Analysis of discussions on Reddit about the Hungarian primary elections in 2021

Topic detection in social media
Data Science Laboratory

Bálint Hantos

supervisor: István Márkusz
Eötvös Loránd University

November 25 2021

Contents

Introduction

In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Topic modeling is a frequently used text-mining tool for discovery of hidden semantic structures in a corpus.

A topic model captures this intuition in a mathematical framework. It allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is. Topic detection methods are ways to represent a corpus in a lower dimension.

Chapter 1

Data sources

1.1 First steps

The original plan was to gather historical data from Twitter from a given time interval mentioning a certain word. I had to register a developer account at Twitter and authorize my profile via phone. Then created a project and waited for approval from Twitter devs. At the same time I installed tweepy, which I used to make queries to the Twitter API using python.

Later on I had to face the fact, that simply I'm unable to gather tweets from the past in any meaningful quantity and quality. The standard plan is restricted to only 3000 tweets/month with a limitation of 128 characters per tweet [1]. We agreed with my supervisor that it was not enough to tackle with the task at hand, as a substantial amount of this 3000 tweets are retweets.

1.2 Change of subject

So, we had to come up with a new idea for the subject of the project on a different platform. For this other platform we choose Reddit, that has a much more communal approach to social media and follows a structure somewhat different from Facebook, Instagram, Twitter, LinkedIn etc. The most important thing is that users are anonymous. The other thing that sets it apart from other sites is that content of a forum is not determined by an algorithm, but it is governed by a voting system, so that the users in a community are to decide which posts are going to be on the front page. [2]

We decided to choose the primary elections in Hungary and the surrounding discussion about it on Reddit. On reddit the largest Hungarian speaking subreddit is r/hungary, which has approximately 150.000 members - although only a fraction of it write posts or comments actively. The gathering period started from Sept 1 2021 and ended at Oct 13 2021.

1.3 Data acquisition

In order to do that I needed to gather data from the r/hungary subreddit with Politics flair. I needed to use two APIs: PSAW to gather post IDs from the given interval, and PRAW to gather the comments from the posts signified with the post IDs.

As in the case of Twitter Reddit also has an official API, which can be accessed through Python with PRAW (Python Reddit API Wrapper). PRAW also needs some authentication on the part of Reddit, so I needed to create an app at the Reddit developer page and generate a token.

I tried to write queries using this PRAW package, which became quite easy to use after a while. But unfortunately, I came to the conclusion that using this package I can only go back as far as 1000 posts (which is the average number of posts in a week on r/hungary). Moreover, time horizon for the posts couldn't be specified neither. But other than that, all the comments for a post were downloaded

So, instead I looked for another method to go even further into the post and comment history and found PSAW (PushShift.io API Wrapper). PSAW comes handy when one wants to gather a massive amount of data from Reddit, as it is much more powerful in some dimensions than PRAW. But it had Using PSAW I could finally gather posts from even the beginning of September 2021, and then I had to make queries for the comments using the post IDs.

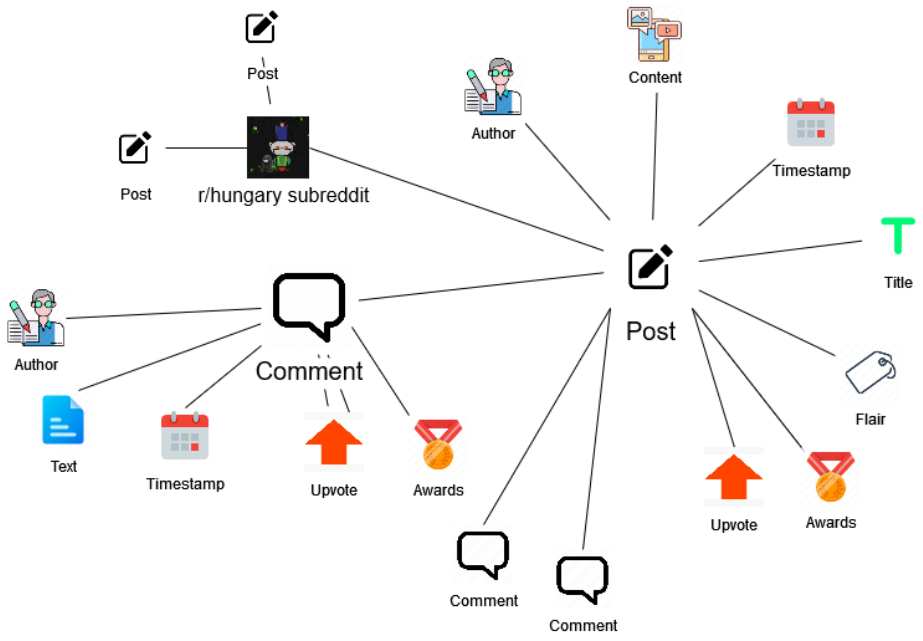

Figure 1.1: Structure of posts and comments on the reddit social media site. Image by author, icons are from various webpages[3].

Chapter 2

Data transformations

There were a couple of steps that needed to be made before the topic detection methods could be applied. The data needed to be cleaned from a technical perspective. Then, I processed the text from a linguistic approach. After that, encoded the words of each document in a term-frequency matrix, and refined it by weighing the words by the inverse document frequency.

2.1 Data cleaning

Firstly, I have removed web links, user mentions, emojis (although they are used quite infrequently on purpose on reddit), punctuations, numbers, newline characters, redundant whitespaces and other string escape characters. I left in subreddit mentions (go by *r/some-other-subreddit*) because they are in a close relationship with the comment's content, just like a hashtag. After having done all that, I made all the words lowercase.

