell működését.

Egész számokat fogunk használni a szótár bejegyzéseinek jelölésére: minden  $w_i$  adott dokumentumunk  $i$ -edik szavát jelöli, ahol  $i$  1-től  $N$ -ig terjed.

Nyilvánvalóan lehetnek duplikációk a szavak között, így legyen  $W$  a szótár egyedi szavainak száma.

Egy  $d$  dokumentumot a szavak  $\mathbf{w}_d$  vektoraként reprezentálhatunk,  $N_d$  a szavak száma. Egy  $D$  dokumentumból álló korpusz a dokumentumvektorok ( $\mathbf{w}$ ) összességével ábrázolható,  $N = \sum_{d=1}^D N_d$ .

A szavak mellett az egyes dokumentumok szerzőiről is van információnk.

Definiáljuk  $d$  dokumentum szerzőit  $\mathbf{a}_d$ -ként.  $\mathbf{a}_d$  egész számokból áll, 1-től  $A$ -ig, ahol  $A$  a korpuszban található dokumentumok szerzőinek a száma. Jelöljük  $\mathbf{a}_d$ -vel a  $d$  dokumentum szerzőinek számát.

Nézzünk egy egyszerű példát:

Tegyük fel, hogy  $D = 3$  dokumentumunk van, melyeket  $A = 2$  szerző írt, akik  $W = 1000$  egyedi szót használtak. Az első cikket az egyik, míg a másodikat a másik szerző írta, a harmadik tanulmányt pedig együtt készítették.

A fent bevezetett jelölésünkkel:

$\mathbf{a}_1 = (1)$ ,  $\mathbf{a}_2 = (2)$ ,  $\mathbf{a}_3 = (1, 2)$  és  $A_1 = 1$ ,  $A_2 = 1$ , és  $A_3 = 2$ .

Tegyük fel, hogy az első dokumentum egyetlen sorból áll: „A nyelv modellezése érdekes kutatási problémákat tartalmaz”. Ha eltávolítjuk a stopszavakat<sup>10</sup>, ami ebben az esetben az „a”, 6 szó marad a dokumentumban.

Ha a szótárunkban a „nyelvmodellezés” a nyolcadik bejegyzés, a „modellezés” a 12., az „érdekes” pedig a 115, akkor  $w_1 = 8$ ,  $w_2 = 12$ ,  $w_3 = 115$  és így tovább.

A szerző-téma modell egy hierarchikus generatív modell, amelyben minden dokumentum  $w_1, w_2 \dots w_N$  szavához két rejtett változó társul:  $x_1, x_2, \dots, x_N$  szerző és egy  $z_1, z_2, \dots, z_N$  topik. Ezek a látens változók két további  $\mathbf{z}$  és  $\mathbf{x}$   $N$ -dimenziós vektorral növelik a  $\mathbf{w}$   $N$ -dimenziós vektort, amik a topikok és a szerzők hozzárendelését jelentik az adott szavakhoz.

Tegyük fel, hogy van  $T$  topikunk. Az egyes szerzőkre a topikok polinomiális eloszlását paraméterezhetjük egy  $T \times A$  méretű mátrixszal, melynek  $\Theta_{ta}$  elemei  $t$  topik hozzárendelésének valószínűségét jelentik egy olyan szóhoz, ami  $a$  szerzőtől származik.

$$\sum_{t=1}^T \Theta_{ta} = 1$$

Az egyes témákkal összefüggő szavakra illesztett polinomiális eloszlások egy  $W \times T$  méretű mátrixszal paraméterezettek, melynek  $\Phi_{wt}$  elemei jelölik annak valószínűségét, hogy a  $w$  szó a  $t$  topikból származik.

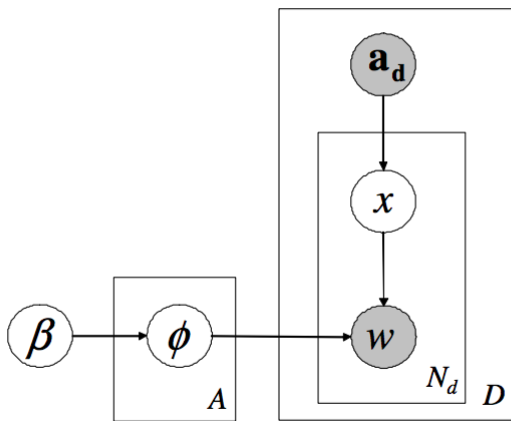
$$\sum_{w=1}^W \Phi_{wt} = 1$$

Feltételezzük, hogy a fenti polinomiális eloszlások szimmetrikus Dirichlet priorokból származnak  $\alpha$  és  $\beta$  hiperparaméterekkel.

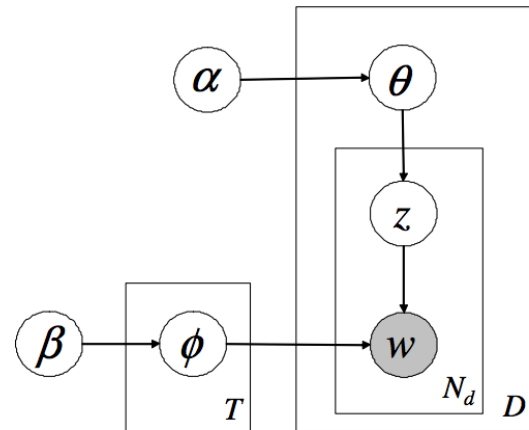
A generatív folyamathoz tartozó grafikus modell a 4. ábrán látható. A becsléseinkhez feltételezzük, hogy minden dokumentum teljes szerzői gárdáját megfigyeltük. Minden szerzőhöz hozzárendeljük a topikok polinomiális eloszlását. Feltéve, hogy adott a szerzői gárdánk és a szerzőkhöz tartozó topikeloszlás, a dokumentumok generálásának folyamata a következőképpen írható le:

<sup>10</sup> Stopszavak alatt azokat a kifejezéseket értjük, melyek nem hordoznak tartalmi információt, és ezért ezeket eltávolítjuk a szövegekből (Tikk et al., 2007).

- először egy egyenletes eloszlás segítségével véletlenszerűen választunk egy szerzőt minden szóhoz, ami előfordul a dokumentumban
- utána minden szóhoz egyelemű mintát veszünk (ami egyetlen topik lesz) a szó szerzőjéhez tartozó topikeloszlásból
- végül az egyes topikokhoz tartozó szavak eloszlásából is mintát veszünk.

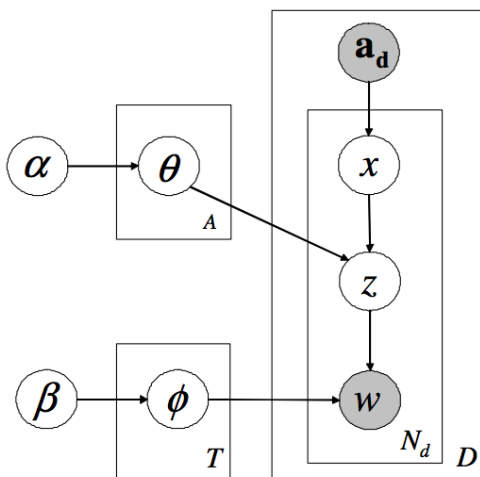


2. ábra: Szerzőmodell (Author Model)



3. ábra: Látens Dirichlet allokáció

(Rosen-Zvi – Griffiths – Steyvers – Smyth 2004)



A grafikus modellhez tartozó pszeudó-kód:

- (1) For each author  $a = 1, \dots, A$  choose  $\Theta_a \sim \text{Dirichlet}(\alpha)$   
For each topic  $t = 1, \dots, T$  choose  $\Phi_t \sim \text{Dirichlet}(\beta)$
- (2) For each document  $d = 1, \dots, D$   
Given the vector of authors  $\mathbf{a}_d$   
For each word  $i = 1, \dots, N_d$   
Conditioned on  $\mathbf{a}_d$  choose an author  $x_{di} \sim \text{Uniform}(\mathbf{a}_d)$   
Conditioned on  $x_{di}$  choose a topic  $z_{di} \sim \text{Discrete}(\Theta_{x_{di}})$   
Conditioned on  $z_{di}$  choose a word  $w_{di} \sim \text{Discrete}(\Phi_{z_{di}})$

4. ábra: Author-Topic model

(Rosen-Zvi – Chemudugunta – Griffiths – Smyth – Steyvers 2008)

Az Author-Topic Model tehát két modell ötvözése, egy-egy valószínűségi modell a szerzőkre és a topikokra. A dokumentumok topikeloszlása a dokumentumok szerzőihez tartozó topikeloszlásnak a keveréke. Az Author-Topic Model esetében minden szerzőhöz egyedi topikeloszlás tartozik, a modell segítségével megismerhetjük, hogy mely szerző mely topikokról ír, és milyen gyakran.

A szerzőket nem kell szigorúan érteni. Esetemben a szerző olyan címkéként jelenik meg a dokumentumokon, amiket a könnyebb kereshetőség elősegítésére a K-Monitor önkéntesei a cikkekhez rendeltek, és amik a cikkek tartalmára utalnak. De lehetne a szerző akár a cikk keletkezésének dátuma, a cikk forrása, vagy bármilyen más metaadat.

#### 4. Futásidő, skálázhatóság

Az algoritmusok esetében fontos kitérni a futási időre (time complexity), ennek bemutatására Rosen et al. (2008), Seroussi—Zukerman—Bohnert (2011) és Mortensen (2007) tanulmányokat használok.

Egy algoritmustól azt várjuk, hogy skálázható legyen. A skálázhatóság annak mutatója, hogy növekvő inputra hogyan változik a futási idő. Ha amikor a számítási kapacitást  $N$ -szeresére növeljük, a teljesítmény is  $N$ -szeresére növekedik, lineáris skálázhatóságról beszélhetünk.

Az algoritmusokat futási idejük – tehát az algoritmus végrehajtására szükséges idő – szerint a nagy ordó jelölés (*big O notation*) segítségével osztályozhatjuk. A big O a függvénynövekedés menetének megfelelően jellemzi a függvényeket, így a függvény növekedésének mértékét fejezi ki. Azért az O betűt használjuk, mert egy függvénynövekedés menetét a függvény „rendjének” (order) is nevezhetjük (Black, 2017).

Az előző részben leírtakat követve legyen  $S$  a Gibbs mintáink száma,  $W$  a tanuló korpuszunk szavainak száma,  $T$  a topikok száma.

A látens Dirichlet allokáció esetében az algoritmus minden Gibbs mintában végigiterál a korpusz minden szaván, és minden egyes szó esetében végigiterál a topikokon. Így a modell tanulási ideje (*time complexity*)  $O(SWT)$ .

A tanulási szakasz után a topikeloszlás inferenciája egy  $N$  hosszúságú teszt dokumentumra  $O(SNT)$ . Az általam használt Author-Topic Model esetében ez még kiegészül a szerzők számával, ha  $A_{\max}$ -nak nevezzük az egyetlen dokumentumhoz társítható szerzők maximális számát, akkor a képletünk a következő lesz:

$O(SNT + A_{\max}T)$ .

Az algoritmus aszimptotikus komplexitása miatt azt mondhatjuk, hogy ha egyetlen tényezőt változtatunk (függetlenül), akkor lineáris növekedésre számíthatunk a futási időben.

Szakdolgozatom megírásához a Pythonban elérhető Gensim csomagot használtam. A Gensimben implementált modell futási idejére a paraméterek bemutatásánál térek vissza (a III. 3. 2. fejezetben).

### III. ESETTANULMÁNY

#### 1. Az elemzés során használt eszközök bemutatása

##### 1. 1. A fejlesztési környezet

A fejlesztési környezet bemutatását azért tartom fontosnak, mert ez adja meg a projekt keretét, ez segít abban, hogy a folyamatok átláthatóak és rendezettek legyenek. Az, hogy virtuális környezetet (*virtual environment*)<sup>11</sup> használtunk a szakdolgozatom elkészítéséhez platform-függetlenné tette a projektet, ezáltal az elemek külön kezelhetővé váltak, a köztük lévő függőségekkel (*dependenciákkal*). Lényegében a projektem így gond nélkül elérhetővé vált bárki számára, aki a projekthez hozzáfért, a projekttel együtt megkapta a szükséges Python csomagokat, függőségeket, számára is futtathatóvá vált a program.

A projekt felépítését a Cookiecutter Data Science<sup>12</sup> *project templatje* adta, valamint ennek köszönhetően használhattuk a Luigi pipeline<sup>13</sup> eszközt is. A projekt során a *Feature Branch Workflow*-t követtük, azaz a *Git* verziókövetőben minden feldolgozási és elemzési lépést új *branchen* (ágon) fejlesztettünk ki és integráltunk a fő fejlesztési ágba. A Gitlab<sup>14</sup> egy Git alapú kollaborációs platform, a projekt során ezt használtuk. A projektet lépésekre bontottuk és feature-ökből<sup>15</sup> építettük fel, tehát minden alfeladatnak külön branchet hoztunk létre a projekt elkészítése során. Ennek köszönhetően a projektünkben volt többek között *data-collector*, *process-rawtext*, *filter-tags* branchünk, ami segítette a könnyebb átláthatóságot, és a munkát az egyes részfeladatokon. Az én feladatom az egyes feature-ök kifejlesztése volt, viszont nem volt feladatom a Luigi pipeline-ba integrálás, ez a fejlesztő feladata.

A Luigi pipeline az *input* -> *output* feladatokat fűzi össze, a könnyebb áttekintés és futtathatóság elősegítésére. A pipeline tehát folyamatosan hajt végre egy komplex feladatsort, de úgy, hogy az egyes műveleti fázisok külön lépésekre vannak osztva. Minden lépés az inputtal kezdődik, a lépés végén elért output, a következő lépést

---

<sup>11</sup> <https://virtualenv.pypa.io/en/stable/>

<sup>12</sup> <https://github.com/drivendata/cookiecutter-data-science>

<sup>13</sup> [https://github.com/ffmmjj/luigi\\_data\\_science\\_project\\_cookiecutter](https://github.com/ffmmjj/luigi_data_science_project_cookiecutter)

<sup>14</sup> <https://about.gitlab.com/>

<sup>15</sup> <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>



elindító input. A pipeline-ban vannak függőségek, amik egymás után jönnek, illetve vannak párhuzamosan végezhető feladatok, amik vagy nem érnek össze egyáltalán, vagy a folyamat során később érnek össze. A melléklet M1. ábráján látható a Luigi függőségi gráfja.

A Cookiecutter magával hozta azt a mappaszerkezetet (*directory structure*), amiben a kódokat és az adatokat tároltuk, ezt nevezik „*convention over configuration*” elvnek, azaz ne konfigurálj, kövesd a konvenciót -- amit a Cookiecutter rögzített. A melléklet M2. ábráján ez a váz látható. A mappa számomra legfontosabb részei a data, a models, és az src mappa voltak. Az src mappa tartalmazza a részfeladatonként megírt kódokat, ezen belül viszont például a modell tanításával kapcsolatos kódok nem kerülnek egy helyre az előfeldolgozással kapcsolatos kódokkal. Az adatokat is külön tároljuk az egyes stádiumok szerint, a nyers fájlok nem keverednek az előfeldolgozáson átesett szövegekkel. Ez a struktúra valóban nagyon sokat segít a kiigazodásban.

## **1. 2. SQL**

Az SQL (*Structured Query Language*, azaz strukturált lekérdezőnyelv) relációs adatbázisokat kezel, részben deklaratív, részben procedurális nyelv.

A relációs modell az információt összefüggő elemekként határozza meg, amelyeknek attribútumai vannak, ezeket táblákban tároljuk. A táblákat kulcsok segítségével tudjuk összekapcsolni: a kulcsok (*primary key*, *foreign key*) lehetnek például azonosítók, melyek egyediek, és több táblának is az elemei.

Különböző relációs adatbázis kezelő programok léteznek, dolgozatomban megírásához a MySQL<sup>16</sup> és a SQLite<sup>17</sup> használatára volt szükségem.