Secondly, processed the text from a linguistic approach: removed stopwords and utilized stemming and lemmatizing separately. The differences can be understood using an example given on Fig (4). Stemming and lemmatizing are different in the sense that stemming chops down some characters from the end of a word, which may result in a loss of meaning. On the other hand, lemmatizing is a bit more refined, as taking into account the morphology of the word, thus the output is closer to the root word. [4] The downside of lemmatizing is that there are only a couple of natural languages for which there is a sophisticated lemmatizer library - and Hungarian is not such a language.

For stemming I used the built-in tools of scikit-learn [5] and for lemmatizing I used simplemma [6] I later compared the two approach because it lead to somewhat different results.

Form	Morphological information	Lemma
studies	Third person, singular number, present tense of the verb study	study
studying	Gerund of the verb study	study
niñas	Feminine gender, plural number of the noun niño	niño
niñez	Singular number of the noun niñez	niñez

Form	Suffix	Stem
studies	-es	studi
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

Figure 2.1: Comparing stemming and lemmatizing.
While stemming chops off the end of the word based
on a list of suffices, lemmatizing considers
morphological information to find the root of the word.
Image credits to [4].

2.2 Encoding

The third step was to encode the features. I used the term-frequency encoding to encode the text. Later I transformed this matrix using the Term Frequency – Inverse Document Frequency (TF-IDF) method. The TF-IDF matrix is a more sophisticated representation of the text when doing topic modeling. This is because the IDF gives more weight to less frequent words and decreases the weight of more common words, thus the $w_{i,j}$ TF-IDF matrix looks as follows:

$$w_{i,j} = t_{i,j} \ln \frac{N}{f_i},$$

where $t_{i,j}$ is the count of word i in document j, f_i is the count of word i in all documents and N is the number of documents.

	brown	dog	fox	lazy	quick	red	slow	the	yellow
“the quick brown fox”	1	0	1	0	1	0	0	1	0
“the slow brown dog”	1	1	0	0	0	0	1	1	0
“the quick red fox”	0	1	0	0	1	1	0	1	0
“the lazy yellow fox”	0	0	1	1	0	0	0	1	1

Figure 2.2: Example of a term frequency matrix of short documents.
The rows are for documents, the columns are for the words in the corpus.
Each cell contains the count of the given word in the document.
Image credits to [7].

Comment length group	Vocabulary size	Number of documents	Number of words
7-10	9118	2572	21220
10-20	14763	3106	42774
20-40	14424	1573	42943
40-60	8625	429	20574
60-100	5278	155	10731
15-40	18363	2594	60079
40-100	11425	572	30585

Table 2.1: Partitioning of the corpus. We found that it was more meaningful to separate comments based on their word length. More topics are likely to be attributed to a longer comment, so they are not only more complex but different.

Chapter 3

Methods

Until this point the cleaning and encoding of the text was discussed, which are necessary preprocessing steps in order to utilize topic detection algorithms.

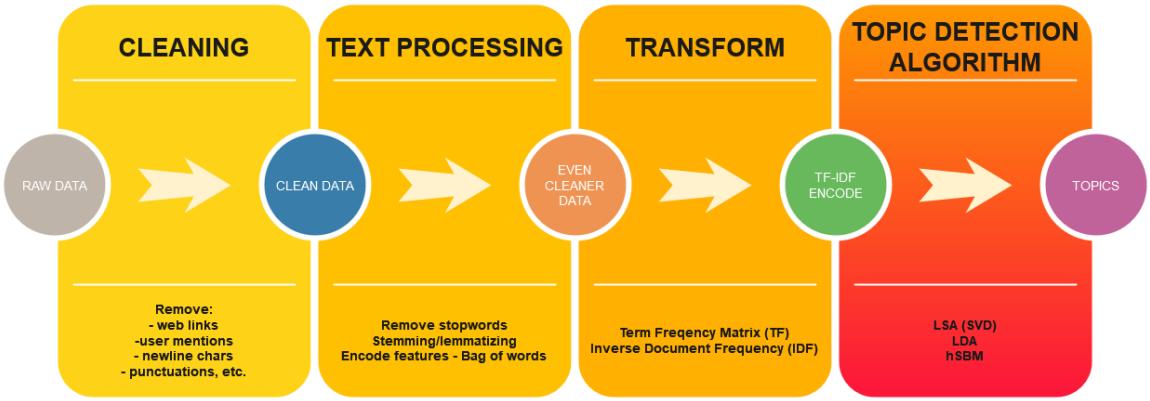

Figure 3.1: Topic detection pipeline. Image by author.

Topic detection is based on the idea that words used in the same context tend to have similar meaning. So, documents with similar topics will use similar words. We consider this phenomena as a *latent class model*, see Figure 3.2. In the following sections I am going to give an introduction on the methods I have been using for this task.

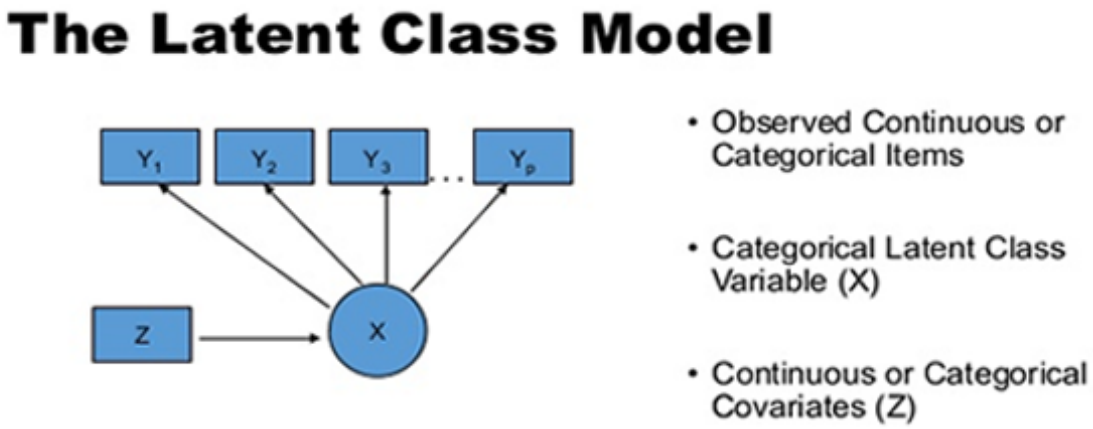

Figure 3.2: Latent class model. Try to reconstruct documents (Z) word by word (Y), which words are connected to the topic(s)(X). In the case of topic detection there are usually multiple topics. Reconstruction: pick a topic from a document and choose words according to that topic. Image credits to [8].

3.1 Latent semantic indexing (LSI)

The LSI algorithm uses a technique called *singular value decomposition* to identify relationship between words and documents in an unstructured text. The documents are represented in the TF-IDF matrix as discussed in the last chapter. The TF-IDF matrix is a sparse matrix, which means, most of the matrix is filled with zeros.

A good idea is to try to reconstruct matrix in lower dimensions, thus eliminating the large number of zeros. This way we not only reduce the size of the matrix but we can get to a close low dimensional representation of it. It is essentially the same as calculating the principal components (PCA), however the this time the mean is not subtracted, thus the matrix retains its sparseness. [9]

Singular value decomposition (**SVD**) is a generalization of eigendecomposition for non-square matrices. Given $\mathbf{A}$ square matrix,

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$$

is its eigenvalue decomposition, where Q is the eigenvectors in a columnar form, and $\mathbf{\Lambda}$ contains the eigenvalues in the diagonal.

Similarly, $\mathbf{M}$ (m $\times$ n) a non-square matrix, is decomposed with SVD as:

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

visually

$$\mathbf{M} = \begin{pmatrix} \vec{u_1} & \vec{u_2} & ... & \vec{u_m} \end{pmatrix} \left(\begin{array}{cccc} \sigma_1 & 0 & & \\ 0 & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \\ \hline 0 & 0 & 0 & 0 \\ & \cdots & & \\ 0 & 0 & 0 & 0 \end{array}\right) \begin{pmatrix} \vec{v_1} & \vec{v_2} & ... & \vec{v_n} \end{pmatrix}^T,$$

where the $\vec{u_i}$ are the singular vectors and σ_i are the singular values.

SVD is a very useful tool in dimension reduction. In order to reconstruct $\mathbf{M}$ economically take just a few singular vectors based on their singular values.

These singular vectors are good descriptors of the corpus, they make up some sort of underlying structure for the text. We might think about these singular vectors as the hidden features of the observable text.

The meaning of each matrix in the LSI regime is the following:

- $\mathbf{U}$ contains the relationship between the topics and the documents.
- $\mathbf{\Sigma}$ contains the information on the most important topics.
- $\mathbf{V}^T$ contains the relationship between the topics and the words.

In my work I used the scikit-learn [5] implementation of the SVD algorithm.

3.2 Latent Dirichlet allocation (LDA)[11]

LDA is one of the most popular topic modeling methods. It goes one step further than LSI: the similar approach is followed as we are looking for a hidden features of the observable text, but instead of using linear algebra, a probabilistic approach is followed.

Documents are probability distributions over latent topics. Topics are probability distributions over words. LDA takes a number of documents. It assumes that the words in each document are related. It then tries to figure out the "recipe" for how each document could have been created. We just need to tell the model how many topics to construct and it uses that recipe to generate topic and word distributions over a corpus. Based on that output we can identify similar documents within the corpus.

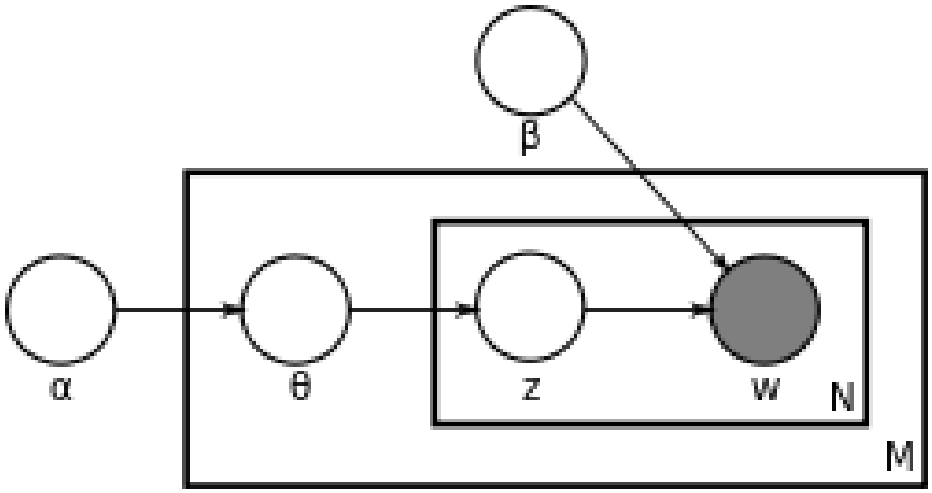

Figure 3.3: Plate notation is a concise way of visually representing the dependencies among the model parameters. Image credits to [10].