## **1. 3. Python**

A Python<sup>18</sup> egy nyílt forráskódú, szabadon felhasználható, magas szintű (*high-level*) általános célú programozási nyelv, amely számos különböző problémakategóriára alkalmazható. A Python értelmezett, interaktív programozási nyelv, ami használható

---

<sup>16</sup> <http://www.oracle.com/technetwork/database/mysql/index.html>

<sup>17</sup> <https://www.sqlite.org/about.html>

<sup>18</sup> A nyelv bemutatásához a következőket használtam: <https://docs.python.org/3/faq/general.html> - [what-is-python](#) és <https://wiki.python.org/moin/BeginnersGuide/Overview>

kiterjesztési nyelvként olyan alkalmazásokhoz is, amelyek programozható *interface-t* igényelnek. A nyelv platformfüggetlen, futtatható számos Unix változatban, használható Macintoshon, és a Windows 2000 vagy újabb verzióin. A nyelvet Guido van Rossum fejlesztette ki az 1990-es évek elején.

A Python kedvelt, könnyen használható nyelv, amely egyszerűvé teszi a program működését, szintaxisának köszönhetően a kódok könnyen olvashatóak. A nyelv úgynevezett „*batteries included*” filozófiájának köszönhetően sok beépített könyvtárral érkezik, amelyek számos gyakori programozási feladatot támogatnak, például a webszerverhez való kapcsolódást, a szöveges tartalmak keresését, a fájlok olvasását és módosítását. Az alapfunkciók könnyedén kibővíthető új modulok hozzáadásával. Dolgozatom megírásához a Python 3.6-os verzióját használtam (bár sokan még ma is a Python 2-es verzióját használják), fejlesztői környezetként (*Integrated Development Environment, IDE*) a PyCharmot vettem igénybe, de az eszköz irányítható parancssorból vagy különböző szerkesztőkkel, IDE-kel.

#### **1. 4. Gensim**

A Python Gensim nevű, topikmodellezésre alkalmas csomagját használtam az elemzésem során. Az Author-Topic Model 2017 januárja óta érhető el a csomagban. A csomag legújabb verziója 2018. március 1-én jelent meg, és gyorsabb futási időt ígér a korábbi verziókhoz képest. A csomagot Mortensen (2017) tanulmánya és a Gensim dokumentációja<sup>19</sup> alapján mutatom be.

A Gensim mottója „*Topic modelling for humans*”, azaz Topikmodellezés embereknek. A projekt három legfontosabb eleme a sebesség, a skálázhatóság, illetve az, hogy felhasználóbarát legyen.

A csomag függőségei (dependenciái) természetesen a Python (úgy a Python3 bármelyik verziója, ahogy a Python2.5, 2.6 és 2.7 is megfelelő), a NumPy, és a SciPy.

A Gensim több algoritmust tartalmaz (LSA/LSI/SVD, LDA, word2vec, stb.). Ahogy elméletben, úgy a Gensimben is nagyon hasonlít az ATM (Gensimben: AuthorTopicModel) az LDA-hoz (Gensimben: LdaModel). Az AuthorTopicModel osztály öröklí a LdaModel osztályt, a kód és az implementáció nagyon hasonló.

---

<sup>19</sup> <https://media.readthedocs.org/pdf/gensim/stable/gensim.pdf>

A Gensim egy ingyenes Python keretrendszer, melynek segítségével a szemantikus témák automatikusan kinyerhetők a dokumentumokból. A Gensim csomag a korpusz, a vektor és a modell koncepciójára épül.

A korpusz maga a digitális dokumentumok gyűjteménye. Ez a gyűjtemény többek között a dokumentumok szerkezetének, a témáinak (topikjainak) automatikus felderítésére szolgál, ezeket a látens struktúrát később felhasználhatjuk arra, hogy topikokat rendeljünk új és ismeretlen dokumentumokhoz. A csomagban a korpusz összes dokumentuma *tuple*-ök (tehát *word id*<sup>20</sup> és *word count*<sup>21</sup> párok) listájaként reprezentálódik, ahol a word id egy egész szám (*integer*), a word count pedig annak az összege, hogy a megfigyelt szó hányszor szerepelt a dokumentumban. Ezt a reprezentációt nevezzük bag-of-words vagy *unigram* modellnek<sup>22</sup>. A korpusz ebben a reprezentációban akár dokumentumok listája is lehet, de mindenképpen iterálhatónak kell lennie.

A Gensimen kívül a python-topic-model csomagban is elérhető az ATM, de a csomag implementációja Markov chain Monte Carlo (MCMC) modellen alapul, így nagyon lassú. A Gensim az MCMC helyett a Variational Bayest használ, amiről részletesebben a II. 1. 2. fejezetben írtam.

### **1. 5. Az adatvizualizációhoz használt eszközök**

A dolgozatomban a vizualizációkhoz Gephi<sup>23</sup> és D3.js JavaScript könyvtárat<sup>24</sup> használtam. A III. 3. 3. 2. fejezetben bemutatott gráfot a Python Networkx<sup>25</sup> csomagjával hoztam létre, és Gephiben készítettem el. A III. 3. 3. 1. fejezetben látható Bubble chartot D3-ban készítettem. D3-ban HTML, SVG és CSS segítségével készíthetünk interaktív vizualizációkat.

A szakdolgozatomhoz tartozó github oldalon a vizualizációk interaktív formában is elérhetőek.

---

<sup>20</sup> Szó azonosító

<sup>21</sup> Szószám

<sup>22</sup> Erről a II. 1. 1. Fejezetben írtam bővebben

<sup>23</sup> <https://gephi.org/>

<sup>24</sup> <https://d3js.org/>

<sup>25</sup> <https://networkx.github.io/>

## 2. Az adatok előkészítése

### 2. 1. Az adatok forrása: K-Monitor Akták

Dolgozatomban a K-Monitor adatbázisából kinyert cikkeket használom. A szervezettől kapott adatok az Akták<sup>26</sup> egy részét tartalmazzák.

Bemutatkozásuk szerint<sup>27</sup> a K-Monitor egy olyan civil szervezet, ami a korrupció visszaszorításáért harcol, valamint azért, hogy a közpénzek felhasználása átlátható legyen. 2007-ben létrehozták adatbázisukat<sup>28</sup>, melyben összegyűjtik a korrupcióról, a közpénzfelhasználásról szóló online sajtóban megjelent cikkeket.

Az összegyűjtött cikkeket címkézik is, hogy könnyen feltárhatóvá tegyék az adott cselekménytípushoz vagy gazdasági szektorhoz tartozó témákat. A címkézést önkéntesek, és a K-Monitor munkatársai végzik, szakdolgozattal ezt a munkát szeretném segíteni és részben automatizálni azzal, hogy címkéket ajánlok fel az önkénteseknek. Tehát amit a Author-Topic Model terminológiájában szerzőnek nevezünk, esetemben címkéket fog jelenteni.

A következőkben az önkéntesek számára kiadott instrukciót tartalmazó dokumentum alapján szeretném bemutatni az adatbázis összeállításának és címkézésének menetét.

Az adatbázisba olyan cikkek kerülnek be, melyek:

1. konkrét korrupciós esetet mutatnak be,
2. szerzője állítja, vagy sugallja, hogy valaki a ráruházott hatalmat saját maga, vagy egy harmadik fél hasznára fordította,
3. korrupciós ügyben történő jogi eljárásról tájékoztatnak,
4. vagy épp ellenkezőleg: egy korrupciós vádat cáfolnak vagy védekeznek,
5. a következő témák esetében elkövetett szabálytalanságokról szólnak: közbeszerzés, pártfinanszírozás, pályázatok, kormányzat szerv vagy állami vállalat gazdálkodása, vagyonosodás, juttatások, privatizáció, *whistleblowing*.
6. a következő kulcsszavak valamelyikét tartalmazzák: korrupció, sikkasztás (közszolga által), hűtlen kezelés, vesztegetés, hivatali visszaélés, hatalommal való visszaélés, befolyással üzérkedés, hanyagság, adócsalás, számviteli

---

<sup>26</sup> <http://k-monitor.hu/adatbazis/aktak>

<sup>27</sup> <http://k-monitor.hu/rolunk>

<sup>28</sup> <http://k-monitor.hu/tevekenysegek/20070101-adatbazis>

fegyelem megsértése, protekció, nepotizmus, jogosulatlan gazdasági előny, versenykorlátozás, kartell, *whistleblowing*/közérdekű bejelentés, közérdekű adatok, átláthatóság.

A cikkek gyűjteményébe nem kerülnek be pártközlemények, publicisztikák, mocskolódások, illetve hivatkozott, vagy más portálról átvett anyagok. Az adatbázis feladata, hogy a cikkek az interneten és a K-Monitor weblapján is könnyen kereshetők legyenek, akár személyekre, intézményekre vagy témakörökre keres a felhasználó.

Az önkéntesek feladatleírásában kiemelik, hogy a címkék esetében a már meglévőket használják, illetve, hogy az adatbázisba egy cikk csak egyszer kerüljön be. MTI cikkek esetében a legrészletesebb forrásból származó cikket viszik be, így elkerülve a tartalmi ismétlődéseket. Az adatbázisba csak online, van online is megjelenő cikkek kerülnek be, így a források minden esetben online hírportálok.

Az Akták olyan témákat járnak körül, melyek gyakrabban előkerülő, több fordulatot vett, vagy a jövőben várhatóan folytatódó ügyekről szólnak.

Az adatokat mysql dump formátumban kaptam meg. Azért, hogy könnyebben kezelhető és hordozható legyen, átkonvertáltuk sqlite3-ba. Az adatok átalakítása egy meglévő eszközt (mysql2sqlite<sup>29</sup>) használtam fel.

A K-Monitortól kapott adatbázis a következő 56 táblát tartalmazza, melyekből természetesen nem volt szükségünk mindre, vastaggal szedtem azokat a táblákat, melyeket a használtunk.

---

<sup>29</sup> <https://github.com/dumblob/mysql2sqlite>

actual_news	news_lang	newsletters_emails_group_link
actual_news_lang	<b>news_news</b>	newsletters_groups
comments	<b>news_newspapers</b>	newsletters_groups_lang
config_languages	<b>news_newspapers_link</b>	newsletters_images
config_modul	news_others	newsletters_lists
countries	news_others_link	newsletters_lists_lang
languages	news_persons	newsletters_sent
menu_table	news_persons_link	pictures_data
menu_table_lang	news_places	pictures_directory
news_categories	news_places_gps	publications
news_categories_lang	news_places_link	publications_lang
news_categories_link	news_related_link	seo_urls_data
news_files	news_static	sqlite_sqlite_sequence
news_files_link	news_static_images	tags_seo_data
news_galleries	news_static_lang	users
news_images	<b>news_tags</b>	users_detail
news_institutions	newsletters_data	users_modul_rights
news_institutions_link	newsletters_data_rights	users_sessions
news_institutions_rel	newsletters_emails	

5. ábra: A K-Monitor adatbázisának táblái

A felhasznált táblák struktúrája a mellékletben látható (M3. ábra). Az adatbázisból a cikkek azonosítóját, a portál nevét, a cikk elérési útvonalát és a cikkhez tartozó címkéket (tageket) gyűjtöttem össze, rendeztem új táblába (FullKmonitor) SQLAlchemy<sup>30</sup> segítségével, és írtam ki egy tsv fájlba.

Az SQLAlchemy-t mint objektum-relációs leképező (ORM, object-relational mapper) használtam, ez a csomag segített abban, hogy az osztályokat egymástól elkülönítsem, és felépítsem az új adatbázisom sémáját.

## 2. 2. Az adatok bemutatása

A K-monitortól kapott adatbázis 18139 URL-t tartalmazott, az adatok tisztítását követően 15195 cikkünk maradt. A leggyűjtést követően tehát 2944 cikket töröltünk. Az eltávolított cikkek nagy része szó szerinti duplikátum volt, illetve az ismétlődő cikkek

<sup>30</sup> <https://www.sqlalchemy.org/>

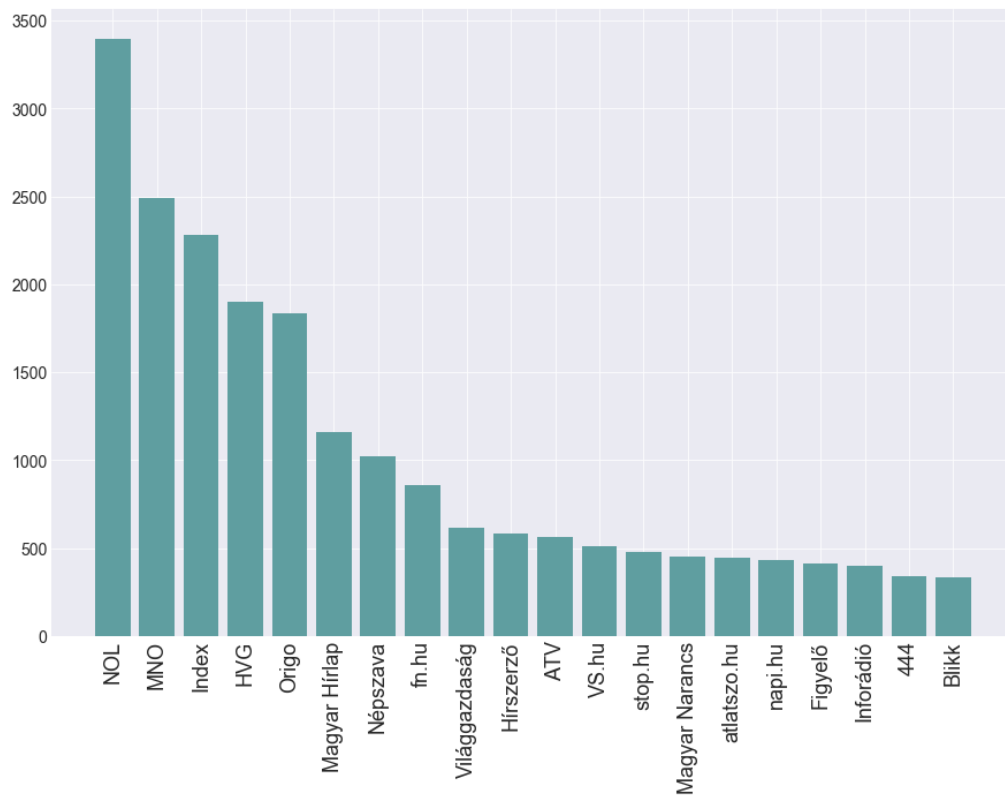
közül 310 tartalmazott hibaüzenetet. Az ismétlődő fájlok esetében minden esetben az elsőt tartottuk meg.

A lementett cikkek forrása nagyon változatos, összesen 112 portálról származnak cikkek. A portálok döntő többsége híroldal, de bulvároldalak és blogok is megjelennek közöttük.