The parameters outside of the rectangle are the Dirichlet-priors α and β. These parameters are the model parameters and do not relate to the corpus specifically. Parameters of the plate diagram [10]:

- M denotes the number of documents.
- N is number of words in a given document (document i has N_i words).
- α is the parameter of the Dirichlet prior on the per-document topic distributions.
  A large α indicates that each document is likely to contain a mixture of most of the topics, not just 1 or 2 in particular. Conversely a small α indicates that each topic contains just a few of the topics.
- β is the parameter of the Dirichlet prior on the per-topic word distribution.
  A high β indicates that each topic will likely contain a mixture of most of the words, while a low beta means that each topic will likely contain just a few words.

- $\theta_i[\alpha]$ is the topic distribution for document i.
- $\phi_k[\beta]$ is the word distribution for topic k.
- z_{ij} is the topic for the j-th word in document i.
- w_{ij} is the specific word.

The generative process of LDA assumes that new documents are created in the following way:

1. Determine the number of N_i words in a document.
2. Choose a topic mixture for the i document over a fixed set of k topics.
3. Generate the words in the document by picking a topic based on θ_i and then picking a word based on ϕ_k.

When we try to detect topics in a corpus we fit the parameters so that LDA backtracks what k topics are likely to have generated the corpus.

1. Randomly assign each word in each document to one of the k topics.
2. For every document d:
    - Assume all topic assignments are correct except for the current one.
    - Calculate the proportion of words in document d that are assigned currently to the topic t: $\theta_i[\alpha] = P(t|d)$.
    - Calculate the proportion of word w assignments to topic t over all documents: $\phi_k[\beta] = P(w|t)$.
    - Multiply these together and sum for all topics to see how well the document is reconstructed: $P(w|d) \approx \sum_t P(w|t) \cdot P(t|d)$.
3. Note that we know this $P(w|d)$ probability. It is the frequency of the w word occurring in document d.

For the LDA algorithm to work I basically switched out the SVD step in my pipeline with the LDA implemented in [5].

3.3 Hierarchical stochastic block model

The hierarchical stochastic block model (hSBM) is an extension of the SBM.
The SBM is a network model which is used to generate random networks with some clusters. [12] The clustering means that edges are more common within communities than between them. These communities are disjoint. It is parameterized by SBM(n, $\mathbf{p}$, $\mathbf{W}$):

- n - number of nodes in the network
- $\mathbf{p}$ - $(p_1, ..., p_k)$ probability vector of one node
  being connected to cluster $i \epsilon k$, k is the number of clusters, $|\mathbf{p}| = 1$.
- $\mathbf{W}$ - $[0, 1]^{k \times k}$ is a symmetric matrix of probabilities being an edge between two nodes belonging to two - not necessarily different - communities.
  So $W_{i,j}$ is the probability of one node from community i and another node from community j being connected.

One of the most important applications of the SBM (or hSBM) model is community detection, meaning to find out if the network has a latent community structure and recover these communities.

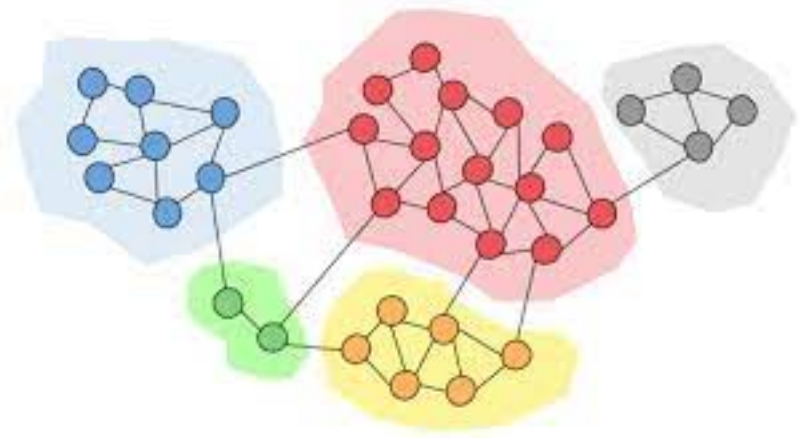

Figure 3.4: Communities in a network. Image credits to [13].

This is where topic detection comes into play.

Consider a bipartite network, a network, where nodes can be divided into two disjoint sets and there are no edges between the nodes in the same set. On one side of the network there are the documents and on the other the words from those documents. Put an edge between two nodes on the two sides if a document contains a word, see Fig 3.5.

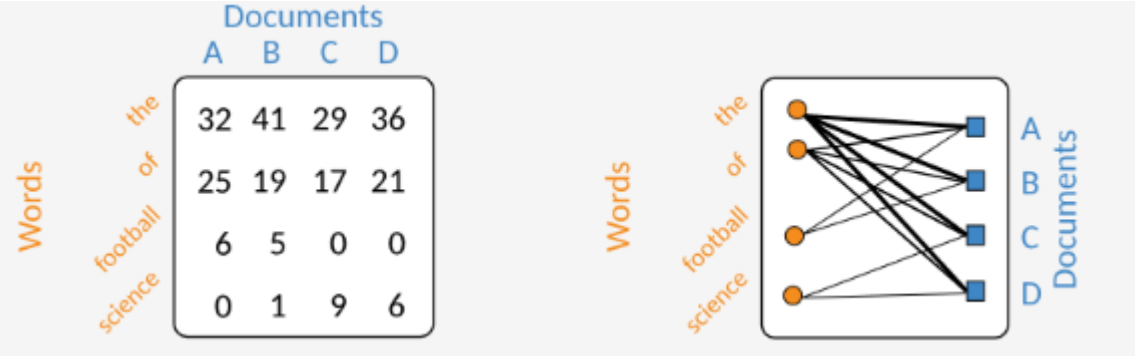

Figure 3.5: Term frequency matrix on the left and its representation as a bipartite graph on the right. Image credits to [14].