444	Dunaújváros	IT.news	napi.hu	Sonline
Om2	Online	Jogi Fórum	Nemzeti Sport	stop.hu
168 óra	Élet és Irodalom	K-Monitor	Népszava	Szabad Föld
abcúg	Eszmélet	Kárpátinfo	NOL	Szabolcs Online
agrárszektor.hu	Fejér Megyei	KEMMA	NRG Report	SZOLJON
Átlátszó Oktatás	Hírlap	Kettős Mérce	nyugat.hu	TASZ
atlatszo.hu	Figyelő	Kisalföld	nyugatmagyar.hu	TEOL
ATV	fn.hu	Kitekintő	Origo	totalcar
bács-kiskun	Gazdasági Rádió	Klubrádió	origo.hu	TV2
online	globusz.net	online	Pécsi Stop	Új Szó
bama.hu	haon.hu	Komment.hu	Pécsi Újság	(Szlovákia)
Bank és Tőzsde	HEOL	Kreatív	Pénzcentrum	Urbanista
BEOL	Hetek	ma.hu	pestisracok.hu	(Index)
Beszélő	Heti Válasz	Magyar Hírlap	Portfolio	valasz.hu
Blikk	Hír24	Magyar Idők	Privátbankár	Vas Népe
BOON	Hir6	Magyar	Propeller	Vasárnapi Hírek
Bors	hirado.hu	Narancs	Pwc.com	vezess.hu
cink	Hírextra	Magyar Rádió	rakosmente.fides	Világgazdaság
délmagyar	Hírszerző	Manager	z.hu	VS.hu
Deutsche Welle	HírTv	Magazin	Revizor	Zalai Hírlap
DH-online	HVG	Mandiner	romaweb.hu	zóna
Direkt36	HWSW	mfor.hu	romnet.hu	ZOOM
Dunántúli Napló	Index	Mmonline	RTL Klub	
DunaTV	Inforádió	MNO	samsungsport.hu	
	ingatlanmagazin	MTV		
		Napi Gazdaság		

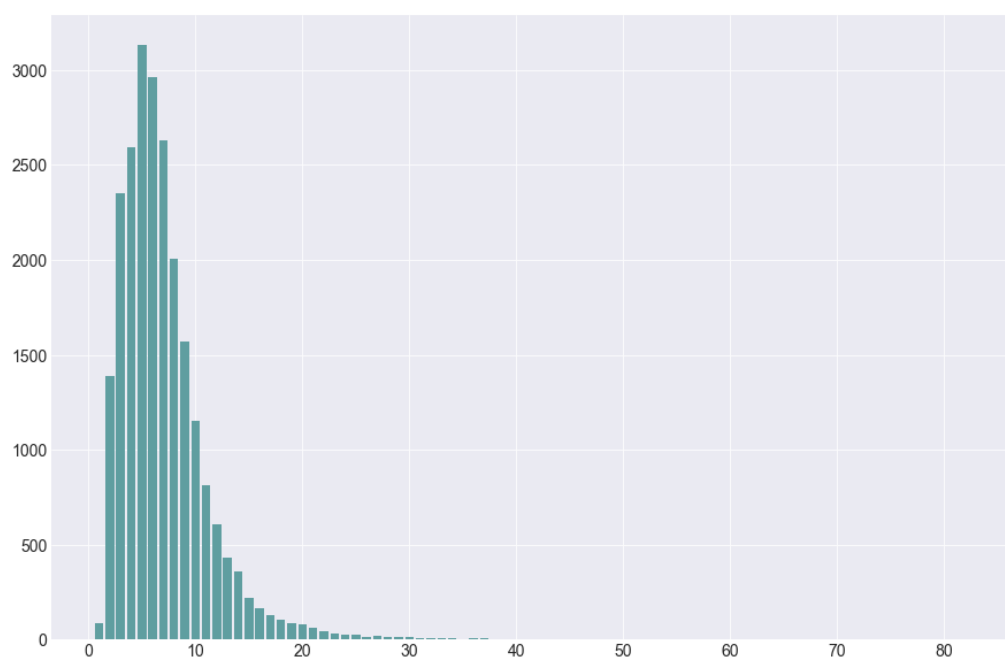
6. ábra: A cikkek forrásoldalai

Míg a fenti táblázat tartalmazza a cikkek összes forrását, a lenti ábrán a 20 leggyakoribb portált ábrázoltam.



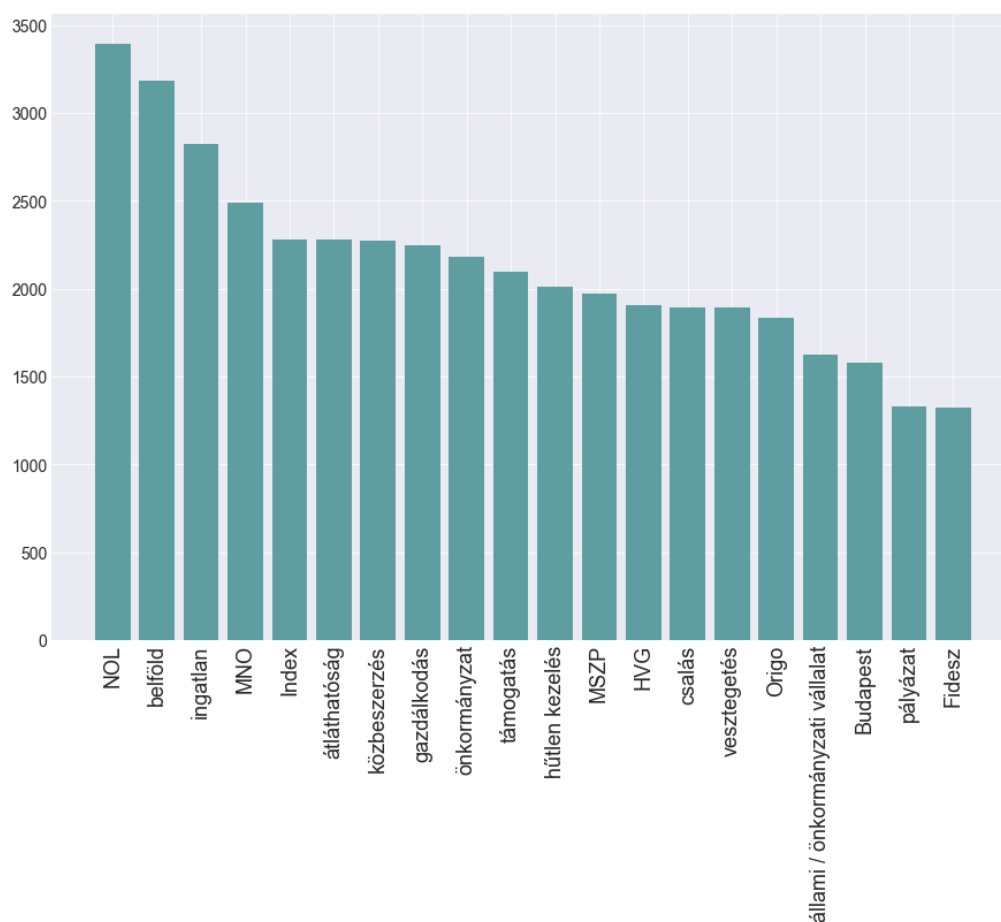
7. ábra: A cikkekhez tartozó címkék

Összesen 10167 címke van és nincs olyan cikkünk, melyhez ne tartozna legalább egy címke. Egy cikkhez átlagosan 7 címke tartozik.



8. ábra: A cikkekhez tartozó címkék számának eloszlása





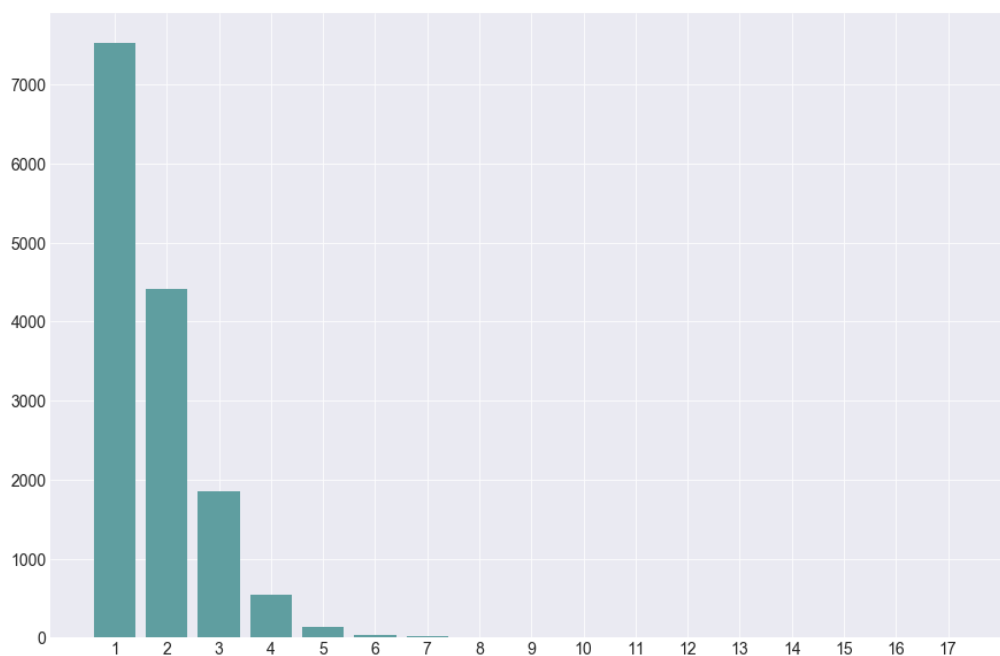
9. ábra: A cikkekhez tartozó leggyakoribb címkék

Bár nem feltétlenül működik jól a modell, ha csupán egy vagy két címke tartozik a cikkekhez, de fontos arra is figyelni, hogy ha túlzottan nagyszámú címkénk van, akkor egyrészt nagyon megnövekszik a futási idő, másrészt sok tartalommal nem feltétlenül rendelkező címkénk lesz. A cikkekhez tartozó címkék számát tehát kezelniünk kellett. Ha csupán a fent ábrázolt, leggyakoribb 20 címkét vizsgáljuk, már láthatjuk, hogy minden cikkhez hozzárendelik az önkéntesek a cikk forrását, és sok esetben viszonylag kevés jelentéstartalommal bíró címkét figyelhetünk meg. Ahhoz, hogy kezelhető mennyiségű címkét kapjunk, először kiszűrtük a cikkek forrását tartalmazó címkéket (ezek rendelkezésünkre állnak az adatbázisban, így szükség esetén később könnyen visszarakhatók, mint címke), valamint figyelembe vettük a címkék gyakoriságát, és csak azokat hagytuk meg az elemzéshez, melyeket az önkéntesek legalább 10 dokumentumhoz hozzárendelték.

A címkéket vizsgálva feltűnő volt, hogy rengeteg tulajdonnév (személynevek, intézménynevek stb.) tartozik a cikkekhez, amik a cikkekben is említve voltak. A címkék tisztításának következő lépésében tehát ezektől próbáltunk megszabadulni. A dolgozatomban a III. 2. 2. 4. 1. fejezetben bemutatott névelem-felismerő (NER) eszközzel szedtük ki a címkék közül azokat a névelemeket, melyeket a cikkek is tartalmaztak. Ezt követően két külső listát alkalmaztunk a tulajdonnevek és földrajzi nevek felderítésére<sup>31</sup>.

Ezt követően összevontam azokat a címkéket, melyek gyakran<sup>32</sup> fordulnak elő együtt. Ha például A és B címke százszor fordul elő külön, együtt pedig 110-szer, vagy 90-szer, ezt még gyakori együttjárásnak tekintem.

A címkék tisztítása után csak 147 címkénk maradt, és az egy cikkhez tartozó maximális címkeszámot sikerült 17-re visszaszorítani, az 10. ábrán látható, hogy valóban kezelhető mennyiségű címke tartozik egy-egy cikkhez.

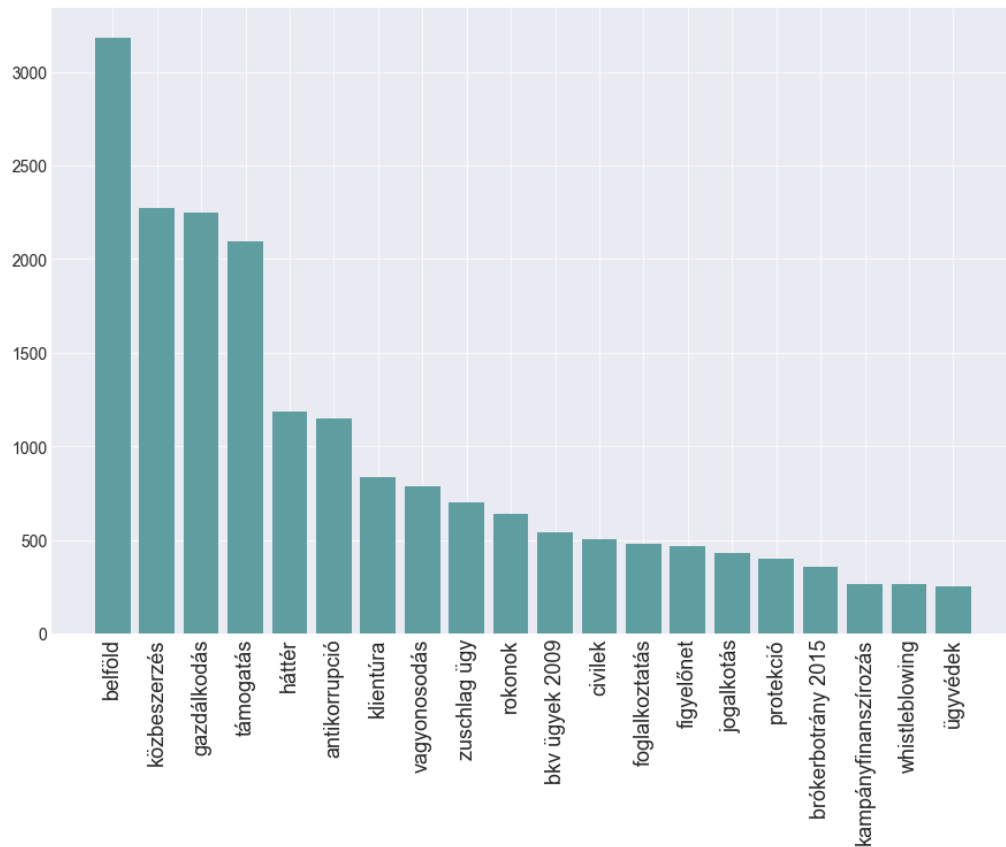


10. ábra: A cikkekhez tartozó címkék számának eloszlása (szűrt címkék)

<sup>31</sup> <http://www.nytud.mta.hu/oszt/nyelvmuvelo/utonevek/index.html> és [http://www.topomap.hu/downloads/telepuleskereso\\_web.xls](http://www.topomap.hu/downloads/telepuleskereso_web.xls)

<sup>32</sup> A gyakoriság meghatározása nem automatikusan történt, az adatok fényében meghatározott heurisztikának tekinthető.

A kiszűrt címkéket bármikor hozzárendelhetjük a cikkekhez, hiszen minden esetben automatikus módszereket alkalmaztunk, kézzel nem szűrtünk a címkék listáján. A maradék címkét átfutva azt látjuk, hogy a szűrést követően főleg a korrupcióhoz valóban kötődő címkéket sikerült megtartanunk.



11. ábra: A cikkekhez tartozó leggyakoribb címkék (szűrt címkék)

Természetesen szem előtt kell tartanunk a továbbiakban, hogy a szűrésnek köszönhetően a dokumentumaink egy része (37,4%) elvesztette az összes hozzá tartozó címkét, a modell alkalmazása során ezt észben kell tartanunk. Szeretném újra kiemelni, hogy ez a szűrés nem a címkék végleges eltávolítását jelenti, nem jelent információvesztést, hiszen a névelemek, amiket töröltünk automatikusan kinyerhetők a szövegekből. Azok a címkék, amik megmaradtak szorosan kapcsolódnak a korrupció témájához, és a tartalomelemzés szempontjából azért fontos a szűrés, mert arra lehetünk kíváncsiak, hogy a megmaradó címkékhez milyen topikok kapcsolódnak.

### **2. 3. Cikkek legyűjtése**

A cikkek letöltéséhez kiindulásként az összegyűjtött URL-eket használtam. Az automatikus adatgyűjtés kapcsán különböző kifejezéseket használunk:

- *Crawling* = belső és külső linkek feltérképezése
- *Scraping* = a crawling során keletkezett adatok kinyerése
- *Parsing* = részekre bontja a scraping során kinyert adatokat

A cikkek címét és szövegeit a Newspaper<sup>33</sup> csomaggal töltöttem le (scrape-eltem) és strukturáltam (parse-oltam) a megadott URL-ről. A csomag lehetőséget adott arra, hogy ne töltssem le a cikk teljes oldalát, hanem csak a számomra releváns részeket gyűjtöttem ki, a cikkek címét és szövegét mentettem el.

Azért tartottam fontosnak, hogy ne csak a szövegeket tartsuk meg, mert sok esetben a cím is releváns információt tartalmazhat, elképzelhető, hogy a szövegben már csak visszautalnak a címben megfogalmazottakra.

Több, mint 18000 cikket töltöttünk le, a futás során 301 hibát kaptunk. A letöltött cikkből 15193 egyedi tartalom állt a rendelkezésünkre, tehát kb. 3000 duplikátumot kaptunk.