The aim of using the SBM is to identify communities in the word-document bipartite graph.

The hierarchical extension of the SBM means that there may be smaller communities inside the large ones and some even smaller communities in the small ones.

For the hSBM algorithm I used the [15] code written by the authors of [14]. Its main dependency is the graph-tool library, but it can only be installed on Linux and Mac natively. As I use Windows I needed to make following steps. In order to use that piece of code I needed to download Docker, pull the graph-tool library's image [16], clone [15] library into the Docker container, and mount my local files.

Chapter 4

Results

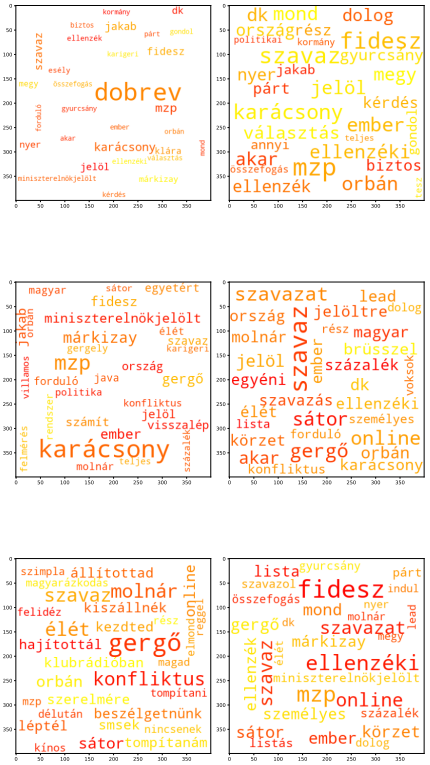

Figure 4.1: LSI: 40-100 words/comment, 6 components, lemmatized words

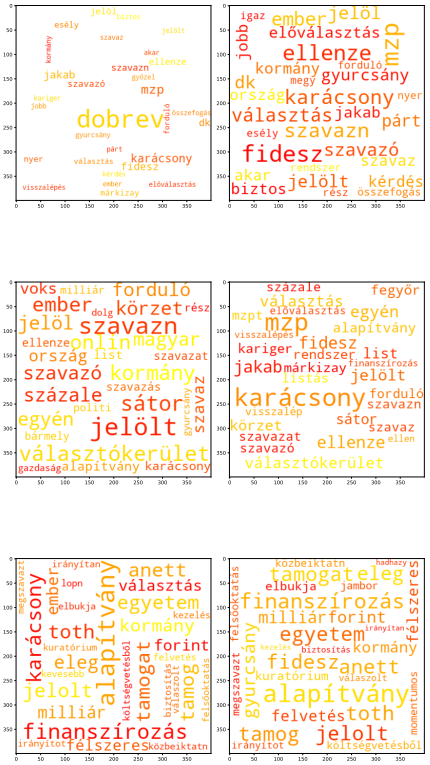

Figure 4.2: LSI: 40-100 words/comment, 6 components, stemmed words

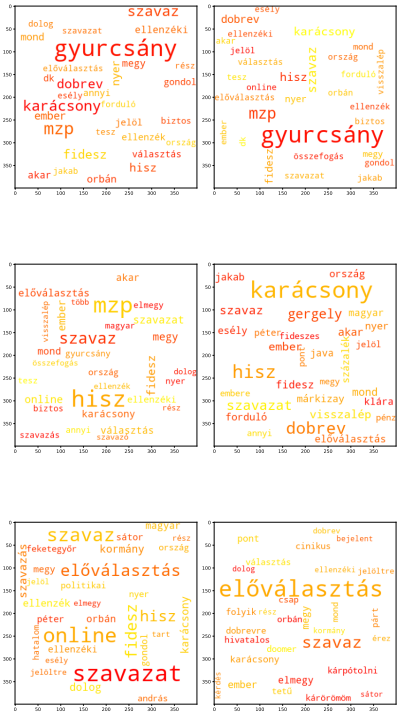

Figure 4.3: LSI: 15-40 words/comment, 6 components, lemmatized words

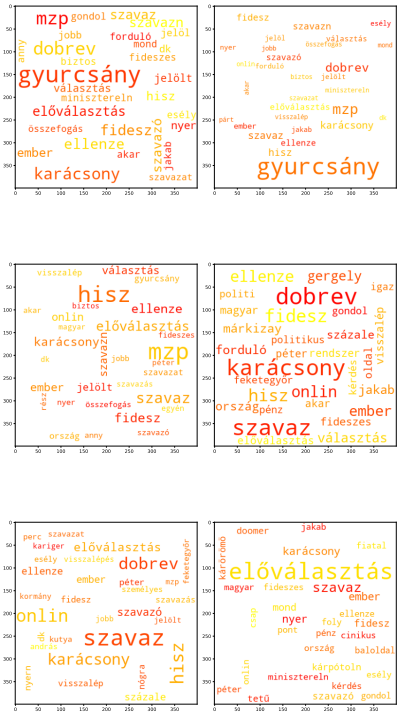

Figure 4.4: LSI: 15-40 words/comment, 6 components, stemmed words words

Figure 4.5: Word document bipartite network with hSBM , lemmatized words, 15-40 words/comment. The words are on the right side, the documents are on the left. Different colors correspond to the different groups.