Az adatbázisban gyakran fordult elő olyan URL, ami már nem létező cikkekre mutatott, a duplikátumok egy részét a 404-es hiba okozta. A duplikátumokból minden esetben csupán egyet tartottunk meg, a hibaüzenetet tartalmazó cikkeket<sup>34</sup> az elemzésből kizártuk.

### **2. 4. Adatfeldolgozás, adattisztítás**

A cikkek legyűjtését követően a megmaradt egyedi tartalmakon kezdtem meg a szövegek előfeldolgozását. Az előfeldolgozás egyes lépései egymással kölcsönhatásba lépnek és befolyásolják a rendszer teljesítményét. Amiatt is kulcsfontosságú,

---

<sup>33</sup> <https://github.com/codelucas/newspaper/>

<sup>34</sup> Melyeknek tartalma: "404 a keresett oldal nem található! Hiba a keresett URL, vagy az oldal már nem létezik!"

mert a hibák javítása később gyakran nagyon költségesek lehet, így megéri az elemzés előtt kezelni őket (Rehurek, 2011).

A dolgozatomban egy kísérletet végzek: három korpuszt különítettem el, az előfeldolgozás három stádiumának megfelelően:

1. Az első korpuszon csak szótövezést és szófaj szerinti szűrést végeztem
2. Az előző, szótövezett, szófaj szerint szűrt korpuszban, megkerestem az automatikusan kinyerhető névelemeket, ezeket egyesítésével jött létre a második korpusz
3. A harmadik korpusz a teljes előfeldolgozási folyamaton végigment: a szótövezett, szófaj szerint szűrt korpuszon a névelemek felismerése is megtörtént, és ezeken felül a kollokációkat is megkerestem a szövegekben.

A következőkben a három lépést: a névelemek felismerését, a szótövezést, és a kollokációk keresését szeretném bemutatni. A célom ezzel a kísérlettel annak megállapítása, hogy mi az a minimális előfeldolgozás, amivel élni kell egy optimalizált és használható modell létrehozásához. Azért fontos ezt vizsgálni, mert a korpuszok előfeldolgozása idő és erőforrásigényes, és mélyebb tapasztalatot igényel az NLP területén.

#### **2. 4. 1. Named Entity Recognition**

A *Named Entity Recognition* vagy *Named Entity Extraction* tulajdonképpen nem más, mint a névelemek felismerése. A folyamat során megkeressük és azonosítjuk az entitásokat, azaz a szövegben szereplő tulajdonnevek, földrajzi nevek, és intézmények vagy szervezetek neveinek különböző előfordulásait, és ezeket egységes alakra hozzuk.

A feladat során a dbpedia spotlight<sup>35</sup> eszközt használtam, ami sok másik nyelv mellett magyar szövegekre is alkalmazható. Az eszköz azokat a kifejezéseket ismeri fel, melyeknek van a Wikipedián szócikke. A spotlight annotate függvénye a szövegünket az általunk beállított paraméterek mentén felcímkézi. A függvénynek két paramétere van, a *confidence* és a *support*.

---

<sup>35</sup> A dbpedia spotlight egy online demo verziója kipróbálható itt: <http://demo.dbpedia-spotlight.org/>

Mendes et al. cikke alapján a confidence paraméter 0 és 1 között vehet fel értéket, és figyelembe veszi a kontextualitást. Ha a paraméternek magas értéket adunk, kockáztatjuk, hogy elveszítünk néhány helyes megoldást, de a bizonytalanságot minimalizálni tudjuk. A support paraméterrel megadhatjuk, hogy a minimum hány bemenő linkkel rendelkező a DBpedia forrásokot annotáljuk (Mendes – Jakob – García-Silva – Bizer, 2011).

Az előfeldolgozás során a confidence értéket 0,2-re, a supportot pedig 10-re állítottam, próbálkozással választottam ki az értéket, újra és újra ellenőriztem, hogy milyen névelemek lettek lecserélve, és melyek nem.

A szövegek feldolgozása során a következő formában kapjuk vissza az annotált névelemeket:

```
{'URI': 'http://hu.dbpedia.org/resource/Demszky_Gábor', 'support': 62, 'types':  
'DBpedia:Agent,Schema:Person,Http://xmlns.com/foaf/0.1/Person,DBpedia:Per  
son', 'surfaceForm': 'Demszky Gábor', 'offset': 544, 'similarityScore': 1.0,  
'percentageOfSecondRank': 0.0}
```

```
{'URI': 'http://hu.dbpedia.org/resource/Debrecen', 'support': 3453, 'types':  
'Schema:Place,DBpedia:Place,DBpedia:PopulatedPlace,DBpedia:Settlement',  
'surfaceForm': 'debreceni', 'offset': 228, 'similarityScore': 0.999999999736247,  
'percentageOfSecondRank': 2.1613937044627336e-11}
```

```
{'URI': 'http://hu.dbpedia.org/resource/Szabad_Demokraták_Szövetsége',  
'support': 429, 'types':  
'DBpedia:Agent,Schema:Organization,DBpedia:Organisation,DBpedia:PoliticalPa  
rty', 'surfaceForm': 'SZDSZ', 'offset': 1315, 'similarityScore': 1.0,  
'percentageOfSecondRank': 0.0}
```

```
{'URI': 'http://hu.dbpedia.org/resource/Parlament', 'support': 436, 'types': '',  
'surfaceForm': 'parlamenttől', 'offset': 661, 'similarityScore': 1.0,  
'percentageOfSecondRank': 0.0}
```

A fenti példákból látható, hogy nekünk az URI és a surfaceForm lesz információértékű, a folyamat eredményeként kicseréltem a szövegekben a felismert entitásokat a Spotlight által javasoltra, így egységesítve az egyes névelemek különböző előfordulásait.

Így elkészült az első korpuszom, melyen folytathattam a további előfeldolgozást.

#### **2. 4. 2. Szótövezés, szófaj szerinti szűrés**

A következő lépés a szövegek szótövezése<sup>36</sup> és szófaj szerinti szűrése, ezt a lépést végrehajtottam a nyers szövegeken, és az előző lépésben kiírt szövegeken is.

A szótárban szereplő kifejezések szótövezése egyfajta normalizálás, mellyel egységes alakra hozzuk a kifejezéseinket, hiszen a morfológia szabályainak megfelelően a szótárban szereplő kifejezések alakjai eltérőek lehetnek. Ezáltal csökkenthetjük a szótárunk méretét. A bag-of-words alapú modellünk, ahogy az elméleti bevezetőben említettem, nem veszi figyelembe a szavaink sorrendjét, a szótövezés segítségével azonban megtalálhatjuk az összetartozó, de különírt kifejezéseket (pl. elváló igekötők).

A szótövezéshez a Precognox Naive Lemmatizerét használtam, ami a Magyar Nemzeti Szövegtár szógyakorisági gyűjteményét<sup>37</sup> használja, szótár alapon működik, és paraméterként kisbetűs szavakat vár. A Naive Lemmatizer előnye, hogy gyors, hátránya viszont, hogy nem minden esetet kezel jól, igék esetében például nem túl jól dolgozik, ez számunkra nem jelent gondot, mert csak az igéket nem tartjuk meg.

A szótövezés mellett szófaji szűrést és egy elsőkörös stopszavazást is végeztem. Mindkét feladathoz a Spacy<sup>38</sup> csomag magyar nyelvre implementált modelljét alkalmaztam.

A szófaji szűrés során megadhatjuk, hogy mely szófaji kategóriába sorolt kifejezéseket szeretnénk a korpuszunkban vizsgálni. Alapvetően a főnév, melléknév és ismeretlen<sup>39</sup> kategóriák szoktak a korpuszba kerülni, az igék kiírása változó. Esetemben a korábban azonosított és lecserélt névelemek az általános szűrés során

---

<sup>36</sup> A szótövezést és a lemmatizálást szinonimaként használom

<sup>37</sup> <http://metashare.nytud.hu/repository/browse/hungarian-word-frequency-list/c1e16f3aaaaf11e3aa7c68b599c26a0615bcb16635874754bcf6b5717986c02e/>

<sup>38</sup> <https://github.com/oroszgy/spacy-hungarian-models>

<sup>39</sup> ADJ, NOUN, X

(a formai megjelenésük miatt) kimaradtak a korpuszból, így a korpuszba kerülő szófaji csoportok listáját bővítenem kellett.

A stopszavazás során megszabadulunk azoktól a kifejezésektől, melyek tartalmi jelentéssel nem rendelkeznek (Tikk et al., 2007).

A bag-of-words modell elkülöníti a tartalomszavakat (*content words*), mint a főnevek, melléknevek a funkciószavaktól (*function words*), ezek közé tartoznak például a kötőszavak. A modell nem veszi figyelembe azokat a nyelvtani információkat (pl. szavak sorrendje), melyekkel a funkciószavak rendelkeznek, így ha bent hagyjuk őket a korpuszban, növeljük a zajt, és ezzel topikmodellek esetén is rontjuk a diszkriminálást. Az alacsony gyakoriságú szavak széthúzzák a topikjainkat, ezért inkább megszabadulunk azoktól a kifejezésektől, melyekre az adott környezetben nincs szükségünk.

Ahogy említettem, az előfeldolgozás ezen szakaszában a Spacy beépített stopszólistáját alkalmaztam, ami a magyar nyelvre jellemző kifejezéseket tartalmazza, mint a névelőket, a személyes névmásokat, a kötőszavakat és egyéb funkciószavakat. A stopszavazást a szövegek előkészítésének egy későbbi szakaszában kiegészítem a korpuszra jellemző stopszavak kiszűrésével.

#### **2. 4. 3. Szignifikáns bigramok**

A következő lépés az előfeldolgozás során a kollokációk vizsgálata, vagyis a szignifikáns bigramok megkeresése. A bigram egy gyakran együtt előforduló szópár (szótöbbes, azaz n-gram). A magyar lexikográfiában állandósult szókapcsolatként, idiómaként szoktak rá hivatkozni (Reményi 2010).

A szignifikáns bigramok kigyűjtéséhez az NLTK könyvtár CollocationFinder függvénye nyújtott segítséget, megtalálásukhoz a chí-négyzet statisztikát alkalmaztam. Első lépésben kiírtam az 500 legjobb (*nbest*) bigramot, az előfordulási alakjukban, valamint az összevont alakban, majd egy következő lépésben a szövegekben szereplő kollokációkat lecseréltem a listában szereplő összevont alakra. A lépések szétbontása azért volt szükséges, mert így manuális módon („*human in the loop*” módszerrel) ellenőrizhetők és szűrhetők a kigyűjtött bigramok.



Példák szignifikáns bigramokra:

adományozóknál\_kedvezményezettünket

agrárirányítás\_szakhatóságainak

agrárpályázatok\_tájékoztói

auditori\_felelősség

barlangi\_üreg<sup>40</sup>

beszedhetetlen\_adórészeket

blaha\_lujza

#### **2. 4. 4. A tartalomszavak összeállítása**

Az előfeldolgozás utolsó lépésében két szólistát készítettem el, ami a végső korpuszunk tartalmának előállításához nyújt segítséget.

Először a szógyakorisági listát készítettem el, ami a teljes korpusz összes szavának gyakoriságát tartalmazza. Erre a listára azok a szavak kerültek fel, melyek esetében a karakterlánc hossza legalább három és az összes dokumentumból legalább ötvenben szerepelnek. A másik segédlista a szavak dokumentumgyakoriságát vizsgálja, tehát azt, hogy az adott kifejezés hány dokumentumban fordul elő. Azokat a kifejezéseket, amik nagyon kevés, vagy nagyon sok dokumentumban szerepelnek a későbbiekben kizárjuk az elemzésből<sup>41</sup>.

A fenti két dokumentum alapján készítettem el azt a listát, ami a végső korpuszban megjeleníteni kívánt szavakat tartalmazza. A tartalomszavak kezdetben tehát a fentiek alapján a stopszavazást követően megmaradt,

- legalább három karakter hosszú kifejezések,
- melyek az összes dokumentumban legalább 50-szer szerepelnek,
- az összes dokumentumból legalább ötvennek részét képezik.

Ezekhez a tartalomszavakhoz hozzárendeljük az összes szóhoz viszonyított arányukat, és még ezt is megsűrjünk úgy, hogy azok a kifejezések kerüljenek a végső korpuszba, melyek se nem túl gyakoriak, se nem túl ritkák. Azt, hogy ezt a két küszöbértéket hol

---

<sup>40</sup> Ahogy a példa is mutatja, természetesen nem csak olyan kollokációk fordulhatnak elő a szövegben, melyek a témához szorosan kapcsolódnak.

<sup>41</sup> A kizárás okát a fejezet 2. 4. 2. alfejezetében kifejtettem.

határozzuk meg mindig az adott korpuszon múlik. Általában erre a szűrésre érdemes akkor visszatérni, ha már az első modellverzióinkat elkészítettük, létrehoztuk a topikjainkat, ekkor látjuk, hogy mely kifejezések azok, amik nem rendelkeznek hozzáadott értékkel, nem differenciálnak. A küszöbérték beállításához mindig figyelembe kell venni az első két lépésben összeállított listát, ezek összehangolásával tudjuk megadni az adott korpuszt jól leíró tartalomszavakat. Esetemben az alsó határ a 0,05, a felső határ pedig a 0,2 volt. A felső határ meglepően alacsonynak tűnhet, de a szóeloszlás egy hatványfüggvény, Zipf-eloszlást követ, azaz leegyszerűsítve azzal indokolhatjuk az alacsony felső küszöböt, hogy a leggyakoribb szavak mindenhol előfordulnak. (Tikk et al., 2007)

### 3. Az Akták cikkeinek elemzése az Author-Topic Modellel

#### 3.1. A korpusz előállítása

A teljes korpusz 15195 dokumentumot tartalmazott. Azonban fent bemutatott előfeldolgozást követően több dokumentum elvesztette a hozzá tartozó címkék mindegyikét. A modell szempontjából fontos, hogy a korpuszunk csak olyan dokumentumokat, azaz cikkeket tartalmazzon, melyekhez szerző, esetünkben címke is tartozik. Azokat a cikkeket, melyek elvesztették a címkéjüket a korpuszból eltávolítottam, így 9519 cikkem maradt, melyekhez 147 címke tartozik.

A cikk-címke párosainkat egy olyan szótárban tároljuk, ahol a szerzők nevei (a címkék) a kulcsok és az értékek azon dokumentumok (cikkek) listái, melyek az adott címkével el lettek látva. Ezt a reprezentációt a Gensimben `author2doc`-nak nevezzük.

Az `author2doc_full` reprezentáció tehát a következőképp néz ki:

```
{ 'kopaszi gát': ['3', '44', '86', '149', '2071', '2385', '2388', '2738', '2793', '2801',  
'2871', '2920', '2992', '3073', '3093', '3187', '3355', '6115', '6844', '7047',  
'11508', '12759', '12766', '13574', '13610', '13611', '14446', '15508', '15542',  
'15566', '15550', '15577', '15716', '15895', '16098', '1', '3380', '15606', '32270',  
'34214', '38399', '38406', '38439', '38476', '38498', '38537', '38605', '39156'],  
... }
```

A fenti szótár a címkéhez tartozó cikkek nevét tartalmazza, melyeket egyedi azonosítóra (ID-ra) cserélünk:

```
{ 'kopaszi gát': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,  
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,  
42, 43, 44, 45, 46, 47],  
... }
```

A Gensimben használt másik reprezentáció a dokumentumok szavait tartalmazó szótár, a `dictionary_full`, egy 501 egyedi kifejezés (*unique token*) listája.

A `corpus_full` egy, már sokat emlegetett bag-of-words reprezentáció, ami tuple-ként tartalmazza a szó-dokumentum párokat:

```
corpus_full:  
... ,(203, 3), (204, 2), (210, 1), (218, 6), (220, 5), (223, 5), (226, 1), (240, 3), ...
```

A teljes adathalmazt tanuló és teszt adatokra bontottam, 80 – 20% arányban.

A szétbontás során két nagyon fontos eshetőségre kell gondolnunk:

1. Amikor szétválasztjuk a teljes adathalmazt, elképzelhető lehet, hogy egy szerzőhöz a bontást követően nem fog már dokumentum tartozni, ezeket a szerzőket töröljük a tanuló és teszt esetekben is.
2. Amikor a teszt korpuszt készítjük arra kell figyelniünk, hogy biztosan csak olyan kifejezések kerüljenek a korpuszba, amik a tanuló szótárban is benne voltak.

A 2. megjegyzést figyelembe véve fontos hangsúlyozni, hogy a modell nem alkalmas új címkék ajánlására, csak a meglévő címkekészletből tud dolgozni. Ahhoz, hogy a teljes automatizáláshoz egy lépéssel közelebb kerüljünk a cikkekben előforduló kulcsszavak keresése egy megoldás lehet. Az automatikus kulcsszókinerő figyelmeztethet arra, ha egy szövegekben előkerülnek olyan kifejezések, melyekkel a címkék gyűjteményét bővíthetnénk.

A tanuló korpuszba így 7615 dokumentum került 145 címkével, a teszt korpuszba pedig 1904 dokumentum 134 címkével.

A tanuló-teszt bontásra a későbbiek során azért lesz szükségünk, mert ennek segítségével szeretnénk majd megkeresni az optimális topikszámot.

### **3.2. A modell tanítása**

A modellnek paramétereit közül elsősorban azokat szeretném a Gensim dokumentációja<sup>42</sup> alapján ismertetni, melyekkel a modellkészítés során foglalkoztunk.

- *num\_topics*: a keresett látens témák száma
- *id2word*: az id-kból (integerből) való leképezés a szavakra (szövegre). A szótár méretének meghatározására, hibakeresésre és a topik visszaadására szolgál.
- *author2doc*: olyan szótár, ahol a szerzők nevei a kulcsok és az értékek a dokumentumok listái

---

<sup>42</sup> <https://radimrehurek.com/gensim/>

- *doc2author*: olyan szótár, ahol a kulcsok a dokumentumazonosítók, és az értékek a szerzők neveinek listái. Tehát ez a fordított leképezése az *author2doc*-nak.
- *passes*: annak a száma, ahányszor a model végigfut az adatokon
- *iterations*: annak a maximuma, ahány loopot csinál a modell az egyes dokumentumokon (M-lépés). Ha elérjük az elméleti részben tárgyalt konvergenciát, az iteráció leáll.
- *chunksize*: „mini-tételek” (*mini-batches*), a méretét szabályozza. A *chunksize* segít, hogy egyszerre csak annyi adattal dolgozzon a modell, ami belefér a memóriába, vagy amennyin gyorsan el tudja végezni az *iterations* paraméterben megállapított loopokat.

Az *alfa* és az *eta* hiperparaméterek, amelyek befolyásolják az *author-topic* (*theta*) és a *topic-word* (*lambda*) eloszlását.

Mindkét paraméter alapértelmezése szimmetrikus ( $\frac{1.0}{\text{num\_topics}}$ ) prior.

Az *alfa* értéke lehet prior, a saját választásunk szerint, aszimmetrikus és automatikus, az *eta* lehet

- egy skalár (szimmetrikus prior a topik/szó eloszláson),
- vektor (a *num\_words* reprezentációja),
- mátrix (a *num\_topics* x *num\_words* reprezentációja), illetve lehet
- automatikus.

Az automatikus *alfa* és *eta* egyaránt az adatainkhoz leginkább illeszkedő értéket adja vissza.

Gensimben elérhető Author-Topic Modelhez három olyan paraméter tartozik, ami a futási időt befolyásolja: a chunkok mérete, a number of passes és az iterációk száma.

1. A chunkok méretének növelése csökkenti a tanulási időt, érdemes lehet tehát nagyobb chunkmérettel próbálkozni. Azonban szem előtt kell tartanunk, hogy bár a futási idő hosszabb, de a chunkok méretének csökkentése javítja a topikbesorolást.
2. Ha egy pass futási ideje a korpuszon *t*, akkor *P* pass futtatása  $P * t$ .

3. Az iterációk számával lineárisan növekszik a futási idő.

Azt, hogy hány pass-re van szükség a konvergálásig függ a chunkok méretétől, és az iterációk számától, tehát fontos, hogy egy, a modell illeszkedése és a futási idő közötti, kiegyensúlyozott paraméterbeállítást határozzunk meg.

A tanítás során hasznos paraméter még a *random\_state*, különböző értékekkel az adataink különböző reprezentációit kapjuk vissza.

A paraméterek beállítására léteznek hüvelykujj szabályok, például, hogy a chunksize körülbelül az adatok felénél, harmadánál optimális. Ha túl magas a chunksize, gyakran topikduplikátumok jelennek meg, tehát a topikokhoz tartozó szavak két vagy több esetben azonosak lesznek.

Ha a chunksize-unk 100.000, és az *update\_every* paraméter 1, ugyanazt az eredményt kapjuk, mint az 50.000 és 2 párosítással. Azért jobb a második megoldás, mert a kisebb chunksize kevesebb memóriát használ. A chunkok mérete tehát a memóriaigényre van hatással: kisebb chunkok esetén az algoritmus kevesebb memóriát igényel, viszont nagyobb méret esetén kevesebb összevonást végez, így jobb teljesítményt várhatunk (Rhurek 2011).

Az ideális paraméterek megtalálására nincsenek kimondott, bevált módszerek, a legjobb beállítás keresése (*tuning*) általában próbálkozással, megérzés alapon történik. A futtatás során érdemes bekapcsolni a *logolást*. Enélkül nem látnánk, hogy mi történik a model tanítása során, míg ha logolunk, követhetjük a folyamatokat, esetleg hibaüzenetet is kapunk, amit enélkül nem ismernénk meg. Gyakori például, hogy eleinte nem a megfelelő passes és iterations értéket adjuk meg, ilyenkor hibaüzenetet kapunk, ez is segíthet a megfelelő paraméterezés megtalálásában.

### **3. 3. Topikok**

A szakdolgozatom talán legfontosabb fejezete ez, melyben célkitűzéseimnek megfelelően bemutatom a címkeajánlás módszerét, majd a predikciót alkalmazva heurisztikát mutatok az optimális topikszám megtalálására. Végül a topikok elnevezésének menetét mutatom be a következőkben.

A topikmodellek társadalomtudományi alkalmazása megkönnyíthetné a tartalomelemzést, mégis nagyon sokat könnyít, és gyorsít a folyamatokon, ha a hagyományos, kvalitatív eszközöket kvantitatív módszerekkel ötvözzük. A szubjektivitást itt sem tudjuk teljesen kizárni, de nagyon sok esetben standardizálhatóbb, általánosíthatóbb eredményt kapunk, mintha a hagyományos módszerekre támaszkodnánk.

A látens topikok feltárása a címkézéshez nem kapcsolódik szorosan, de jól kiegészíti azt, és a kutatás témájának függvényében akár jóval hangsúlyosabb részét is képezheti egy elemzésnek, mint esetemben. Ezért is tartom fontosnak, hogy bemutassam, én milyen módszerekkel igyekeztem leküzdeni az elemzés során felmerülő akadályokat.

### **3. 3. 1. Az optimális topikszám keresése**

Az optimális topikszám megtalálása nagy kihívás a topikmodellek készítése során, ezért volt a szakdolgozatom egyik fő célja, hogy egy alternatív heurisztikát nyújtsak az optimális topikszám megtalálására, és megkönnyítsem a topikok elnevezését.

A perplexitás az a mérőszám ami a szakirodalom szerint alkalmas lenne az optimális topikszám keresésére, így a Tudományos Diákköri Konferenciára írt dolgozatomban ez volt a mutató, amit én is alkalmazni próbáltam – sajnos eredménytelenül<sup>43</sup>. A következőkben röviden írok a perplexitásról, majd bemutatom a módszert, amit a dolgozatomban alternatívaként alkalmaztam.

A perplexitást gyakran használják egy nyelvmódel hasznosságának mérésére.

A  $W=w_1, w_2, \dots, w_N$  szavakat tartalmazó tesztkészlet esetében a módel perplexitása:

$$PP(W)=P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}}$$

---

<sup>43</sup> Amikor megpróbáltuk kideríteni, hogy vajon miért nincsen egy olyan törés az ábrán, ahol az optimális topikszámot találnánk, kiderült, hogy probléma fedezhető fel a Python implementációban: a perplexitás a dokumentumok számával csökken.

A perplexitás normalizálva van a szavak számával. Bár a perplexitás nem határozza meg egyértelműen a nyelvmodell hasznosságát, de alkalmazható mérőszám a nyelvi modellek összehasonlításához (Jurafsky – Martin, 2008).

Az előző szakaszban a tanuló-teszt bontás azért volt szükséges, mert a modellünket a tanuló adatokon készítjük el, de a mutatót a teszt adatokból számoljuk. A perplexitás nem bizonyult a korpuszomon alkalmazható mutatónak.

A TDK dolgozatomban nem állt a rendelkezésemre annotált adathalmaz, de a mostani korpuszom *gold standard* korpusznak tekinthető.

2 és 50 közötti topikszámra készítettem el azonos paraméterbeállításokkal a modellem, ebben az intervallumban fogom keresni az optimális topikszámot.

Az összes cikkhez megvan nekünk az összes címke, amit az önkéntesek rendeltek a cikkjeinkhez, ezeket nevezzük *true labelnek*.

A prediktáláshoz nem az ATM-et, hanem hasonlósági mutatókat használtam, melyekkel a dokumentumok szóeloszlását vetjük össze a szerzőkhöz tartozó szóeloszlásokkal. Két mutatót próbáltam ki: a Helinger és a Jensen mutatókat. A hasonlósági függvény minden, 0-nál nagyobb hasonlóságot mutató címkével tér vissza. Első próbálkozásra minden cikkhez öt címkét prediktáltunk, de ez nem tűnt jó módszernek, mert az egyes hasonlósági értékek között vagy nincs, vagy nagyon minimális a különbség. Így végül az összes szerzőt megtartottunk, aki hasonlóságot mutat a dokumentumokhoz.

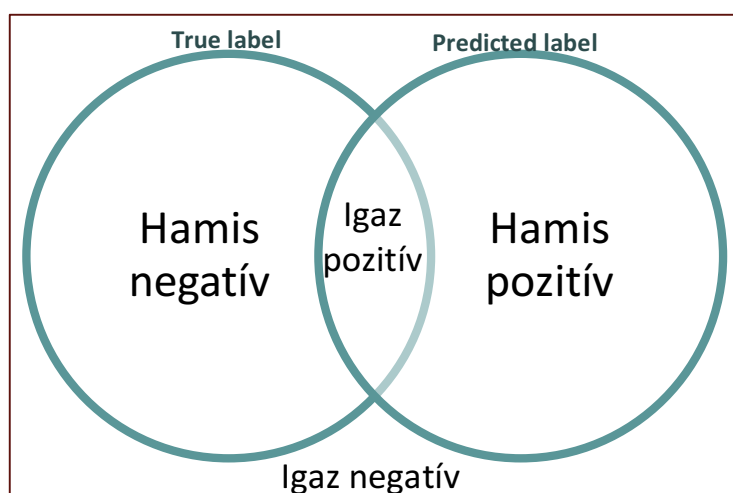
A K-Monitor önkénteseinek ezzel a predikcióval ajánlunk fel címkéket, melyek közül ők kiválaszthatják azokat, melyekkel a cikket el szeretnék látni. Azért jelenthet ez számukra segítséget, mert így egy irányított szótárból származó, konzisztens címkék közül választhatnak, így gyorsabbá és egységesebbé válhat a címkézés. A módszer csak a már meglévő címkék közül tud javaslatokat adni, de egy kulcsszókinyerő segítségével új témák megjelenésére is felhívhatjuk az önkéntesek figyelmét.

A cikkekből a személynevek, helységnevek, szervezetek nevei, tehát a névszók szintén kinyerhetőek, így ezekkel is bővíthető az ajánlott címkék sora.

Csupán a modellünk segítségével tehát nem tudjuk a folyamatot automatizálni, de konzisztensebbé, gördülékenyebbé tehetjük az önkéntesek munkáját.



Minden modell esetében minden cikkhez prediktáltam címkéket (*predicted label*), és megnéztem, mennyire teljesít jól a modell a különböző topikszámok esetében. A modell teljesítésének vizsgálatát négy egyszerű, tévesztési mátrixon (*confusion matrix*) alapuló mérőszám segítségével vizsgáltam. A tévesztési mátrix a helyesen és helytelenül besorolt elemeket vizsgálja, osztályozási modellek teljesítményének kiértékelésére használják.



12. ábra: Tévesztési mátrix vizualizációja

4 mutatót vizsgáltam, melyek a 12. ábrán alapulnak, a definíciókat Tikk et al. (2007) cikke alapján határoztam meg:

- *precision*: pontosság, azt méri, hogy a címkéi mennyire relevánsak, a modell által meghatározott címkék hány százaléka egyezik meg az önkéntesek címkéivel.

$$\frac{\text{Igaz pozitív}}{\text{Igaz pozitív} + \text{hamis pozitív}}$$

- *recall*: felidézés, azt mutatja meg, hogy az önkéntesek által adott címkéket milyen arányban találta el a modell.

$$\frac{\text{Igaz pozitív}}{\text{Igaz pozitív} + \text{hamis negatív}}$$

- *F-Score*: a precision és a recall harmonikus átlaga.

$$2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

- és *accuracy* értéket:

$$\frac{\text{igaz pozitív} + \text{igaz negatív}}{\text{igaz pozitív} + \text{igaz negatív} + \text{hamis pozitív} + \text{hamis negatív}}$$

A precision volt az a mutató, ami illeszkedett az adatainkhoz és értelmezhető eredményeket adott. Mivel a prediktált címkék száma túl magas, az igazi címkék száma pedig alacsony, így a recall nem tűnt jó mutatónak.

Végül a precision-re fókuszáltunk, hogy a perplexitást helyettesíteni tudjuk. A precision eleinte azért tökéletes, mert a *get\_author* függvény, mellyel a címkéinket prediktáltuk, akkor még az összes címkével tér vissza – ami túlillesztést eredményez. A függvény azonban a topikszám növekedésével egyre rövidebb listával tér vissza, ezért alkalmazható heurisztikaként a módszerünk.

A következő ábrán kirajzoltam, hogy milyen precision értékeket kaptunk a különböző topikszámok esetében.



13. ábra: Precision értékek az 50 topik esetében

A következő lépésben kiírtam az egyes topikokhoz tartozó szavakat minden modellre. A topikszavak alapján 30 felett elkezdenek duplikátum topikok megjelenni, amik a top-n topic words duplikátumai. A duplikátumok nagyon megnehezítik a dolgunkat, hogy megkülönböztessük emberileg a topikokat.