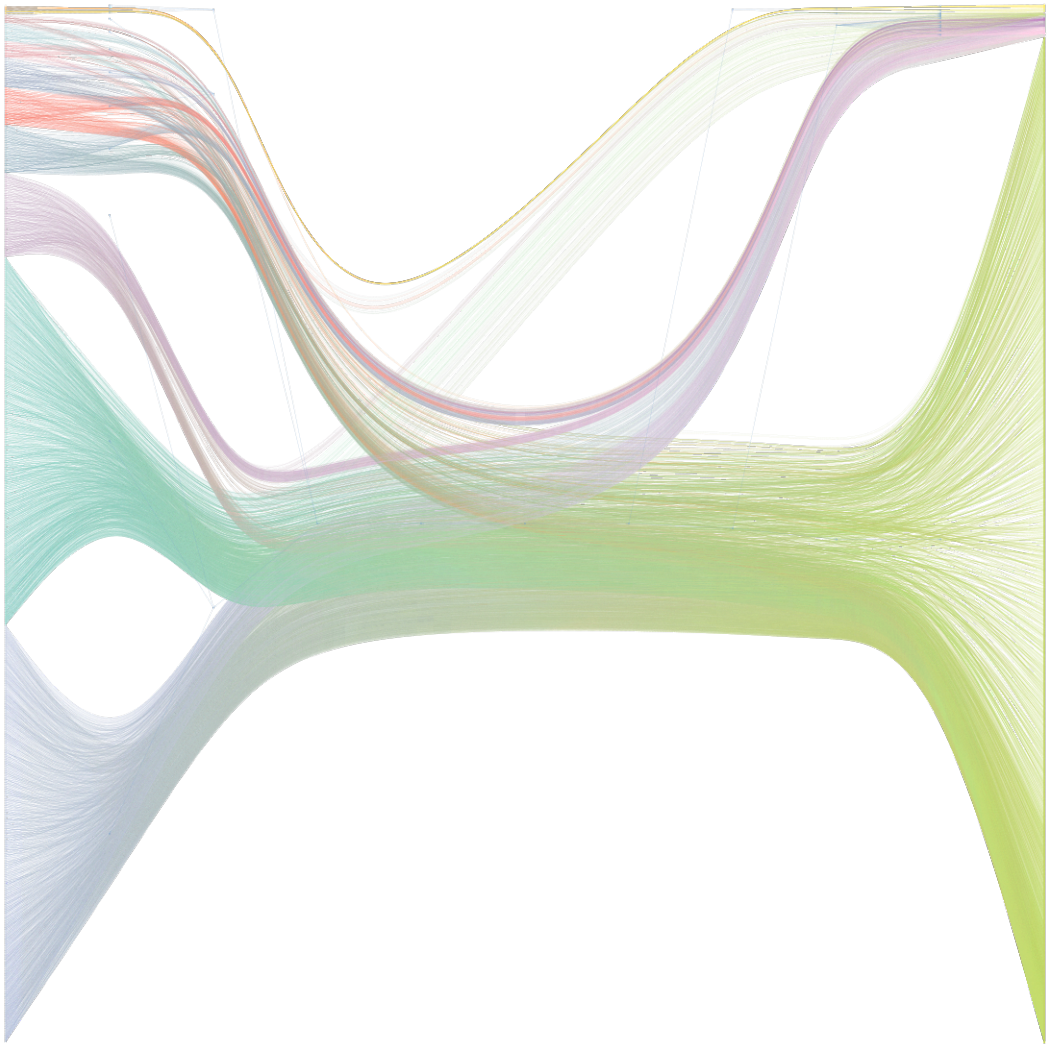

Figure 4.6: Word document bipartite network with hSBM, lemmatized words, 40-100 words/comment. The words are on the right side, the documents are on the left. Different colors correspond to the different groups.

	topic_0	value_0	topic_1	value_1	topic_2	value_2	topic_3	value_3	topic_4	value_4	topic_5	value_5
0	ember	0.0074	ország	0.0383	Fidesz	0.0957	Gergő	0.0943	mzp	0.1973	Csaba	0.0890
1	mond	0.0067	magyar	0.0269	Orbán	0.0482	délután	0.0849	karácsony	0.0959	momentumos	0.0681
2	akar	0.0058	politikai	0.0269	rész	0.0383	isten	0.0849	Jakab	0.0491	toth	0.0419
3	megy	0.0057	kérdés	0.0256	párt	0.0351	igaza	0.0660	Klára	0.0351	eleg	0.0366
4	dolog	0.0051	kormány	0.0253	tesz	0.0351	konfliktus	0.0566	esély	0.0351	Anett	0.0366
5	annyi	0.0038	igaz	0.0247	biztos	0.0319	nincsenek	0.0566	szavazó	0.0312	tamogat	0.0314
6	gondol	0.0038	politikus	0.0199	gyurcsány	0.0307	kérés	0.0566	forduló	0.0304	jambor	0.0262
7	tart	0.0031	politika	0.0193	teljes	0.0263	magad	0.0377	ellen	0.0195	bomba	0.0209
8	rendszer	0.0027	kerül	0.0158	hisz	0.0251	molnár	0.0377	jobbik	0.0195	peldaul	0.0209
9	több	0.0026	helyzet	0.0155	fideszes	0.0243	szimpla	0.0377	dobrevet	0.0164	jeloltnel	0.0209
10	szeret	0.0025	Magyarország	0.0155	sajnos	0.0231	zsarol	0.0377	mzpre	0.0164	tamogatja	0.0209
11	rossz	0.0023	oldal	0.0155	pont	0.0227	élét	0.0377	karigeri	0.0164	erdemi	0.0209
12	komoly	0.0023	erős	0.0155	nagy	0.0207	kínos	0.0283	látszik	0.0164	jeloltet	0.0209
13	amúgy	0.0023	miniszterelnök	0.0152	egyszerű	0.0199	felidéz	0.0283	plusz	0.0156	beszelni	0.0209
14	embere	0.0023	egyéb	0.0142	Péter	0.0195	felmutat	0.0283	visszalép	0.0156	kvazi	0.0209
15	beszél	0.0023	fontos	0.0136	mindenki	0.0163	állítottad	0.0189	szavazókat	0.0140	kepvisel	0.0157
16	vesz	0.0022	eset	0.0130	érdekel	0.0163	klubrádióban	0.0189	java	0.0133	dinoszocit	0.0157
17	szint	0.0021	képes	0.0117	szól	0.0159	beszélgetnünk	0.0189	ráadás	0.0133	kinjukban	0.0157
18	mennyi	0.0021	előre	0.0114	nyilván	0.0152	tompítani	0.0189	megnyer	0.0133	hadhazynak	0.0157
19	csinál	0.0020	probléma	0.0114	kezd	0.0148	kiszállnék	0.0189	mzpnek	0.0125	drukkol	0.0157