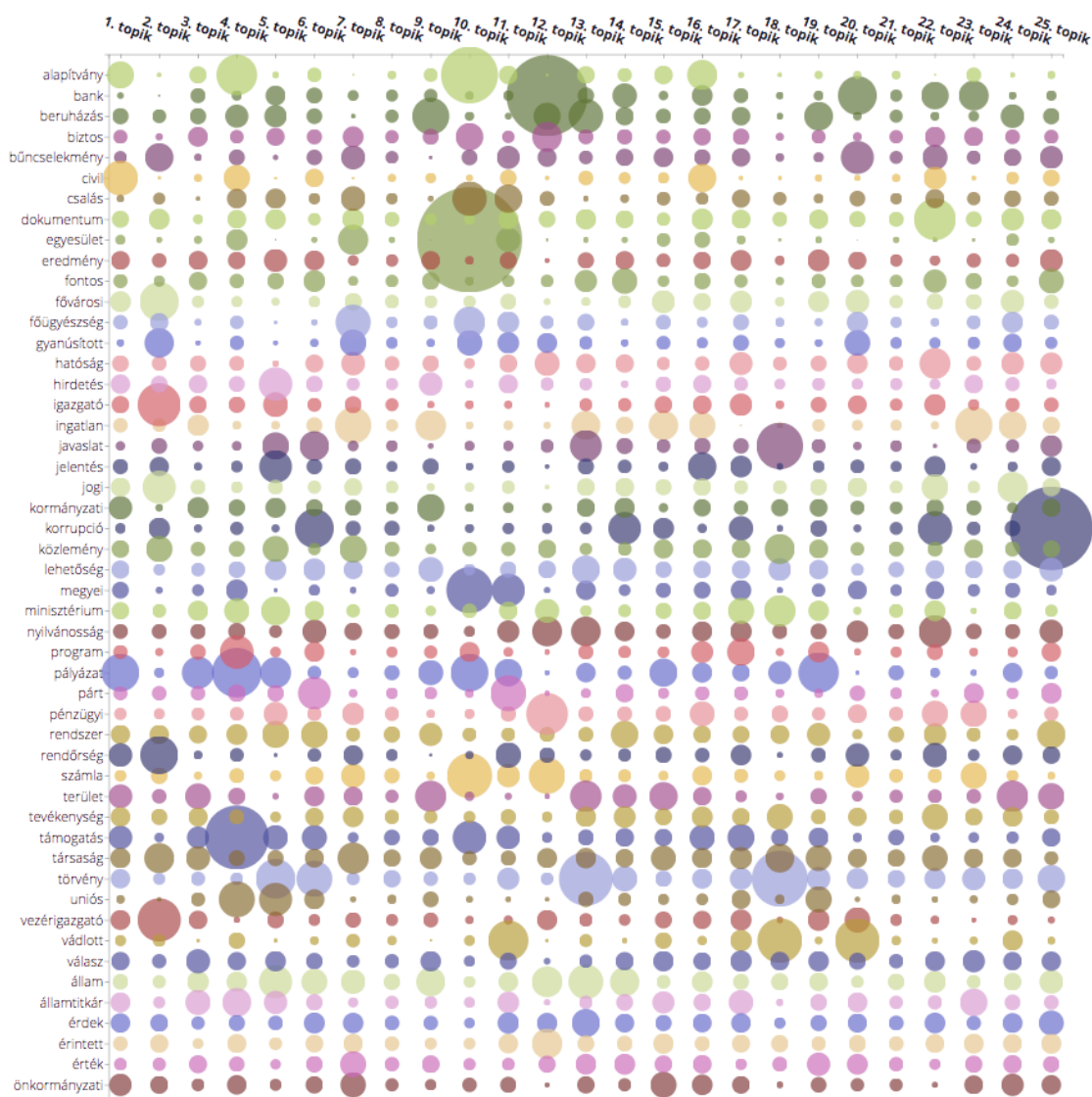
Az optimális topikszámot tehát a Jensen mutató precision értékének lokális maximuma alapján határozom meg. Figyelembe véve a fentieket (a túlillesztés lehetőségét és a duplikátum topikokat) 15 és 30 között keresem az optimális topikszámot, ami a precision alapján esetünkben a 25 lesz.

### **3. 3. 2. Topikok elnevezése**

Miután megtaláltuk, hogy hány topikba szeretnénk sorolni a cikkeinket, legtöbbször ezeket a topikokat el is szeretnénk nevezni, a tartalmuknak megfelelően. Ez is nehéz feladat, legtöbbször nem egyértelmű első ránézésre a kiírt topikszavak alapján, hogy melyik topik milyen témával foglalkozik. A topikszavak listája minden topikra ugyanazokat a kifejezéseket tartalmazza, a szavak sorrendjében van eltérés, a topikra legjellemzőbb szavak vannak legelől, minél kevésbé jellemző egy kifejezés egy topikra, annál hátrább sorolódik. A lista böngészése alacsonyabb topikszám esetében sem egyszerű, de 25 topiknál főleg átláthatatlannak tűnhet, ebben nyújthat segítséget a Bubble chartunk, amit Chuang – Manning – Heer (2012) tanulmánya, és vizualizációja inspirált.

Az ábrán az egyes szavak topikvalószínűségét ábrázoljuk. A valószínűségeket a buborékok mérete mutatja. Az interaktív vizualizációnkon a szavakat kétféleképp rendezhetjük: *saliency* és *frequency* szerint.

A saliency egy, a nyers gyakoriságnál jóval inkább megszürt korpuszon mutatja a szavak topikonkénti előfordulását, így nagyobb eltérések figyelhetők meg az ábrán, míg a frequency a szavak valódi gyakoriságát mutatja, egy jóval általánosabb stopszavazás után.



14. ábra: A term saliency mutató Bubble charton<sup>44</sup>

A saliencyt amit Chuang – Manning – Heer (2012) tanulmánya alapján implementáltuk, a mutató azt méri, hogy egy adott kifejezés mennyire jellemzi az adott topikot egy bármilyen random kiválasztott szóhoz képest. Egy adott  $w$  szóhoz kiszámítjuk a  $P(T|w)$  feltételes valószínűséget, azaz annak valószínűségét, hogy a megfigyelt  $w$  szót

<sup>44</sup> Az y tengelyen szereplő kifejezések: [alapítvány, bank, beruházás, biztos, bűncselekmény, civil, család, dokumentum, egyesület, eredmény, fontos, fővárosi, főügyészség, gyanúsított, hatóság, hirdetés, igazgató, ingatlan, javaslat, jelentés, jogi, kormányzati, korrupció, közlemény, lehetőség, megyei, minisztérium, nyilvánosság, program, pályázat, párt, pénzügyi, rendszer, rendőrség, számla, terület, tevékenység, támogatás, társaság, törvény, uniós, vezérigazgató, vádlott, válasz, állam, államtitkár, érdek, érintett, érték, önkormányzat]

a  $T$  topik generálta.  $P(T)$  annak a marginális valószínűsége, hogy bármely  $w'$  véletlenszerűen kiválasztott szót a  $T$  topik generált.

A  $w$  szó disztinktív képességét (jel.:  $\text{dist}$ ) a  $P(T|w)$  és a  $P(T)$  közötti a Kullback-Leibler divergenciával határozzuk meg:

$$\text{dist}(w) = \sum_T P(T|w) \log \frac{P(T|w)}{P(T)}$$

Ezzel leírhatjuk, hogy mennyire informatív az adott  $w$  kifejezés a topik meghatározására, egy véletlenszerűen kiválasztott  $w'$  kifejezéshez képest.

Ha egy szó minden témában előfordul, kevésbé járul hozzá egy adott dokumentumban keveredő topikok feltárásához, a szó disztinktív képessége alacsony.

$$\text{saliency}(w) = P(w) \times \text{dist}(w)$$

A két mutatót (frequency és saliency) vizsgálhatjuk bigramokon vagy egyszerűen a szavak listáján is. A fenti ábrán az 50 legmagasabb saliency értékkel rendelkező kifejezés szerepel.

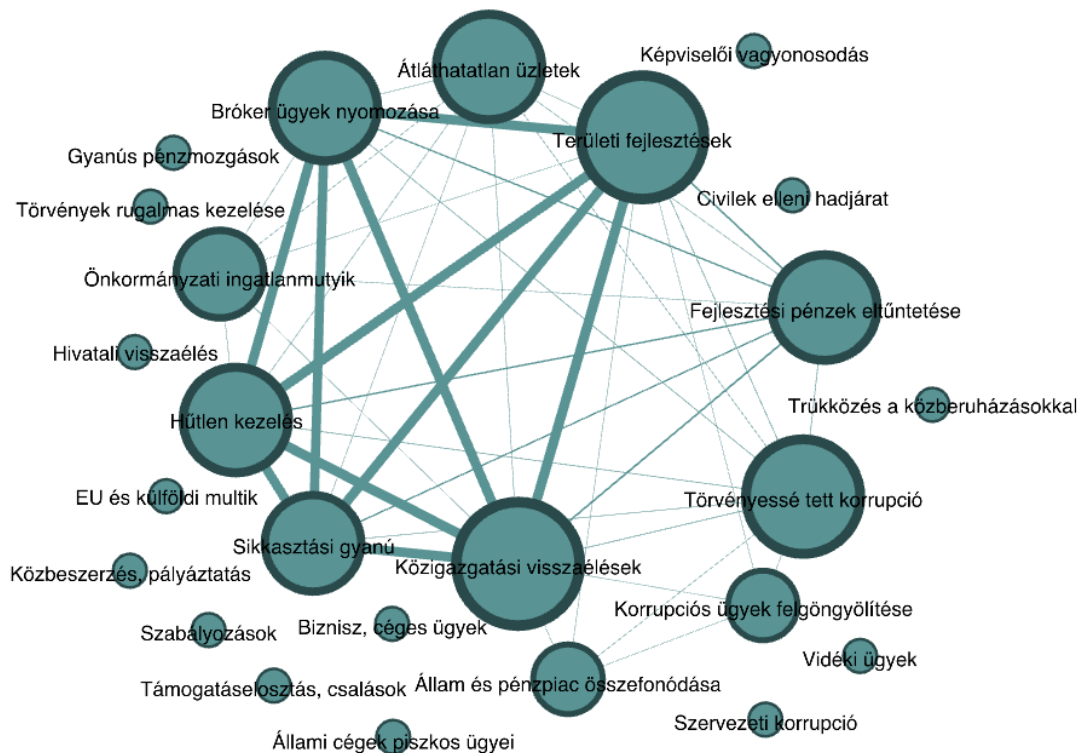
A topikok elnevezése során nem kell csak és kizárólag a topikszavakra hagyatkoznunk, a cikkeket rendezni tudjuk a topikok szerint is. Mivel Gensimben nincs implementálva, módszer, mellyel visszakaphatnánk a különböző topikokhoz tartozó cikkeket, így mi készítettünk erre is egy alternatívát: a dokumentum szerzőinek topikeloszlását átlagoljuk. A topikokhoz tartozó cikkek kiírása során megadhatjuk, hogy mit szeretnénk, a kiírt cikkek milyen mértékben jellemezzék az adott topikot. Nagyon sokat segít, hogy bele tudunk olvasni a cikkekbe topikonként, mert a Bubble charton is lehetnek olyan szavak, amik nagyon elhúzzák a topikot, de irányítják a figyelmünket a cikkek olvasása során. Így a következő neveket adtam a topikjaimnak:

- |                              |                                  |
|------------------------------|----------------------------------|
| 1: Vidéki ügyek              | 7: Hivatali visszaélés           |
| 2: Szervezeti korrupció      | 8: Trükközés a közberuházásokkal |
| 3: Képviselői vagyonosodás   | 9: Szabályozások                 |
| 4: EU és külföldi multik     | 10: Civilek elleni hadjárat      |
| 5: Közbeszerzés, pályáztatás | 11: Támogatáelosztás, csalások   |
| 6: Biznisz, céges ügyek      | 12: Hűtlen kezelés               |

13: Törvényessé tett korrupció  
 14: Állami cégek piszkos ügyei  
 15: Önkormányzati ingatlanmutyik  
 16: Törvények rugalmas kezelése  
 17: Fejlesztési pénzek eltűntetése  
 18: Területi fejlesztések  
 19: Gyanús pénzmozgások

20: Bróker ügyek nyomozása  
 21: Közigazgatási visszaélések  
 22: Sikasztási gyanú  
 23: Állam és pénzpiac összefonódása  
 24: Átláthatatlan üzletek  
 25: Korrupciós ügyek felgöngyölítés

A cikkekben a topikok keverednek, egy cikk több topikba is besorolható. A következő ábrán a topikok közötti kapcsolatokat szemléltetem. A hálózatban az élek azt mutatják, hogy a topikok milyen gyakran fordultak elő egy-egy cikkre leginkább jellemző topikként. Az ábrát a topikokhoz tartozó dokumentumok alapján hoztam létre, a cikkekre legjobban jellemző tíz topik között képeztünk éleket. Az élek vastagsága azt mutatja meg, hogy a topikok hány dokumentumban szerepeltek együtt.



15. ábra: A topikok közötti kapcsolatok gráfja

A dolgozatom keretein túlmutat a topikok mélyebb elemzése, a további vizsgálódás lehetőségét nyitva hagyom, mint lehetséges irány a dolgozatom folytatására.

#### 4. Az előfeldolgozás fázisainak összehasonlítása

Az adataim előfeldolgozása során három korpuszt hoztam létre, annak megfelelően, hogy a cikkek mennyire alaposan lettek előkészítve az elemzésre. Arra voltam kíváncsi, hogy mennyire eltérő eredményeket kapunk a három korpuszon. Bár szerintem nagyon fontos és érdekes kérdés, sajnos a szakdolgozatom keretei végül nem engedték, hogy olyan mélyen vizsgáljam a kérdést, mint szerettem volna.

Az első korpuszon csak szótövezést végeztem, a második korpuszon szótövezést és névelemfelismerést.

Amit az első eredményeim alapján megállapíthatunk, az az, hogy a névelemfelismeréssel kiegészített korpusz precision értéke magasabb, mint amit a csupán szótövezésen átesett korpusz esetében láthatunk, ezen a 13. ábra alapján a szignifikáns bigramok megkeresése már nem sokat változtat.

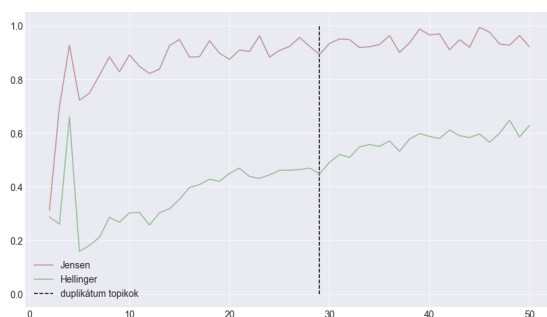
Az optimális topikszám mindkét esetben a 23, és mindkét korpuszon 29 topik után kezdenek megjelenni a duplikátum topikok.

Saliency és szógyakoriság alapján nincsenek nagy különbségek, a mellékletben látható Bubble chartok alapján (M4. és M5. ábra) hasonló topikok, jelennek meg mind a három korpusz esetében.

##### 1. Szótövezés

##### 2. Névelem felismerés és szótövezés

**Precision:**



**Optimális topikszám:**

N = 23

N = 23

**Duplikátumok megjelenése**

N = 29 után

N = 29 után

---

**Word frequency:**

['önkormányzat', 'támogatás', 'pályázat', 'fidesz', 'jános', 'törvény', 'bizottság', 'magyarország', 'százalék', 'istván', 'politikai', 'lászló', 'tulajdonos', 'állam', 'miniszter', 'fővárosi', 'forrás', 'társaság', 'február', 'polgármester', 'közbeszerzési', 'korrupció', 'minisztérium', 'péter', 'rendszer', 'szocialista', 'nyomozás', 'helyzet', 'terület', 'ingatlan', 'alapítvány', 'orbán', 'gábor', 'ország', 'párt', 'mszp', 'civil', 'információ', 'bank', 'beruházás', 'nyilvánosság', 'hónap', 'államtitkár', 'válasz', 'megyei', 'iroda', 'önkormányzati', 'fideszes', 'program', 'uniós']

['önkormányzat', 'korrupció', 'támogatás', 'pályázat', 'százalék', 'törvény', 'politikai', 'jános', 'bizottság', 'tulajdonos', 'forrás', 'hivatal', 'budapest', 'polgármester', 'állam', 'rendszer', 'gazdasági', 'lászló', 'február', 'miniszter', 'nyomozás', 'istván', 'társaság', 'magyar', 'nemzeti', 'alapítvány', 'terület', 'ingatlan', 'helyzet', 'péter', 'párt', 'ország', 'államtitkár', 'információ', 'nyilvánosság', 'beruházás', 'hónap', 'szóló', 'lehetőség', 'program', 'egyébként', 'projekt', 'érdek', 'feladat', 'dokumentum', 'lévő', 'civil', 'önkormányzati', 'fővárosi', 'válasz']

---

**Word saliency:**

['pályázat', 'minisztérium', 'támogatás', 'korrupció', 'fidesz', 'fővárosi', 'civil', 'társaság', 'ingatlan', 'vezérigazgató', 'törvény', 'alapítvány', 'szocialista', 'állam', 'terület', 'beruházás', 'bank', 'mszp', 'dokumentum', 'igazgató', 'szabó', 'nyilvánosság', 'megyei', 'iroda', 'család', 'tárca', 'előzetes', 'nyomozó', 'fideszes', 'tamás', 'kerület', 'érdek', 'önkormányzati', 'rendőrség', 'vádlott', 'rendszer', 'válasz', 'közbeszerzési', 'érték', 'államtitkár', 'gyanúsított', 'miklós', 'lehetőség', 'jogi', 'vállalkozás', 'bűncselekmény', 'projekt', 'érintett', 'párt', 'gyula']

['korrupció', 'támogatás', 'pályázat', 'törvény', 'civil', 'alapítvány', 'állam', 'társaság', 'gazdasági', 'vezérigazgató', 'ingatlan', 'terület', 'magyar', 'párt', 'nemzeti', 'nyilvánosság', 'hivatal', 'rendszer', 'beruházás', 'dokumentum', 'államtitkár', 'bank', 'válasz', 'fővárosi', 'lehetőség', 'pénzügyi', 'önkormányzati', 'érdek', 'tevékenység', 'program', 'vádlott', 'projekt', 'vállalkozás', 'választás', 'érintett', 'érték', 'minisztérium', 'család', 'közlemény', 'jelentés', 'fontos', 'tulajdon', 'eredmény', 'levél', 'tanácsadó', 'hatóság', 'javaslat', 'kampány', 'bűncselekmény', 'hirdetés']

Rövid vizsgálódásom alapján azt mondhatjuk, hogy ha nagyon kevés idő áll rendelkezésünkre, vagy valamilyen okból kifolyólag nincs lehetőségünk az alapos előfeldolgozásra, elkészíthetjük a modellt a szótövezett korpuszunkon is, azt szem előtt tartva, hogy a névelemek megkeresése már javítana az eredményeinken.



## IV. ÖSSZEGZÉS

Dolgozatomat öt célkitűzés mentén építettem fel:

1. Szerettem volna megmutatni, hogy hogyan jutunk el a nyers szövegtől a látens topikok felderítéséig, külön hangsúlyozva a korpusz előkészítésének menetét.
2. Ehhez kapcsolódó célkitűzésem volt a modell használhatóságának vizsgálata a három előfeldolgozási stádiumban.
3. Szerettem volna megmutatni, hogyan tudjuk a meglévő címkéinket egy nagy adathalmazra átvinni, és heurisztikát adni, az optimális topikszám megtalálására.
4. Különböző eszközökkel próbáltam megkönnyíteni a topikok értelmezését, és elnevezését.
5. Célom volt a K-Monitor önkénteseinek segítése, munkájuk standardizálása és gyorsítása.

Ennek megfelelően szakdolgozatom első felében bemutattam a topikmodellezés háttérét, definiáltam a főbb fogalmait és igyekeztem megmutatni, hogy hol foglalja el helyét a Machine Learning módszereken belül.

Részletesen bemutattam az Author-Topic Model módszertanát, és érintőlegesen a két másik modellt, melyre az ATM épül, az Author Modelt és az LDA-t.

A dolgozatom másik felében bemutattam a felhasznált eszközöket, az adatgyűjtés és adatelőkészítés módját. Az adatok bemutatását követően a korpusz és a modell elkészítésére tértem át. A szakdolgozatom fontos része a topikokkal foglalkozó fejezet, melyben célkitűzéseimnek megfelelően bemutattam a címkeajánlás módszertanát, az optimális topikszám megtalálásának egy lehetséges módját és a topikok elnevezésének folyamatát.

A predikció során két hasonlósági mutatót vizsgáltunk (Hellinger és Jensen), melyek közül a Jensen bizonyult jobbnak. A mutató a dokumentumok szóeloszlását hasonlítja össze a szerzőkhöz tartozó szóeloszlással. Azokat a címkéket tudjuk megmutatni az önkénteseknek, melyek 0-nál nagyobb hasonlósági mutatóval térnek vissza, ők pedig ezekből tudják kiválasztani azokat, melyeket a cikkekhez szeretnének társítani.

A címkék ekkor már nem tartalmazzák a tulajdonneveket, ezeket még az adatok

előkészítése során kiszűrtük, hiszen egy egyszerű NER eszközzel bármikor megtalálhatjuk a szövegekben a névelemeket. A címkeajánlást bővíteni lehetne egy kulcsszókinyerő alkalmazásával, ami az újonnan megjelenő témákra is felhívja az önkéntesek figyelmét.

Az optimális topikszám megtalálására a perplexitás mutató helyett a predikcióim pontosságát (*precision*) vizsgáltam. 49 modellt készítettem azonos paraméterbeállítások mellett, csupán a topikszám változtatásával. A használt modell jellegéből adódóan (mivel minden szerzőhöz összefűzi a hozzá tartozó dokumentumokat), a topikszám növelése mellett a duplikátum topikok megjelenésének valószínűsége is nő, a duplikátum topikok nehezítik az elemzést. Ezért az első duplikátumot tartalmazó modellig bezárólag kerestük a legjobb modellt. Az optimális topikszámot az a modell adja, ahol a precision mutató lokális maximuma található. Két vizualizációt készítettem a topikok könnyebb értelmezéséhez, a Bubble charttal a topikokhoz tartozó szavakat vizsgálhatjuk, ami a topikok elnevezését segíti, a gráf pedig a topikok kapcsolatainak vizsgálatában nyújt segítséget.

Dolgozatom célkitűzései szerteágazóak voltak, minden irányban továbblépési lehetőséggel. Sajnos a szakdolgozat szűk keretei miatt nem sikerült mélyen vizsgálnom a modell teljesítését az előfeldolgozás különböző szakaszaiban, de azt láthattuk, hogy nincsenek nagyon nagy eltérések a különböző stádiumok között. Azt mondhatjuk, hogy a névelemek kiszűrése javíthat az eredményeinken ahhoz képest, ha a korpuszon csak szótövezést végzünk, de a szignifikáns bigramok megkeresése már kevésbé látványos eredményeket mutatott az általam felhasznált szövegeken. Ennek ellenére még mindig azt gondolom, hogy ez egy fontos irány lehet a módszertani kutatásban.

A topikokat társadalomtudományi szempontból is vizsgálhatnánk. Így jobban megismerhetnénk a korrupció Magyarországon megjelenő formáit, összevethetnénk eredményeinket más kutatások eredményeivel és méginkább hangsúlyozhatnánk azt, hogy a topikmodellek mennyire hasznosak lehetnek a szociológiában, és új lehetőségeket nyitnak a tartalomelemzés módszertanában.

## V. Irodalomjegyzék

Balogh Kitti (2015). A látens Dirichlet allokáció társadalomtudományi alkalmazása. A kuruc.info romaellenes megnyilvánulásainak tematikus elemzése.

<https://labs.precognox.com/kuruc-info-visualization/>

A\_latens\_Dirichlet\_allokacio\_tarsadalomtudomanyi\_alkalmazasa\_Balogh\_Kitti.pdf

[Utolsó megtekintés: 2018. 04. 11.]

Basu, S. – Bilenko, M. – Mooney, R. J. (2004) A Probabilistic Framework for Semi-Supervised Clustering. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Seattle, WA.

<http://www.cs.utexas.edu/~ml/papers/semi-kdd-04.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Black, P. E. (2017). "big-O notation". *Dictionary of Algorithms and Data Structures*.

<https://www.nist.gov/dads/HTML/bigOnotation.html>

[Utolsó megtekintés: 2018. 04. 11.]

Blei, D. M. (2011). Probabilistic topic models. Conference paper in *Communications of the ACM*.

Blei, D. M. – Lafferty, J. D. (2007). A correlated topic model of science. *The Annals of Applied Statistics*. Vol.1, No 1, 17-35. Institute of Mathematical Statistics.

<http://www.cs.columbia.edu/~blei/papers/BleiLafferty2007.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Blei, D. M. – Lafferty, J. D.(2006) Dynamic Topic Models. *International Conference on Machine Learning, New York, NY, USA*. 113-120

<http://www.cs.columbia.edu/~blei/papers/BleiLafferty2006a.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Blei, D. M. – Lafferty, J. D.(2009). Topic Models. In A. N. Srivastava and M. Sahami, editors, *Text mining: Classification, Clustering, and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series.

<http://www.cs.columbia.edu/~blei/papers/BleiLafferty2009.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Blei, D. M. – McAuliffe, J. D. (2010). Supervised topic models. *Advances in neural information processing systems*, 121-128

<http://papers.nips.cc/paper/3328-supervised-topic-models.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Blei, D. M. – Ng, A.Y. – Jordan, M.I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993-1022

Blei, D. M.– Kucukelbir, A. – McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, Vol. 112 , Iss. 518  
<https://arxiv.org/pdf/1601.00670.pdf>  
[Utolsó megtekintés: 2018. 04. 11.]

Byord-Graber, J. – Blei, D. (2009). Syntactic Topic Models. *Advances in neural information processing systems*, 185-192.  
<http://www.cs.columbia.edu/~blei/papers/Boyd-GraberBlei2009.pdf>  
[Utolsó megtekintés: 2018. 04. 11.]

Chang, J. – Gerrish, S. – Wang, Ch. – Blei, D. M. (2009). Reading Tea Leaves. How Humans Interpret Topic Models. *Advances in neural information processing systems*, 288-296.

Chuang, J. – Manning, Ch. D. – Heer, J. (2012). Termite. Visualization Techniques for Assessing Textual Topic Models. Proceedings of the international working conference on advanced visual interfaces. 74-77  
<http://vis.stanford.edu/files/2012-Termite-AVI.pdf>  
[Utolsó megtekintés: 2018. 04. 11.]

Deerwester, S. – Dumais, S.T. – Furnas, G.W. – Landauer, T.K. – Harshman, R.: (1990). Indexing by latent semantic analysis. In: *Journal of the Association for Information Science and Technology*.

Gelman, A – Carlin, J. B. – Stern, H. S. – Dunson, D. B. – Vehtari, A. – Rubin. D. B. (2014). Bayesian Data Analysis. Chapman & Hall/CRC Press.

Gilks, W. – Richardson, S. – Spiegelhalter, D. (1996). Markov Chain Monte Carlo in Practice. Chapman & Hall, New York, NY.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2), 177-196.  
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.6341&rep=rep1&type=pdf>  
[Utolsó megtekintés: 2018. 04. 11.]

Jurafsky, D. – Martin J. H. (2008). Speech and Language Processing. Prentice Hall.  
<https://nlp.stanford.edu/~manning/xyzy/JurafskyMartinEd2book.pdf>  
[Utolsó megtekintés: 2018. 04. 11.]

Kao, A. – Poteet, S. R. (2007). Natural Language Processing and Text Mining. Springer-Verlag London.

[http://129.219.222.66/publish/pdf/natural\\_language\\_processing\\_and\\_text\\_mining.pdf](http://129.219.222.66/publish/pdf/natural_language_processing_and_text_mining.pdf)

[Utolsó megtekintés: 2018. 04. 11.]

Liddy, E. D. (2001). Natural Language Processing. Natural Language Processing. In *Encyclopedia of Library and Information Science*, 2nd Ed. NY. Marcel Decker, Inc

<https://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub>

[Utolsó megtekintés: 2018. 04. 11.]

Mendes, P. N.– Jakob, M. – García-Silva, A. – Bizer, Ch. (2011). DBpedia Spotlight: Shedding Light on the Web of Documents. I-SEMANTICS 2011, 7th Int. Conf. on Semantic Systems.

<http://www.dbpedia-spotlight.org/docs/spotlight.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Mitchell, T. M.: (1997). Machine Learning. McGraw-Hill Science/Engineering/Math

[https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-](https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-%20Machine%20Learning%20-Tom%20Mitchell.pdf)

[%20Machine%20Learning%20-Tom%20Mitchell.pdf](https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-%20Machine%20Learning%20-Tom%20Mitchell.pdf)

[Utolsó megtekintés: 2018. 04. 11.]

Mohri, M. – Rostamizadeh, A. –Talwalkar, A. (2012). Foundations of Machine Learning. MIT Press. England.

Mortensen, O. (2017) The Author-Topic Model.

[http://www2.imm.dtu.dk/pubdb/views/edoc\\_download.php/6971/pdf/imm6971.pdf](http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6971/pdf/imm6971.pdf)

[Utolsó megtekintés: 2018. 04. 11.]

Newman, D. – Lau, J. H. – Grieser, K. –Baldwin, T. (2010) Human Language Technologies. *The 2010 Annual Conference of the North American Chapter of the ACL*, pages 100–108, Los Angeles, California.

<http://www.aclweb.org/anthology/N10-1012>

[Utolsó megtekintés: 2018. 04. 11.]

Rehurek, R. (2011). Scalability of Semantic Analysis in Natural Language Processing.

[https://radimrehurek.com/phd\\_rehurek.pdf](https://radimrehurek.com/phd_rehurek.pdf)

[Utolsó megtekintés: 2018. 04. 11.]

Reményi Andrea (2010). Kollokációk korpuszalapú vizsgálata. In: *Fordítástudomány XII.* 2. szám 67–95.

Robert, C. P. – Casella, G: Monte Carlo Statistical Methods. (2004). Springer Verlag.  
[https://www.researchgate.net/profile/Christian\\_Robert2/publication/41222435\\_Monte\\_Carlo\\_Statistical\\_Method/links/0c9605323701881344000000/Monte-Carlo-Statistical-Method.pdf](https://www.researchgate.net/profile/Christian_Robert2/publication/41222435_Monte_Carlo_Statistical_Method/links/0c9605323701881344000000/Monte-Carlo-Statistical-Method.pdf)

[Utolsó megtekintés: 2018. 04. 11.]

Rosen-Zvi, M. – Griffiths, T. – Steyvers – Smyth, P. (2004a). The Author-Topic-Model.  
<https://www.slideshare.net/FREEZ7/author-topic-model>

[Utolsó megtekintés: 2018. 04. 11.]

Rosen-Zvi, M. – Griffiths, T. – Steyvers – Smyth, P. (2004b). The Author-Topic Model for Authors and Documents. In *AUAI'04: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* 487–494. AUAI Press, Arlington, VA.  
<https://mimno.infosci.cornell.edu/info6150/readings/398.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Rosen-Zvi, M. – Chemudugunta –Griffith, T. – Smyth, P. – Steyvers (in press). (2008). Learning Author-Topic Models from Text Corpora. *ACM Transactions on Information Systems*, Vol. V, No. N, October 2008, Pages 1–38.  
[http://psiexp.ss.uci.edu/research/papers/AT\\_tois.pdf](http://psiexp.ss.uci.edu/research/papers/AT_tois.pdf)

[Utolsó megtekintés: 2018. 04. 11.]

Rosen-Zvi, M. – Griffiths, T. – Smyth, P. (2012). The Author-Topic Model for Authors and Documents. *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. 487-494  
<https://mimno.infosci.cornell.edu/info6150/readings/398.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Roberts M. E. – Stewart B. M. – Tingley D. – Airoldi E. M. (2013). The Structural Topic Model and Applied Social Science. *Advances in Neural Information Processing Systems Workshop on Topic Models: Computation, Application, and Evaluation*  
<https://scholar.princeton.edu/files/bstewart/files/stmnips2013.pdf>

[Utolsó megtekintés: 2018. 04. 11.]

Seroussi, Y. – Zukerman, I. – Bohnert, F. (2011). Authorship Attribution with Latent Dirichlet Allocation. *Association for Computational Linguistics*  
<https://pdfs.semanticscholar.org/26a2/41fea8e193afc8f87713d493144129bf3ae2.pdf>

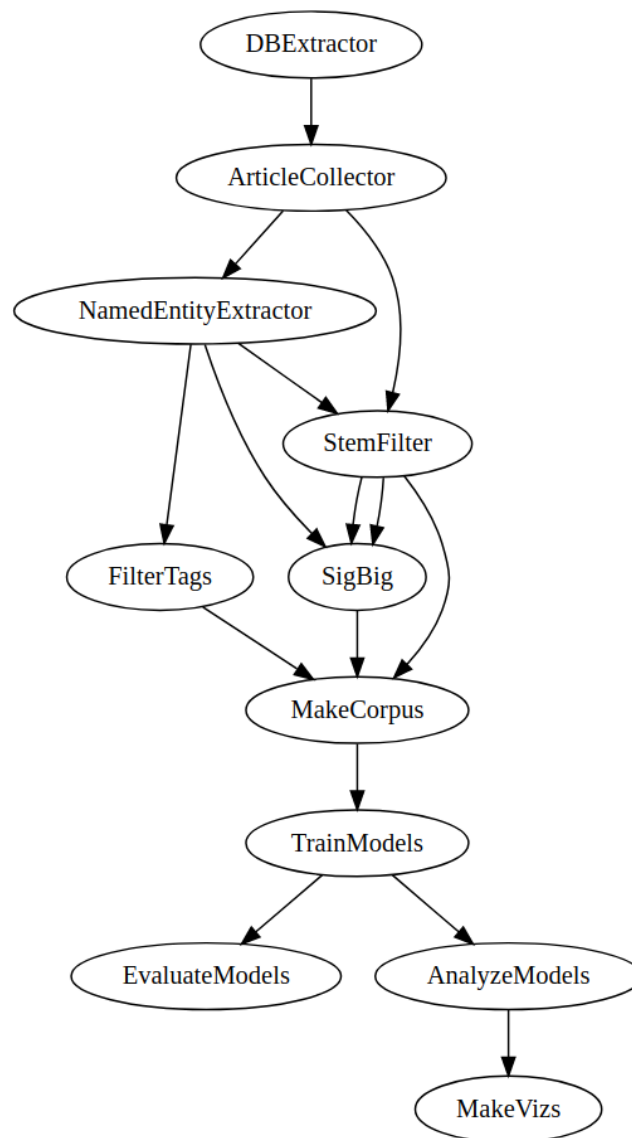
[Utolsó megtekintés: 2018. 04. 11.]