Table 4.1: Level 0 of hSBM topics, part 1, lemmatized, 40-100 words/comment

	topic_6	value_6	topic_7	value_7	topic_8	value_8	topic_9	value_9	topic_10	value_10	topic_11
0	szavaz	0.1382	dobrev	1.0	pénz	0.0963	szavazat	0.1379	online	0.2202	ellen\r
1	ellenzéki	0.0742	None	NaN	támogatás	0.0609	százalék	0.1073	szavazás	0.2083	alapja
2	dk	0.0681	None	NaN	forint	0.0609	megye	0.0843	személyes	0.1607	hajnal
3	jelöl	0.0660	None	NaN	milliárd	0.0472	körzet	0.0766	folyamat	0.0536	Dávid
4	választás	0.0654	None	NaN	állami	0.0354	lead	0.0498	szavazzon	0.0417	kihivo
5	nyer	0.0592	None	NaN	Budapest	0.0354	szavazz	0.0421	érdeklődés	0.0417	jelolt
6	ellenzék	0.0592	None	NaN	tanár	0.0295	pest	0.0383	voksok	0.0417	Anna
7	összefogás	0.0442	None	NaN	oktatás	0.0295	egyéniben	0.0307	szavazólapot	0.0238	peter
8	indul	0.0225	None	NaN	gazdaság	0.0236	kutyapárt	0.0307	bővít	0.0238	tunik
9	előválasztás	0.0218	None	NaN	külföldi	0.0216	nyugodt	0.0268	előzetes	0.0238	mucsi
10	közös	0.0204	None	NaN	piac	0.0177	szavazol	0.0268	várakozás	0.0238	Piroska
11	támogat	0.0197	None	NaN	biztosíték	0.0177	listás	0.0268	hétvége	0.0238	nehany
12	helye	0.0163	None	NaN	igény	0.0177	mkkp	0.0268	azonosítás	0.0238	lengyel
13	miniszterelnökjelölt	0.0163	None	NaN	olcsó	0.0157	választókerületében	0.0230	kapacitás	0.0179	olga
14	lista	0.0163	None	NaN	gyakorlat	0.0157	elment	0.0192	jutott	0.0179	Bernadett
15	előválasztáson	0.0150	None	NaN	euró	0.0138	listájuk	0.0192	ezúttal	0.0179	dorosz
16	országos	0.0136	None	NaN	fokozat	0.0138	szavazatok	0.0192	előválasztani	0.0179	Tibor
17	fideszre	0.0136	None	NaN	pedagógus	0.0138	leváltása	0.0192	ellenőrzött	0.0179	visi
18	jövő	0.0136	None	NaN	költ	0.0138	szavazatokat	0.0153	None	NaN	neve
19	szavaztam	0.0136	None	NaN	fizetés	0.0138	mkkpra	0.0153	None	NaN	kaman

Table 4.2: Level 0 of hSBM topics, part 2, lemmatized, 40-100 words/comment

	topic_0	value_0	topic_1	value_1	topic_2	value_2	topic_3	value_3	topic_4	value_4
0	ember	0.0074	mzp	0.0428	Fidesz	0.0957	dobrev	0.2590	pénz	0.0522
1	mond	0.0067	szavaz	0.0343	Orbán	0.0482	Csaba	0.0212	online	0.0394
2	akar	0.0058	karácsony	0.0208	rész	0.0383	momentumos	0.0162	szavazat	0.0384
3	megy	0.0057	ország	0.0205	párt	0.0351	Gergő	0.0125	szavazás	0.0373
4	dolog	0.0051	ellenzéki	0.0184	tesz	0.0351	délután	0.0112	forint	0.0330
5	annyi	0.0038	dk	0.0169	biztos	0.0319	ellen\r	0.0112	támogatás	0.0330
6	gondol	0.0038	jelöl	0.0164	gyurcsány	0.0307	isten	0.0112	százalék	0.0299
7	tart	0.0031	választás	0.0162	teljes	0.0263	vállal	0.0100	személyes	0.0288
8	rendszer	0.0027	ellenzék	0.0147	hisz	0.0251	toth	0.0100	milliárd	0.0256
9	több	0.0026	nyer	0.0147	fideszes	0.0243	alapja	0.0100	megye	0.0235
10	szeret	0.0025	politikai	0.0144	sajnos	0.0231	tapasztalat	0.0100	körzet	0.0213
11	rossz	0.0023	magyar	0.0144	pont	0.0227	hivatkozik	0.0100	állami	0.0192
12	komoly	0.0023	kérdés	0.0137	nagy	0.0207	erőszakos	0.0087	Budapest	0.0192
13	amúgy	0.0023	kormány	0.0135	egyszerű	0.0199	Anett	0.0087	oktatás	0.0160
14	embere	0.0023	igaz	0.0132	Péter	0.0195	eleg	0.0087	tanár	0.0160
15	beszél	0.0023	összefogás	0.0110	mindenki	0.0163	heti	0.0087	lead	0.0139
16	vesz	0.0022	Jakab	0.0107	érdekel	0.0163	hajnal	0.0087	gazdaság	0.0128
17	szint	0.0021	politikus	0.0107	szól	0.0159	igaza	0.0087	szavazz	0.0117
18	mennyi	0.0021	politika	0.0103	nyilván	0.0152	tamogat	0.0075	külföldi	0.0117
19	csinál	0.0020	kerül	0.0085	kezd	0.0148	tapasztal	0.0075	pest	0.0107