Shalev-Shwartz, S. – Ben-David, S. (2014). Understanding Machine Learning. Cambridge University Press.  
<http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf>  
[Utolsó megtekintés: 2018. 04. 11.]

Tikk, D. – Farkas, R. – Kardkovács, Z. T. – Kovács, L. – Répási T. – Szarvas G. – Szaszko S. – Vázsonyi M.: *Szövegbányászat*. Typotex, Budapest, 2007

## VI. Melléklet

M1. Ábra: A Luigi dependency gráfja





## M2. ábra: Directory structure a Cookiecutter Data Science template-ből

```

├── LICENSE
├── Makefile          <- Makefile with commands like `make data` or `make train`
├── README.md         <- The top-level README for developers using this project.
├── data
│   ├── external      <- Data from third party sources.
│   ├── interim       <- Intermediate data that has been transformed.
│   ├── processed     <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
├── docs              <- A default Sphinx project; see sphinx-doc.org for details
├── models            <- Trained and serialized models, model predictions, or model summaries
├── notebooks         <- Jupyter notebooks. Naming convention is a number (for ordering),
│                       the creator's initials, and a short '-' delimited description, e.g.
│                       `1.0-jqp-initial-data-exploration`.
├── references        <- Data dictionaries, manuals, and all other explanatory materials.
├── reports           <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures       <- Generated graphics and figures to be used in reporting
├── requirements.txt  <- The requirements file for reproducing the analysis environment, e.g.
│                       generated with `pip freeze > requirements.txt`
├── src               <- Source code for use in this project.
│   ├── __init__.py   <- Makes src a Python module
│   ├── data          <- Scripts to download or generate data
│   │   └── make_dataset.py
│   ├── features      <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   ├── models        <- Scripts to train models and then use trained models to make
│   │                   predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   └── visualization <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
└── tox.ini           <- tox file with settings for running tox; see tox.testrun.org

```



M3. ábra: A felhasznált adatbázis struktúrája

<b>news_news</b>	
<b>news_id</b>	INTEGER
status	CHAR
rights	CHAR
source	VARCHAR
source_url	VARCHAR
<b>source_url_ok</b>	CHAR
source_url_check	INTEGER
source_url_string	VARCHAR
cre_time	INTEGER
mod_time	INTEGER
pub_time	INTEGER
cre_id	INTEGER
mod_id	INTEGER
pub_id	INTEGER
views	INTEGER
<b>news_type</b>	VARCHAR
<b>frontpage</b>	VARCHAR
<b>sticky</b>	VARCHAR
<b>reported</b>	CHAR
<b>starred</b>	VARCHAR
<b>halora</b>	CHAR
<b>comments</b>	INTEGER
<b>comments_allow</b>	CHAR
size_type	CHAR
<b>stat_point</b>	FLOAT
<b>news_rel</b>	CHAR
comment	TEXT
poll_id	INTEGER
import_id	INTEGER
inner_id	VARCHAR

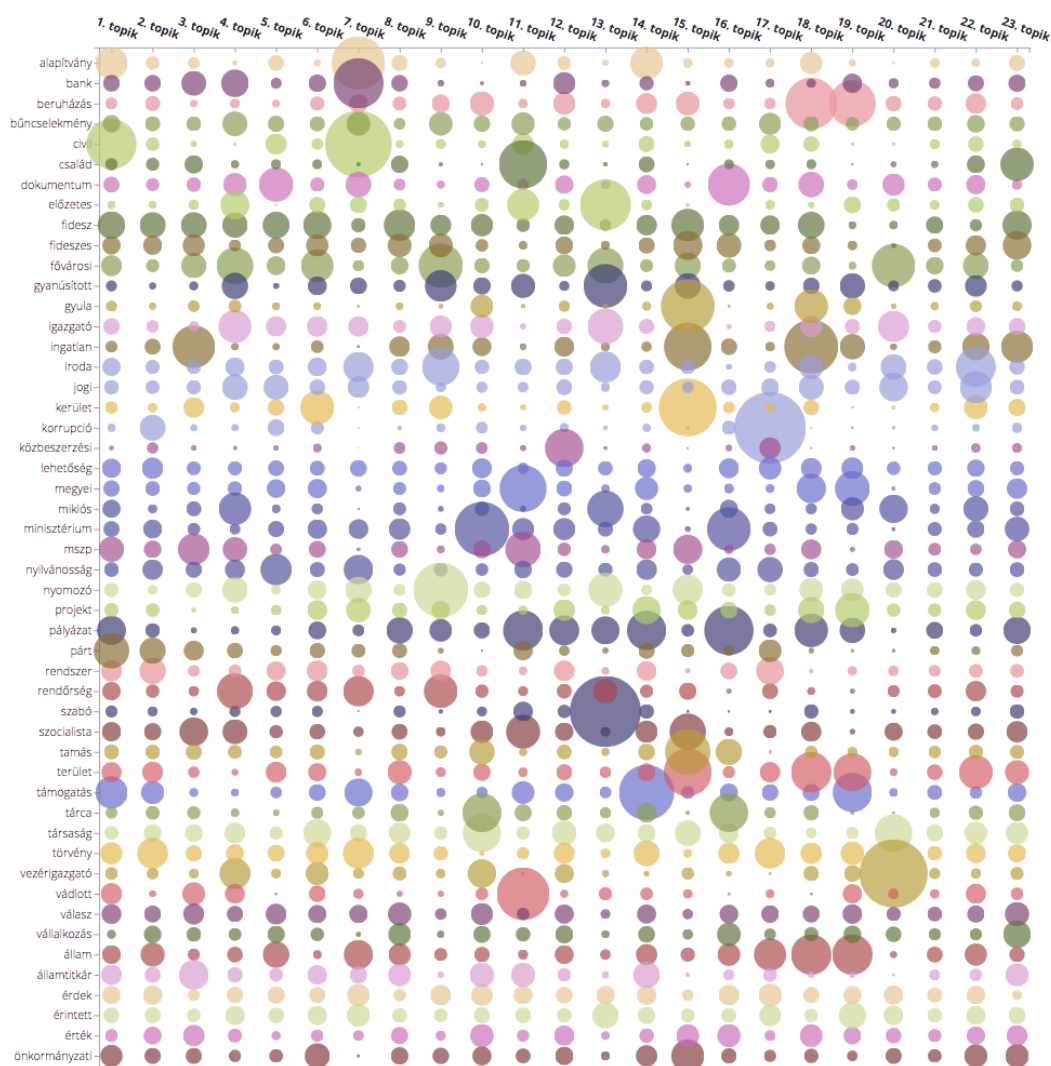
<b>news_newspapers</b>	
<b>newspaper_id</b>	INTEGER
status	CHAR
name	VARCHAR
teaser	TEXT
bodytext	TEXT
cre_time	INTEGER
mod_time	INTEGER
cre_id	INTEGER
mod_id	INTEGER
<b>import_id</b>	INTEGER

<b>news_newspapers_link</b>	
<b>news_id</b>	INTEGER
<b>newspaper_id</b>	INTEGER

<b>news_tags</b>	
<b>tag_id</b>	INTEGER
news_id	INTEGER
names	VARCHAR

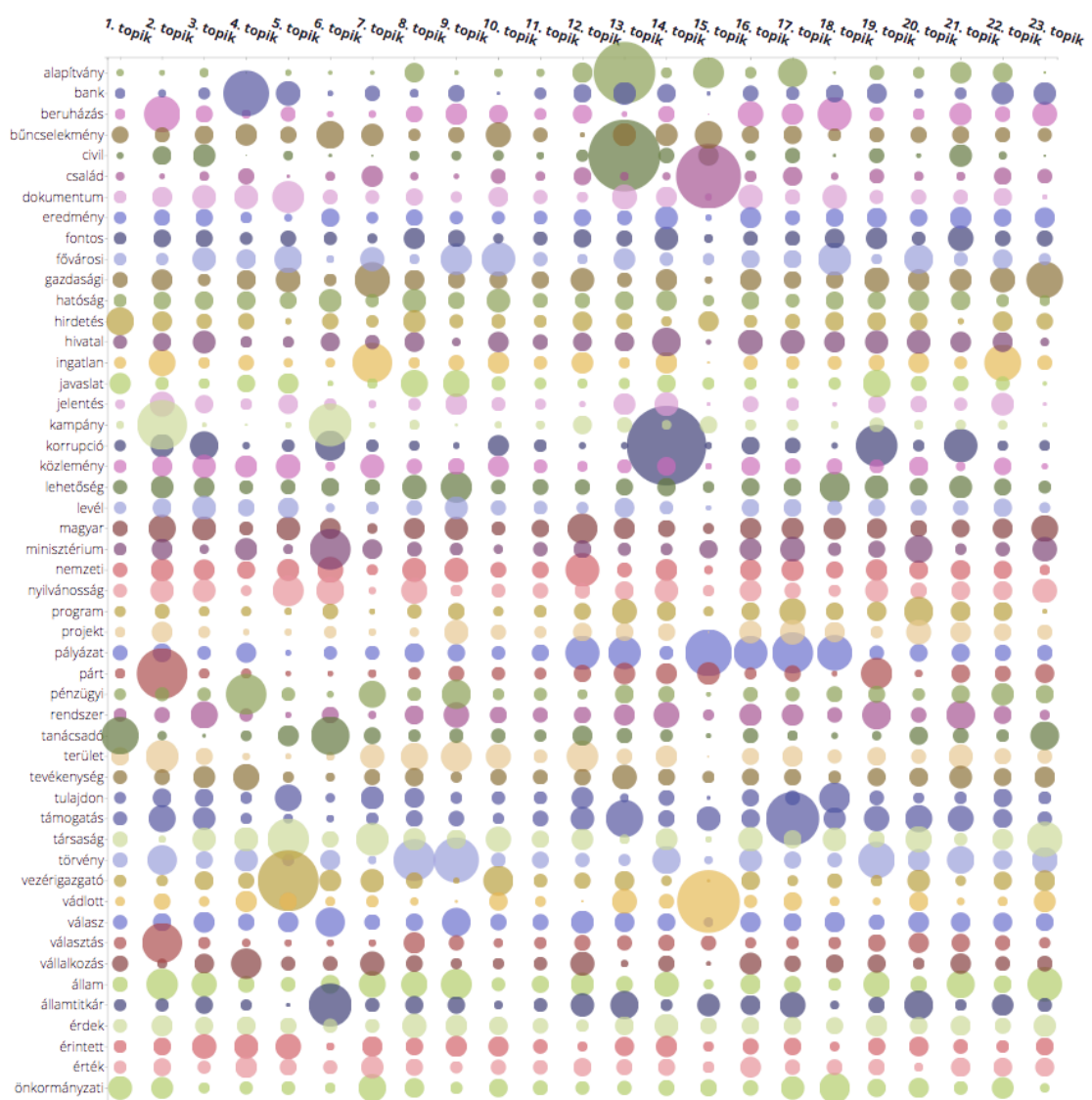
<b>FullKmonitor</b>	
 <b>news_id</b>	INTEGER
 <b>newspaper_id</b>	INTEGER
newspaper	VARCHAR
source_url	VARCHAR
tag	VARCHAR

M4. ábra: Term saliency a szótövezett korpuszon



Az y tengelyen látható kifejezések: ['pályázat', 'minisztérium', 'támogatás', 'korruptió', 'fidesz', 'fővárosi', 'civil', 'társaság', 'ingatlan', 'vezérigazgató', 'törvény', 'alapítvány', 'szocialista', 'állam', 'terület', 'beruházás', 'bank', 'mszp', 'dokumentum', 'igazgató', 'szabó', 'nyilvánosság', 'megyei', 'iroda', 'család', 'tárca', 'előzetes', 'nyomozó', 'fideszes', 'tamás', 'kerület', 'érdek', 'önkormányzati', 'rendőrség', 'vádlott', 'rendszer', 'válasz', 'közbeszerzési', 'érték', 'államtitkár', 'gyanúsított', 'miklós', 'lehetőség', 'jogi', 'vállalkozás', 'bűncselekmény', 'projekt', 'érintett', 'párt', 'gyula']

M5. ábra: Term saliency a szótövezett és a kinyert névelemeket tartalmazó korpuszon



Az y tengelyen látható kifejezések: ['korruptio', 'tamogatás', 'palyázat', 'torvény', 'civil', 'alapitvány', 'állam', 'társaság', 'gazdasági', 'vezérigazgató', 'ingatlan', 'terület', 'magyar', 'párt', 'nemzeti', 'nyilvánosság', 'hivatal', 'rendszer', 'beruházás', 'dokumentum', 'államtitkár', 'bank', 'válasz', 'fővárosi', 'lehetőség', 'pénzügyi', 'önkormányzati', 'érdek', 'tevékenység', 'program', 'vádolt', 'projekt', 'vállalkozás', 'választás', 'érintett', 'érték', 'minisztérium', 'család', 'közlemény', 'jelentés', 'fontos', 'tulajdon', 'eredmény', 'levél', 'tanácsadó', 'hatóság', 'javaslat', 'kampány', 'bűncselekmény', 'hirdetés']

### A felhasznált jelölések feloldása:

A jelöléseket Rosen-Zvi – Chemudugunta – Griffith – Smyth – Steyvers (2008) és Rehurek (2011) tanulmányaira alapoztam, ezeket hoztam egységes alakra.

$A$	Szerzők a korpuszban (set)
$\mathbf{a}_d$	d dokumentum szerzői
$A_d$	d dokumentum szerzőinek száma
$C^{TA}$	Adott topikhoz és szerzőhöz hozzárendelt szavak száma
$C^{WT}$	Adott topikhoz és szóhoz hozzárendelt szavak száma
$D^{train}$	Set a szerzőkből és szavakból a tanuló adathalmazon
$A$	Szerzők száma
$a$	Egy adott szerző
$D$	Dokumentumok száma
$d$	Egy adott dokumentum
$N_d$	Szavak száma d dokumentumban
$N$	Szavak száma adott korpuszban
$T$	Topikok száma
$t$	Egy adott topik
$V$	Szavak száma adott szótárban
$V_d$	Szavak száma d dokumentumban
$v$	Adott szótár egyetlen szava
$\mathbf{w}_d$	d dokumentum szavai
$\mathbf{w}$	Adott korpusz szavai
$w_{di}$	d dokumentum i-edik szava
$\mathbf{x}$	szerzőhozzárendelés
$x_{di}$	$w_{di}$ szó szerzőhozzárendelése
$\mathbf{z}$	topikhozzárendelés
$z_{di}$	$w_{di}$ szó topikhozzárendelése
$\alpha$	Dirichlet prior
$\beta$	Dirichlet prior
$\Phi$	Topikok valószínűsége adott szavak esetében
$\Theta$	Topikok valószínűsége adott szerzők esetében
$Dir$	Dirichlet eloszlás
$Unif$	Egyenletes eloszlás
$Mult$	Polinomiális eloszlás