Table 4.3: Level 1 of hSBM topics, lemmatized, 40-100 words/comment

topic_0	value_0	value_0
0	mzp	0.0073
1	Fidesz	0.0069
2	dobrev	0.0060
3	szavaz	0.0059
4	ember	0.0052
5	mond	0.0047
6	akar	0.0041
7	megy	0.0040
8	dolog	0.0036
9	karácsony	0.0036
10	ország	0.0035
11	Orbán	0.0035
12	ellenzéki	0.0032
13	dk	0.0029
14	jelöl	0.0028
15	választás	0.0028
16	rész	0.0028
17	annyi	0.0027
18	gondol	0.0027
19	tesz	0.0025

Table 4.5: Table of topics on hSBM level 3, lemmatized words, 40-100 words/comment.

Chapter 5

Discussion

5.1 Topic distributions

In the longer comment corpus, we could detect more topics, as it can be seen on Table 4.1. If we take a look at the topic distributions we can see, that the most dominant topics were topic 0, 1 and 2. So in fact, we only have 3 major topics.

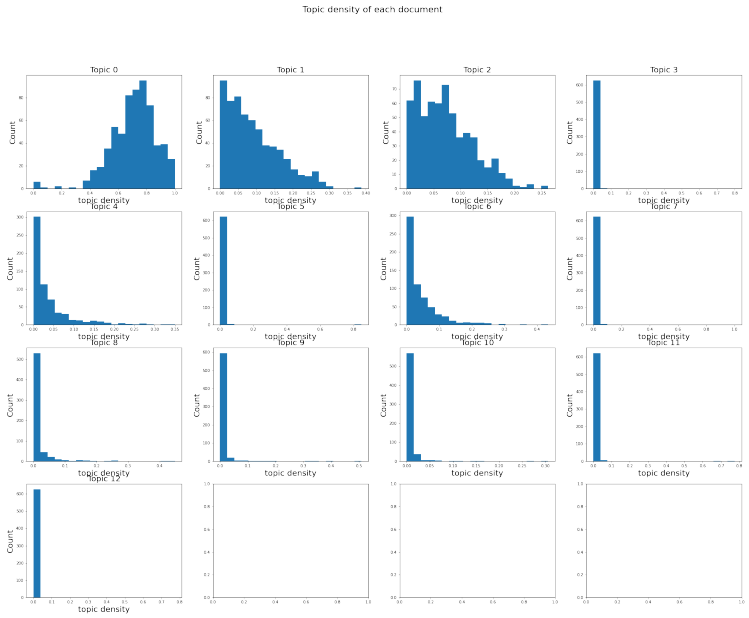

Figure 5.1: Topic densities with hSBM, lemmatized words, 15-40 words/comment.

Bibliography

[1] Twitter Premium API Search Docs
https://developer.twitter.com/en/docs/twitter-api/premium/search-api/overview

[2] *Nefelejcs Gergő*: Szervezhet-e a felmondók fóruma általános sztrájkot?
https://tett.merce.hu/2021/11/12/be-tudsz-ugrani-dugd-fel-magadnak-szervezhet-e-a-felmondok-foruma-altalanos-sztrajkot/

[3] *Icons from various websites*
https://www.svgrepo.com/
https://www.flaticon.com/
https://icons8.com/

[4] What is the difference between stemming and lemmatizing?
https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/

[5] Scikit-learn: Machine Learning in Python, *Pedregosa et al.* JMLR 12, pp. 2825-2830, 2011.
https://scikit-learn.org/stable/index.html

[6] Barbaresi A. (2021). Simplemma: a simple multilingual lemmatizer for Python. Zenodo.
http://doi.org/10.5281/zenodo.4673264
https://pypi.org/project/simplemma/

[7] *Databricks Academy*: Introduction to Latent Semantic Analysis
https://www.youtube.com/watch?v=hB51kkus-Rc&ab_channel=DatabricksAcademy

[8] *Zsuzsa Bakk*: Latent class analysis https://www.universiteitleiden.nl/en/research/research-projects/social-and-behavioural-sciences/stepwise-latent-class-analysis

[9] Wikipedia: Latent semantic analysis
https://en.wikipedia.org/wiki/Latent_semantic_analysis#Latent_semantic_indexing

[10] Wikipedia: Latent Dirichlet allocation
https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

[11] *Scott Sullivan*: LDA Algorithm Description
https://www.youtube.com/watch?v=DWJYZq_fQ2A&ab_channel=ScottSullivan

[12] Stochastic block models and probabilistic reductions - *Emmanuel Abbe*
https://www.youtube.com/watch?v=kMIxM8C0Ruo&ab_channel=InstituteforAdvancedStudy

[13] *Thamindu Dilshan Jayawickrama*: Community Detection Algorithms
https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae

[14] Gerlach, Martin, Tiago P. Peixoto, and Eduardo G. Altmann. "A network approach to topic models." Science advances 4.7 (2018): eaaq1360.

[15] Martin Gerlach et al.: hSBM_Topicmodel github repository
https://github.com/martingerlach/hSBM_Topicmodel

[16] graph-tool: Efficient network analysis with python
https://graph-tool.skewed.de/